# Protecting Web Services from DDoS attacks by SOTA

**Ashley Chonka, *Member,* IEEE, Wanlei Zhou, *Member, IEEE*, Yang Xiang, *Member, IEEE***

*Abstract*-- **In the area of SOA and Web Service Security, many well defined security dimensions have been established. However, current Web Security Systems (WS-Security for example) are not equipped to handle Distributed Denial of Service (DDoS) attacks. In this paper we extend upon our previous work on, Service Oriented Traceback Architecture (SOTA), in order to defend Web Services against such attacks. SOTA's main objective is to identify the true identity of forged messages, since an attacker tries to hide their identity, in which to avoid current defence systems and escape prosecution. To accomplish the main objective, SOTA should be attached as close to the source of the attack. When an incoming SOAP message comes into the router, it is tagged with our own SOAP header. The header can be used to traverse the network back to the true source of the attack. According to our experimental evaluations we find that SOTA is simple and effective to use against DDoS attacks.**

***Index Terms*-- Traceback, Service-Oriented Architecture (SOA), Service-Oriented Computing (SOC), Distributed Denial of Service.**

## I. INTRODUCTION

In recent events, a group called anonymous used a Distributed Denial of Service attack, to bring down a prominent website [30]. This attack is another example of the serious threat that DDoS poses to information infrastructures [6][7]. The main objective of a DDoS attack is to attempt to exhaust computer resources (CPU time, Network bandwidth etc) [8][9]. Another objective of DDoS, is for the attackers to hide their identity by mimicking a legitimate web service [10][13]. Organizations, through the use of Web Services, expose their core elements over the Internet, via the use of Extensible Markup Language (XML) in conjunction with HTTP and SMTP. With this exposure, organizations open themselves up to those who have a malicious intent.

Current security for web services encompasses the areas of integrity, confidentiality and availability [1][2]. WS-Security [3], XML-Signature [14], XML-Encryption [15] employ these areas. These standards work in conjunction with Simple Object Access Protocol (SOAP) [5]. From these developments, a new standard for Web Security has emerged, called Security Assertions Markup Language (SAML) [16][18]. The major problem of these security standards is the focus on protecting message content, and not on the message itself [4]. The paper by Jenson et. al. [1] discusses the depth of this problem.

Our contribution in this paper is to expand upon our previous research [28], in adopting a product-neutral approach, called Service Oriented Traceback Architecture (SOTA). SOTA can be used to prevent DDoS and XDoS (XML based DoS) attacks on Web services. Current Web Security Services show, that new enhancements are needed against the current flow of attacks. SOTA provides the resources to traceback through the network, so that the true source of DDoS attack is identified. Upon the discovery of the identity of an attacker, the appropriate preventive mechanisms can be triggered, like using firewalls to filter out attack messages. The remainder of the paper is made up of the following: Section 2 reviews the related work on Web Security Services. Section 3 covers the details of our SOTA framework. Section 4 presents our experiments and performance evaluation. Lastly, Section 5 provides our conclusions

## II. PRCEDURE FOR PAPER SUBMISSION

Service-Oriented Computing (SOC) utilises services as the cornerstone for developing Web Application solutions. With DDoS attacks occurring on a daily basis [11][12][19], attackers have discovered how easy it is to disrupt web services. In this section we briefly discuss two defense systems that have been developed to handle Web Based DDoS attacks, and their problems in dealing with DDoS.

### *A. Current Web Service Defense Systems*

Ye et al. [27] proposed a SOA approach to handle DDoS attacks. Their Service Hub is built upon Web Services and placed in between the client and the service provider. It contains two modes, a normal and an attack mode. The messages go through to the service provider in normal mode. In attack mode, the Service Hub authenticates messages, authorizes it and passes it onto the service provider. The main problem with this system is that it is incapable of handling a reflective attack [20]. The second problem with the Ye's system is that authenticator

This work was supported by the ARC Linkage grant (Project number LP0562156).

Mr A. Chonka is a PhD candidate at the School of Engineering & Information Techonology, Deakin University, Waurn Ponds, Australia, (e-mail: ashley@deakin.edu.au).

Prof W. Zhou is currently the Chair Professor of Information Technology and the Associate Dean (International), Faculty of Science and Technology, Deakin University, Melbourne, Australia. (e-mail: wanlei@deakin.edu.au).

