Protecting Web Services with Service Oriented Traceback Architecture

Ashley Chonka, Wanlei Zhou
School of Engineering & Information Technology
Deakin University
Geelong, 3220, Australia
{ashley, wanlei}@deakin.edu.au

Yang Xiang
School of Management and Information Systems
Central Queensland University
Rockhampton, 4702, Australia
y.xiang@cqu.edu.au

Abstract

Service Oriented Architecture (SOA) is a way of reorganizing software infrastructure into a set of service abstracts. In the area of applying SOA to Web Service Security, there have been some well defined security dimensions. However, current Web Security Systems, like WS-Security are not efficient enough to handle Distributed Denial of Service (DDoS) attacks. Our new approach, Service Oriented Traceback Architecture (SOTA), provides a framework to be able to identify the source of an attack. This is accomplished by deploying our defence system at distributed routers, in order to examine the incoming SOAP messages and place our own SOAP header. By this method, we can then use the new SOAP header information, to traceback through the network the source of the attack. According to our experimental performance evaluations, we find that SOTA is quite scaleable, simple and quite effective at identifying the source.

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

1. Introduction

Service-Oriented Computing (SOC) is today's solutions for developing web services. By utilising web services, organizations expose their core elements over the Internet via the use of Extensible Markup Language (XML). This had lead corporations and businesses in general, to earn huge profits. One of the most serious threats to these organizations, comes in the form of a Denial-of-Service (DoS) and its larger counterpart Distributed Denial-of-Service (DDoS) attack [5][6]. The two main objectives of these attacks are, to exhaust computer resources (CPU time, Network bandwidth) so that it makes services unavailable to legitimate users. The second objective, is to hide their identity by mimicking legitimate web service traffic, in order to create a large group of agents to launch an attack [9][12].

These types of attacks, at least 4000 according to the Prolexic Zombie Report 2007, happen on a daily basis [17]. Some of the reasons why a person would use a DoS or DDoS attacks are, competitive advantage [7], extortion of online business [8] and Employee Vilification [9]. Some recent DDoS attacks brought down the C-Gold Chat Forum website [10] and SE-NSE Forums [11].

Current security for Web Services using the Service-Oriented Architecture (SOA) encompasses the areas of integrity, confidentiality and availability [1][2]. Any breaches or violation on these areas is considered an attack on the network. Some of the most important specifications that address web service security are WS-Security [3], XML-Signature [13] and XML-Encryption [14]. These standards work in conjunction with Simple Object Access Protocol (SOAP) [4] specifications, in order to provide web service security regardless of the transport protocol. Some of the web security applications that employ this specification are, WS-Security and Security Assertions Markup Language (SAML) [15][16].

In this paper, our contribution is to adopt a product-neutral approach, in order to prevent DDoS and XML based DoS (XDoS) attacks on web services. Based on our observations of current web security services, new enhancements are needed to handle the current flow of attacks on web services. We propose a Service Oriented Traceback Architecture (SOTA) framework, to leverage existing security infrastructure. The framework provides a way to enable a traceback through the system, in order to identify the true identity of the attack. Upon the discovery of this identity, appropriate preventive mechanisms can be triggered, updating your firewall to filter the traffic for example. The remainder of this paper is made up of as follows. Section 2, covers the related work on web security services and IP traceback. The details of our SOTA framework are introduced in Section 3. Section 4, we present our experiments and performance evaluation

and Section 5, provides our conclusion and future work.

2. Related Work

With the severity of DDoS attacks occurring on a daily basis [10][11][17], attackers have discovered how simple and easy it is to disrupt web services. In this section we briefly cover current Web Security systems and there problems in dealing with DDoS. Lastly, IP traceback methods are covered.

2.1. WS-Security

One of the most important providers of security, in regards to web services, is WS-Security. WS-Security collaborates with SOAP specifications, in order to extend a SOAP message, by placing a so-called security header into it. This header is made of parts that have been defined by WS-Security Policy [18]. Through this policy, the web security services of integrity, confidentiality and authentication are defined and agreed upon by client and server. WS-Security employs the use of XML Signature and XML encryption. With their use, an encrypted XML fragment is placed inside the SOAP header, which provides security for the data to remain confidential. Based on our observations, we see little compatibility between the objectives of DDoS attack (Deprive legitimate users and cover the tracks of the attacker) and security protection provided by WS-Security (Protect SOAP message content). To further our claim, we can demonstrate a very simple way of how an attacker can get around WS-Security. One of the ways an attacker can cover their identity is by stealing a known legitimate user's id (Spoofing). With this spoofed message, the attacker can send an oversize message to crash the server, or just simply keep pumping the messages to the victim until their server crashes due to congestion [19].

2.2. IP Traceback

IP traceback schema can be categorized into two main areas, proactive and reactive [21]. Reactive traceback respond to an attack instead of trying to prevent the attack. This means it must be active while the attack is going on, otherwise it can not react to a DDoS attack. Link testing methods such as Control flooding [22] and Input debugging [23] are examples of reactive schemas. One of the problems with reactive schemas is that they require ISP co-operation for them to work, which most are reluctant to do, due to competitive reasons. In contrast, proactive schemas are actively recording trace information as packets transgress the network. With the recording information, the victim can reconstruct the path the packets had taken and subsequently identify the source of the attack. Some examples of proactive schemas include messaging [24][25], logging [26][27] and packet marking [28][29]. The problem with proactive schemas is that they require a great deal of resources to be committed to record and store the collected data. This makes them complex, complicated and redundant to use, especially if each router or server is required to log mostly the same information.

3. SOTA Framework

3.1. Introduction

Service-Oriented Traceback Architecture (SOTA) main objective is to apply a SOA approach to traceback methodology, in order to identify the true source of a DDoS. SOTA is based upon a popular form of packet marking called Deterministic Packet Marking (DPM) [30]. DPM is a packet marking algorithm that marks the ID field and reserved flag within the IP header. As each incoming packet enters an edge ingress router it is marked, outgoing packets are usually ignored. The marked packets will remain unchanged for as long as the packet traverses the network. We propose, in a SOTA framework, to employ some of the DPM methodology by placing our own Service-Oriented Traceback Mark (SOTM) within a web service message. If current web security services are being employed already, SOTM will replace the 'token' that contains the client identification with its own. The SOTM tag contains the real source identification, which is then placed inside the SOAP message, as the message enters the edge router. This tag will not change as it traverses through the network. With this SOTM tag, the victim of a DDoS attack will be able find the true source of the DDoS attack and filter it.