Dr Y. Xiang is currently with School of Management and Information Systems, Central Queensland University. (e-mail: y.xiang@cqu.edu.au).

**ICITA2008 ISBN: 978-0-9803267-2-7**

can be spoofed with a forged legitimate user id. The Padmanabhuni et. al. [17] framework is another Web Security System. Its main task is to detect and filter out XDoS attacks against web services. Their framework focuses on validating XML, in order to authenticate legitimate users. An XML message, with the forged id of a legitimate user, can be used to get around this defence.

## III. SOTA FRAMEWORK

### A. SOTA Description

In our previous paper [28], we cover SOTA in-depth, so in this section we briefly cover our model. SOTA is a web security service application that is product-neutral. Its main objective is to apply a SOA approach to traceback methodology, in order to identify a forged message id, since one of the main objectives of DDoS is hide the attacker's true identity. Figure one displays where SOTA is located within the network. The basis of SOTA is founded upon the Deterministic Packet Marking (DPM) [23] algorithm. DPM marks the ID field and reserved flag within the IP header. As each incoming packet enters the edge ingress router it is marked. The marked packets will remain unchanged as they traverse the network. Outgoing packets are ignored. DPM methodology is applied to our SOTA framework, by placing the Service-Oriented Traceback Mark (SOTM) within web service messages. If any other web security services (WS-Security for example) are already being employed, SOTM would replace the 'token' that contains the client identification. Real source message identification are stored within SOTM, and placed inside the SOAP message. SOTM, as in DPM tag, will not change as it traverses through the network. The composition of SOTM is made up of one XML tag, so not to weigh down the message, and stored within a SOAP header. Upon discovery of a DDoS attack, SOTM can be used to identify the true source of forged messages.

SOTA does not directly eliminate a DDoS attack message; this is left for the filter section of a defense system (Firewalls). Instead SOTA main goal is to deal with one of the two main objectives of DDoS, which is the forging the id. Spoofing an ID is done for two reason, these are: exploit a known vulnerability, in order to bring down system. These vulnerabilities could be found in communication channels (flooding for example) or known exploits within the services provided (for example, an attacker can Overload their messages, which will result in the web server crashing). The second reason is that attackers try to hide their identity. The reasons vary for this second reason, which depends on what type of attack, but usually it is to cover their crime or to bypass a known defense that is in place to prevent it. It is with this second objective that SOTA attempts to cover, as other traceback methods, like Probability Packet Marking (PPM) [21][22] and DPM.

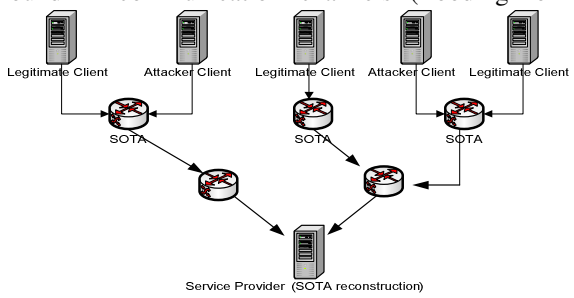


**Figure 1. SOTA from the network service prospective**

There are many reasons for to employ a SOTA type framework, these are:

- Current web security is not up to handling an XDoS or DXDoS attack. In fact, as Jension et al. shows how WS-Security can be used in an XDoS attack.
- With IPv6 coming into fruition [29], current IP traceback methods will no longer be viable. This is due to the changes that IPv6 introduces, such as, IPSec and the packet header format no longer holds support the fields that are required for IP traceback.
- SOTA does not violate IP protocols, in order to store information for traceback purposes.

Using the SOA model, SOTA can be employed on any ubiquitous grid system.

### B. SOTA approach to SOA

SOA organizes the infrastructure into a set of interacting services for SOC. There are a number of basic properties and services [24] contained in SOTA. These characteristics are as follows [25]:

- Loosely Coupled – SOTA is made from the XML base language. This means that it can be run on different platforms, regardless of the programming language.
- Message based interaction – The interaction between the client, SOTA, and service provider are all message based.
- Dynamic Discovery – WSDL is attached to SOTA so that all services are known to the public. This means that any client can connect to SOTA at any time over the internet.
- Late Binding – SOTA and the service provider all run in real-time. This allows clients to access services anytime.
- Policy based behavior – SOTA aligns itself with WS-Security Policy. It also implements its own policy called SOTA-Policy. This policy dictates what messages are marked.

## IV. Performance Evaluation

### A. Simulation Setup

Experiments were carried out to evaluate the performance of the SOTA system. These experiments were performed on a Dell Dimension DM501 Intel Pentium single-core CPU, 3.0 GHz, 2 GB of RAM and 2 300GB SATA hard-drives. All our programs were implemented with .NET Web Services with the use of VB.Net. Figures 2 and 3 display the algorithms used to insert

```
SOTM procedure at SOTA, edge Interface I

For each incoming request message w
If no header then
   create SoapHeaderAttribute("client id")
   Invoke Header new SoapHeaderAttribute
Else
   get WSSusernameToken(xx)
   WSSusername = new client id
```

**Figure 2. Pseudo Code to extract Header information**

```
Identification reconstruction procedure at web server
For each message request w from source Sx
        Create a table array
        Ws.tx = extract Transactioninfo()
        Ws.tx.time_and_date = timestamp
        Ws.tx.usernameId = usernameID
        Table_array[]+ = Ws.txusernameID
End
Display of username at particular time of the attack
For each Table_array[]
        Get what time of attack
        Get usernameID from Table_array[]
Display usernameID
```

**Figure 3. Pseudo Code to extract, store and display username identification**

and extract the SOTM tag.

The experiments we conducted were broken up into two groups. The first group of experiments compared SOTA against SOAP authentication and WS-Security. The second group of experiments simulated XDoS attacks against the service provider. We selected to simulate the oversize payload, SOAPAction spoofing and XML injection from the Jenson et. al [1] paper.

### *B. Assumptions used for our experiments*

The following assumptions made about our first group of experiments are that:

- An attacker may control any number of client machines that are widely distributed across the Internet.
- Attackers might know that they are being traced.
- It only takes a few messages to get to the SOTA reconstruction for a traceback to begin.
- SOTA has not itself been compromised by the attackers.
- That the service provider of web service has limited resources.
- SOAP headers are being used by the client.
- Real Source ID is the location of the edge router.

With the second group of experiments we decided to simulate three XDoS attacks. The reason for the simulations is due to the legality of implementing such attacks. For simulating message passing, we generated 20 messages within our code. 5 of these messages were selected randomly to represent the attack. To simulate the success of one attack, we introduced a 50/50 chance that the message might crash the web-service. If the web server did not crash, the service provider was able to trace the message source and initiate filtering procedures. However, if the attack was successful no more messages will be generated. Upon the web server crash, we assume the service provider would restart it. Upon the restart, the service provider would access SOTA reconstruction, find the source of the attack and filter the messages out.

### *C. Evaluations of the first group of experiments*

In our first experiment we developed a basic SOAP Web Service using .Net and VB.Net. The program contained a basic header for authentication purposes. To simulate SOTA, the program extracted the name id from the header and replaced with the real user id (010101). It is assumed that a one-way transmission delay between client and Web Server is 10ms. The delay is simulated by the program going into a wait mode for 10secs, and is added to the response time data. The measurements we used in this experiment were the processing time over the response time. The result shown in figure 4, was that over a 2 seconds of processing time, SOTA was far more effective then the SOAP authentication procedure. One of the reasons for this is because of a quicker response time, due to SOTA swapping the tag. Having the extra response time will lead to a reduction of computer resources during a DDoS or XDoS attack.

In our second experiment, we ran a WS-Security interaction application against Amazon Elastic Compute Cloud (Amazon EC2) [26]. The WS-Security application contained a signed certificate for authentication purposes. SOTA, in this experiment, was to exchange the username id for the authentication name. This was done before it was sent to the Amazon SOAP service, to ensure that the message would be received and that we got a response. The results are based on how long the application had taken to process a response from Amazon. SOTA was used in conjunction with WS-Security, in