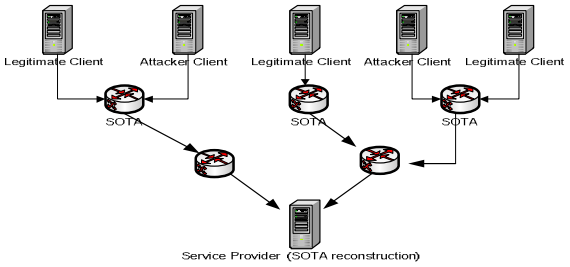

Figure 1. SOTA from the network service prospective

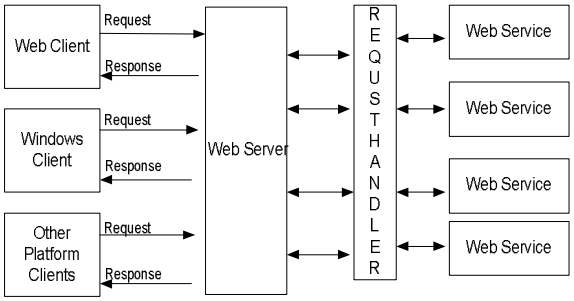

Figure 2. Web Service Architecture without SOTA

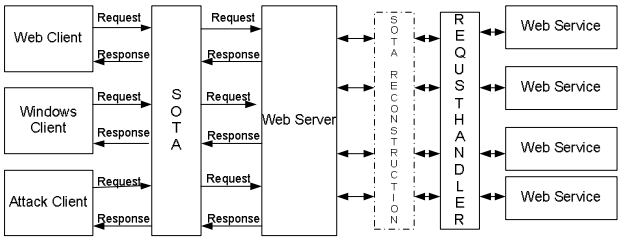

Figure 3. Web Service Architecture with SOTA defence system

3.2. SOTA Description

SOTA is deployed at the edge routers in order to be closer to the source end of the network. Figure one gives a network service display of where SOTA is placed. Usually if no security services are in place for web services, as seen in figure 2, then the system is quite vulnerable to attacks. SOTA remedy's this by being located before the Web Server, figure 3, in order to place a SOTM tag within the SOAP header. This is accomplished by attaching the Web Service Definition Language (WSDL) to SOTA instead of the web server. As a result, all service requests are first sent to SOTA for marking. Some of the consequence of placing SOTA before the web server are, we effectively remove the service providers address and prevent a direct attack. If an attack is discovered or was successful at bringing down the web server, the victim will be able to recover the SOTM tag and reconstruct where the attack came from.

In an attack scenario, the attack client will request a web service from SOTA, which in turn will pass the request to the web server. The attack client will then formulate a SOAP request message based on the service description formulated by WSDL. Upon receipt of SOAP request message, SOTA will place a SOTM within the header. We assume that WS-Security services are not being used. Otherwise the SOTM would replace the 'wsse' username tag with its own username tag. Once the SOTM has been placed, the SOAP message will be sent to the Web Server. Upon discovery of an attack, the victim will ask SOTA

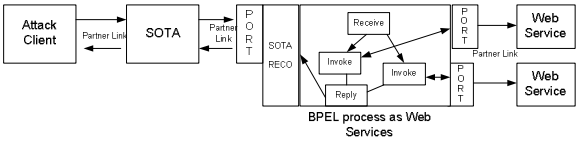

Figure 4. SOTA attached to the BPEL process

reconstruction to extract the mark and inform them of the message origins and begin to filter out the attack traffic. If the message is normal, the SOAP message is then forwarded to the request handler for processing. Upon the receipt of the SOAP request the Web Service will prepare a SOAP response. The Web Server then takes the SOAP response and sends it back to the client as part of the HTTP response. SOTA will not interfere with response requests or any outgoing message.

A practical example of where SOTA can be applied is the Business Process Execution Language (BPEL) [21] process. BPEL, is the favourite candidate for becoming the predominant of the Web Service composition standards. Figure 4 gives a display of BPEL process with the implementation of the SOTA framework system. SOTA reconstruction (SOTA RECO) is located at between the port and BPEL process in order to reconstruct the path back to the attack client.

3.2.1 SOTA approach to SOA

SOTA has the number of basic properties and characteristics of the service model [31]. These characteristics are as follows [32]:

- 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 and access the services of the service provider.
- Late Binding – SOTA and the service provider all run in real-time. This allows clients to access services when and wherever they are.
- Policy based behaviour – We plan to implement our own policy called SOTA-Policy, in the future, which will follow the WS-Security Policy. This policy will dictate messages marking procedures.

SOTA acts like a service broker within a SOA model (see figure five). Service Brokers are a repository for service descriptions, such as WSDL or Universal Description, Discovery, and Integration (UDDI).