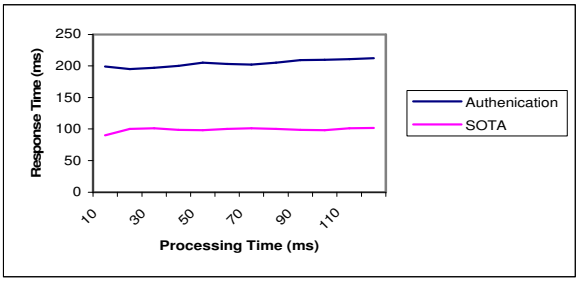


**Figure 4. Results of SOAP Authentication and SOTA**

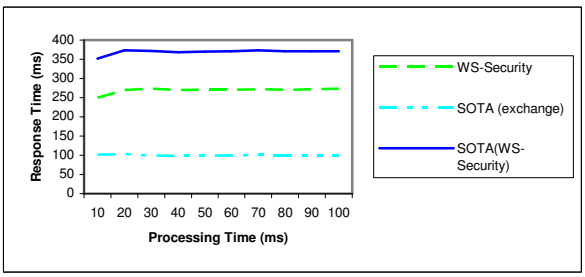


**Figure 5. Results of WS-Security, SOTA(exchange) and SOTA(WS-Security)**

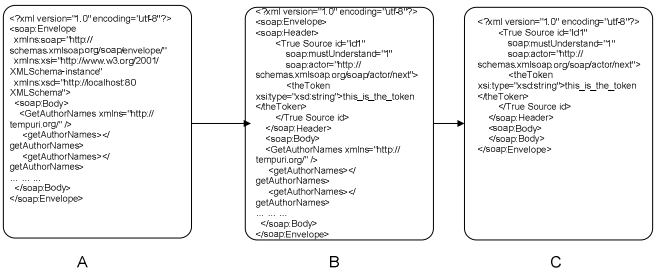


**Figure 6. Event Descriptor Graph for Oversize payload attack (Client attack message (A), SOTM tag (B), True identity of the message, requested by the service provider (C).**

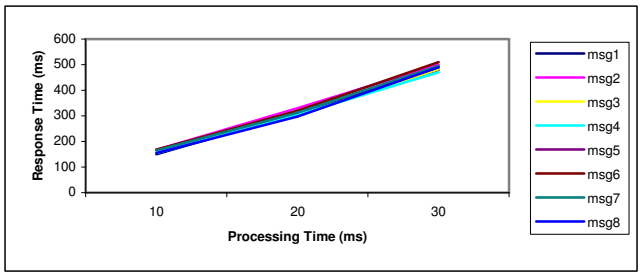


**Figure 7. Messages generated by our first simulation (Oversize Payload).**

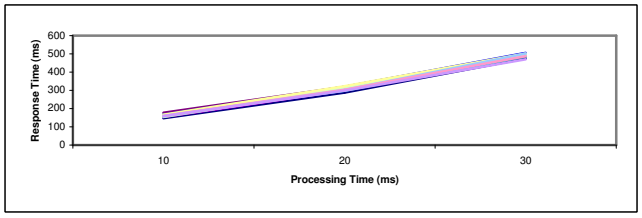


**Figure 8. 5 attack messages, out of the 20 generated, were removed after traceback and filter protocols (Overaize Payload Attack).**

order to replace the name id for the real-source id. The results show in figure 5, by introducing SOTA into WS-Security, an increase in response time was up by thirty percent. This increase means that during a DDoS attack, more processing time is required to handle the extra burden. The benefits of taking on this extra burden are: the true identification maybe found and additional integrity is applied to the message.

Also in figure 5, we see a comparison between WS-Security and SOTA (exchange). According to the results, WS-Security is over twice the response time, shown in figure 9. The reason for the increase was due to WS-Security having to build a security token. This token was placed in the message before it was sent to the Amazon Web Server. Upon the receipt of the token, Amazon tested the authentication of the message. However, in comparison, SOTA only has to exchange the identification information. Assuming Amazon had SOTA on their system. Traceback to the source of attack could occur instead of just authenticating the message.

### *D. Evaluations of the Second group of experiments*

The second group of experiments consists of implementing three XDoS attacks. The first of these is the oversize payload attack. Its objective is to exhaust web service resources. Following Jenson et. al [1] in the construction of this attack,

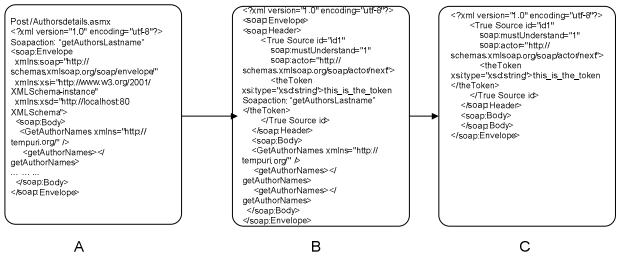


**Figure 9. Event Descriptor Graph for SOAPaction attack (Spoofed SOAPAction message (A), SOTM tag (B), True identity of the message, requested by the service provider (C).**

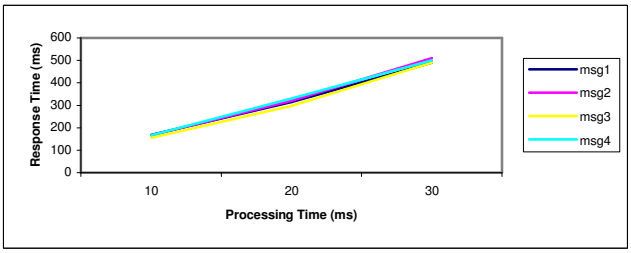


**Figure 10. Messages generated by our first simulation (SOAPAction attack)**

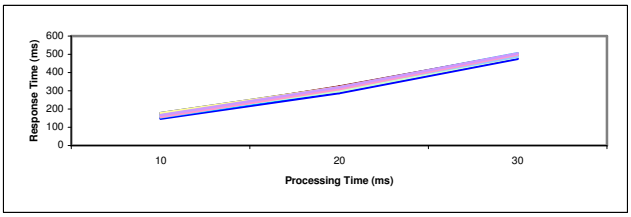


**Figure 11. 5 attack messages, out of the 20 generated, were removed after traceback and filter protocols (SOAPAction attack).**

we developed an oversize payload message (see Figure 6a). Figure 7 displays the messages that our simulation generated. As each message passed through SOTA it was marked with a SOTM tag (see figure 6b). Further, we can see from figure 7 that the messages stop at Msg8, this means that an attack was successful. The service provider, in the light of a successful attack would initiate the following procedures: Restart the system, search SOTA reconstruction for the true source of the attack (see figure 6c), and instigate filtering protocols (See figure 8). To simulate these procedures, we restarted the program to generate 20 more messages. With the traceback and filtering controls in place, we found 5 attacks and 15 normal messages (figure 8).

The next simulation was a spoofed SOAPaction attack. It invokes an operation that is different within the SOAP body, and usually results in a web server crash. Figure 9a displays our spoofed SOAPAction message used in this simulation. The message contains within the SOAPAction the author's first name, but only the author's last name is within the SOAP body. This message composition could result in the server behaving erratically or crashing it. Figure 10 displays the messages that our simulation generated. As each message passed through SOTA it was marked with a SOTM tag (see figure 9b). Further, we can see from figure 14 that the message stops at msg3, this means that an attack was successful. The service

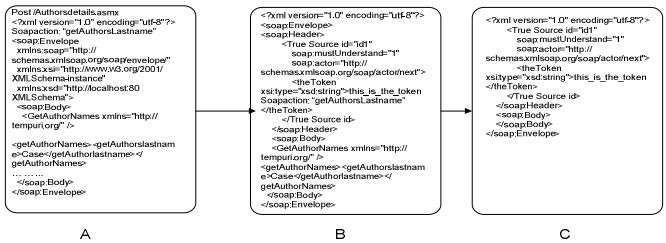


**Figure 12. Event Descriptor Graph for XML injection attack (Spoofed XML message (A), SOTM tag (B), True identity of the message, requested by the service provider (C).**

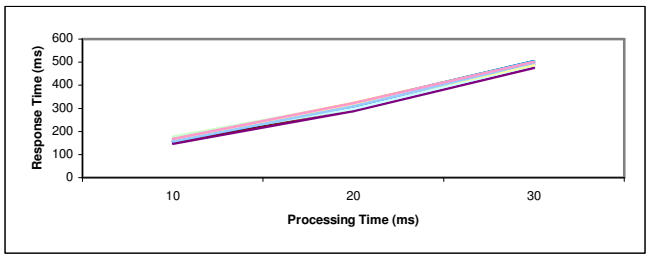


**Figure 13. 4 attack messages, out of the 20 generated, were removed after traceback and filter protocols (XML Injection attack).**

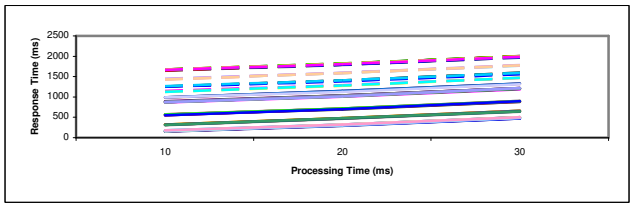


**Figure 14. 84 normal messages were processed. 9 floods were successful in crashing the system. 7 attacks message were filtered.**

provider will instigate the following procedures: Restart the system, search SOTA reconstruction for the true source of the attack (see figure 9c), and instigate filtering protocols (See figure 11). To simulate these procedures, we restarted the program to generate 20 more messages. With the traceback and filtering controls in place, we found 5 attack and 15 normal messages (figure 11).

An XML Injection attack was our last simulation of the 3 XDoS chosen. This attack tries to modify the XML structure of our SOAP message. Figure 12a shows that the authorname tag has another tag within it called authorlastname. The result of this message could lead to a server crash, though it is unlikely. Instead, as shown in figure 12a, the content has been changed. This content change, would lead to incorrect information, being displayed from the tag. Figure 13 displays the messages that our simulation generated. As each message passed through SOTA it was marked with a SOTM tag (Figure 12b). The result of the XML injection attack, shown in Figure 13, is that 4 attack messages were filtered. The first attack message signaled the service provider to instigate SOTA reconstruction. With the discovery of the attacker id, the service provider was able to filter out the rest of the attack messages.

The final simulation we conducted was a message flood attack, using XML Injection. The simulation program was setup to generate a total of 100 messages. If one of those messages was an attack, it had 50/50 chance to crash the system. If the system did crash, a number between 100 and 300 ms was added to the next lot of response time. This was to simulate the time taken by the service provider to restart their system, locate the source, and filter it. From our results, we got 84 normal messages. Further, was the unusually high, 9 successful attacks that crashed the system. The reason for the crashes was due to the chance nature built within our code. These successful attacks are displayed by the groupings within figure 14. Of the attacks that got filtered, 7 attacks messages were discovered.

## V. Conclusion and Future Work

This paper builds upon our previous paper [36], in which identifies the real source of DDoS attacks. SOTA is a traceback system that is constructed on the basis of Web Services. Loose Coupling, Policy Based, Message Based and Dynamic discovery are some of criteria employed by the SOTA framework. The empirical data from our experiments shows that SOTA is efficient and effective. The experimental data also shows that SOTA is able to traceback to the source. Once an attack has been discovered and the attacker's identity known, counter measures can be initiated. The people, who will be interested in this research, are those that want to their protect web services in a cheap and efficient manner. In the future, we will build a filtering application and extend SOTA to protect grid networks.

## References


[1] Jensen, M., Gruschka, N., Herkenh¨oner, R., and Luttenberger, N., (2007), " SOA and Web Services: New Technologies, New Standards – New Attacks" Fifth European Conference on Web Services, 0-7695-3044-3/07, 2007.

[2] Bishop, M, 'Computer Security', Addison Wesley, 2003

[3] Nadalin, A., Kaler, C., Monzillo, R., and Hallam-Baker. P., (2008), 'Web Services Security: SOAP Message Security 1.1 (WSSecurity 2004)', http://docs.oasis-open.org/wss/v1.1/, 2008.

[4] Secure Socket Layer (SSL), (2008), http://en.wikipedia.org/wiki/Secure_Sockets_Layer

[5] SOAP 1.1, (2008), http://www.w3.org/TR/soap/

[6] Bouzida, Y.; Cuppens, F.; Gombault, S., (2006), '*Detecting and Reacting against Distributed Denial of Service Attacks*' Communications, 2006 IEEE International Conference on Volume 5, June 2006.

[7] Trostle, J, (2006), '*Protecting Against Distributed Denial of Service (DDoS) Attacks Using Distributed Filtering*', Securecomm and Workshops, 2006 Aug. 28 2006-Sept. 1 2006 Page(s):1 – 11.

[8] Poulsen. K., (2004), '*FBI Busts Alleged DDoS Mafia*', 2004. http://www.securityfocus.com/news/9411.

[9] Pappalardo, D., and Messmer, E., (2005), '*Extortion via DDoS on the rise, NetworkWorld*', May 2005. http://www.networkworld.com/news/2005/051605-ddos-extortion.html.

[10] Bhaskaran, M., Natarajan, A.M. and Sivanandam, S.N., (2007), '*Tracebacking the Spoofed IP Packets in Multi ISP Domains with Secured Communication*' IEEE - ICSCN 2007, MIT Campus, Anna University, Chennai, India. Feb. 22-24, 2007. pp.579-584.

[11] Digital Money, (2008), 'C-Gold Chat Forum Crash', http://www.digitalmoneyworld.com/ , 11 January, 2008.

[12] SE-NSE Forums, (2008), http://forums.se-nse.net/index.php , 10 January, 2008.

[13] Trostle, J, (2006), '*Protecting Against Distributed Denial of Service (DDoS) Attacks Using Distributed Filtering*', Securecomm and Workshops, 2006 Aug. 28 2006-Sept. 1 2006 Page(s):1 – 11.

[14] XML –Signature, (2008), 'XML-Signature Syntax and Processing' http://www.w3.org/TR/xmldsig-core/

[15] XML- Encryption, (2008), 'XML-Signature Syntax and Processing' http://www.w3.org/TR/xmlenc-core/
[16] Salz, R., (2005) "*Essential XML Web Services Security Practices*", http://www.idealliance.org/papers/dx_xml03/papers/05-2/05-04-02.pdf
[17] Padmanabhuni, S.; Singh, V.; Senthil kumar, K.M.; Chatterjee,A.Web Services, 2006, "*Preventing Service Oriented Denial of Service (PreSODoS): A Proposed Approach*", ICWS apos;06. International Conference on Volume , Issue , Sept. 2006 Page(s):577 – 584
[18] Oasisopen.com, (2008), 'Security Assertions Markup Language (SAML)', http://www.oasisopen.org/committees/tc_home.php?wg_abbrev=security , (2008)
[19] Prolexic Technologies, 'Prolexic Technology Report,(2007), http://www.prolexic.com/zr/zombie_july_2007.pdf
[20] He, Y., Chen, W., Peng, W., and Yang, M., (2005), "*Efficient and Beneficial Defense Against DDoS Direct Attack and Reflector Attack*", ISPA, 2005, LNCS 3758, pp 576- 587.
[21] Adler, M, (2002),'Tradeoffs in Probabilistic Packet Marking for IP Traceback,' *Proc. 34th ACM Symp. Theory of Computing*, ACM Press, 2002, pp. 407–418.
[22] Peng,T., Leckie, C., and Kotagiri, R., (2002), 'Adjusted Probabilistic Packet Marking for IP Traceback', Networking 2002.
[23] Belenky, A.,and Ansari, N., '*Tracing Multiple Attackers with Deterministic Packet Marking (DPM)*', Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing.
[24] Papazoglou, M.P., (2003), 'Service-Oriented Computing: Concepts, Characteristics and Directions', Proc of the Fourth International Conference on Web Information Systems Engineering (WISE'03), 2003
[25] Aiello, M and Dustdar, S, (2006), 'Service-Oriented Computing: Service Foundations', Dagstuhl Seminar Proceedings,05462, http://drops.dagstuhl.de/opus/volltexte/2006/528
[26] Amazon.com, (2008), 'Amazon Elastic Compute Cloud', http://aws.amazon.com/ec2
[27] Ye, X And Singh, S, [2007], '*A Soa Approach To Ddos Attac*ks', IEEE International Conference On Web Services (ICWS 2007), pp. 567-574, 2007
[28] Chonka,A., Zhou, W., and Xiang, Y., (2008), "Protecting Web Services with Service Oriented Traceback Architecture", *IEEE 8th International Conference on Computer and Information Technology*, IEEE, 2008.
[29] Van Beignum, I, (2008), '*IPv6: coming to a root server near you*', ARS technical, http://arstechnica.com/news.ars/post/20080102-icoann-to-add-ipv6-address-for-root-dns-servers.html , 02 January, 2008
[30] Brett, (2008), '*Anonymous, scientology, and the story that the media is too afraid to tell*', http://www.associatedcontent.com/article/612153/anonymous_scientology_and_the_story.html , 20 February, 2008