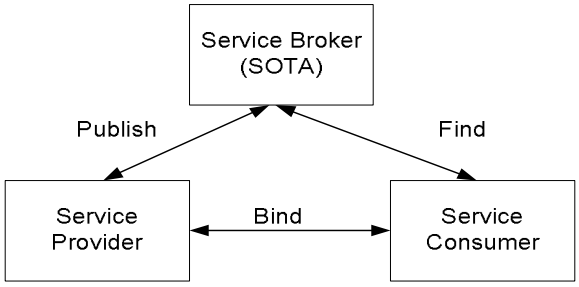

Figure 5. SOA diagram with SOTA as the Service Broker

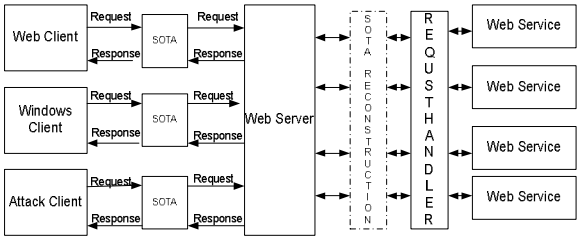

Figure 6. Mini-SOTA

3.3 Alternative SOTA

Figure 3 shows the main SOTA used in this paper as a Monolithic structure. An alternative approach is shown in Figure 6 using mini SOTA. Mini SOTA can be created with its own WSDL. WSDL is used to broadcast to the public what services are available and can be requested. Mini SOTA can allow service providers the option of splitting their services into different categories. This would make for a more efficient and effective use of service resources. Services that are categorized make it simple and easy for service providers to find a particular service. Searching for a particular service in a mini WSDL would be relatively easy instead of one monolithic WSDL.

4. Performance Evaluation

4.1. Simulation Setup

4.1.1 Assumptions

The following assumptions are made about SOTA:

- It is assumed that an attacker may control any number of client machines that are widely distributed across the Internet.
- It is further assumed that 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.

```
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 7. 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 8. Pseudo Code to extract, store and display username identification

- That the service provider of Web Service has limited resources.
- SOAP headers are being used by the Client.

4.1.2 Header Format for SOTM

After SOAP message has been re-formatted, the SOAP header information regarding identification is extracted from the message, Figure 7. If there is no SOAP header, then a header is created with the client identification attached. Once the header has been created or updated the message is forwarded to the Web-Server.

4.1.3 ID reconstruction

SOTA reconstruction is to handle the reconstruction of the path back to the true source of the message. Also, SOTA reconstruction is given instruction by the service provider at the time of the attack or at the end to start the reconstruction process. The information that is extracted, should lead to the source of the message attack and initiation of protection measures, such as telling the firewall to block message from that particular client for example. Figure 8 displays the pseudo code that extracts the SOAP header information and stores the information in a table array.

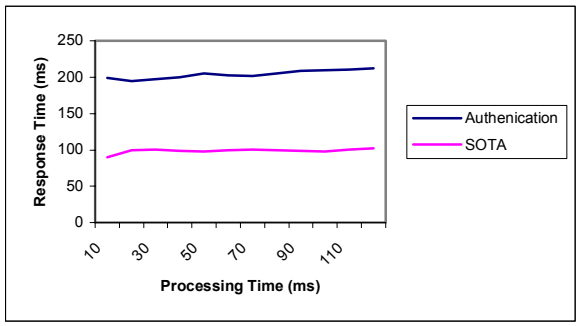

Figure 9. Results of SOAP Authentication and SOTA

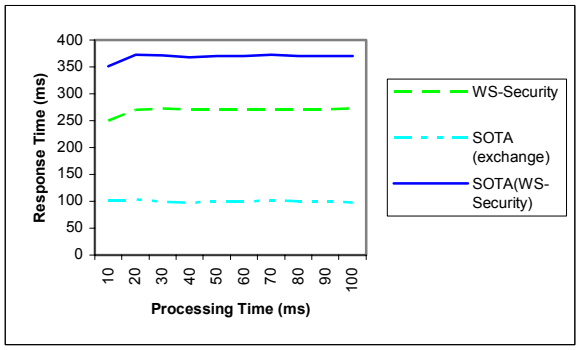

Figure 10. Results of WS-Security, SOTA(exchange) and SOTA(WS-Security)

4.2 Evaluation

Experiments were carried out to evaluate the performance of the SOTA system. A Dell Dimesion DM501 Intel Pentium single-core CPU, 3.0 GHz, 2 GB of RAM and 2 300GB SATA hard-drives was used for our experiments. All our programs were implemented with .NET Web Services with the use of VB and ASP high-level language. In our first experiment we developed a basic SOAP Web Service using the ASP.net Web Services. The program contained a basic header for authentication purposes.

To simulate SOTA, the program extracts from the header the name id and replaces it. It is assumed that a one-way transmission delay between client and Web Server is 10ms. This delay is simulated by the program going into a wait mode for 10ms before a message is sent to the web-server again. It is further assumed that SOTA is on the same tier as the Web-Server instead of being in front of the Web-Server to protect it. The measurements in this experiment represent the processing time over the response time of each message that was sent to the Web-Server and to SOTA. Figure 9 shows the results of the experiment. The results show that it took more then 2 secs of processing time. This shows that SOTA is far more efficient and effective then the authentication procedure. SOTA in this experiment only had to replace the id within the header while authentication had to process and verify the identification. One of the reasons for SOTA being able to respond quicker to the messages than SOAP authentication is that the service provider only has to access SOTA if they want to know the true source of where the message came from. Having the extra response time to process messages will cut down the load on computer resources during a DDoS and XDos.

In our second experiment we run a WS-Security interaction application against Amazon Elastic Compute Cloud (Amazon EC2) [33] SOAP service. The WS-Security application contained a signed certificate for authentication purposes. SOTA in this experiment was to exchange the username id name for the authentication name before it was sent to the Amazon SOAP service. This was to ensure that no errors were to occur during the experiment. The results are based on how long the applications had taken to process, and how quickly the response came back from Amazon. SOTA can be used in conjunction with WS-Security in order to re-place the name id for the real-source id. The results show in figure 10 that when SOTA is introduced to WS-Security an increase in the response time is around thirty percent. The increase in response time means that during a DDoS attack more processing time is required to handle the extra burden. The advantage with this increase is that the true identification of the source of the attack will most likely be identified. WS-Security compared to SOTA [SOTA (exchange)] is over double the response time. WS-Security had to build a security token before the message was sent to the Amazon Web Server. On receipt of the token, Amazon can test the authentications of the message. However, SOTA only has to place or exchange the identification information.

5. Conclusion and Future Work

This paper proposes a framework for identifying the real source of XDoS and 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 at being able to traceback, to the source the real identity of XDoS and DDoS attack on Web Services. Once an attack has been discovered and the identity known then counter measures can be initiated.

6. Acknowledgements

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

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

[5] 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

[6] 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

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

[8] 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

[9] 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.

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

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

[12] 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

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

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

[15] Salz, R., (2005) "*Essential XML Web Services Security Practices*", http://www.idealliance.org/papers/dx_xml03/papers/05-2/05-04-02.pdf

[16] Oasisopen.com, (2008), 'Security Assertions Markup Language(SAML)', http://www.oasisopen.org/committees/tc_home.php?wg_abbrev=security , (2008)

[17] Prolexic Technologies, 'Prolexic Technology Report,(2007), http://www.prolexic.com/zr/zombie_july_2007.pdf

[18] Kaler, C and Nadalin, A., (2005), "Web Services Security Policy Language (WS-SecurityPolicy) 1.1" http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf 2005.

[19] 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.

[20] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I and Weerawarana, S., (2003). *Business Process Execution Language for Web Services Version 1.1*, http://www.ibm.com/developerworks/library/specification/ws-bpel/ , May 2003.

[21] Aljifri, M., (2006), '*IP Traceback: A NewDenial-of-Service Deterrent?*' Published By The Ieee Computer Society 1540-7993/03 2003

[22] Stone, R, (2000) "CenterTrack: An IP Overlay Network for Tracking DoS Floods," *Proc. 9th Usenix Security Symp.*,Usenix Assoc., 2000, pp. 199–212

[23] Burch, H., and Cheswick, B., "Tracing Anonymous Packets to Their Approximate Source," *Proc. 14th Conf.Systems Administration*, Usenix Assoc., 2000, pp.313–322.

[24] Bellovin, S., Leech, M., and Taylor, T., (2003), 'ICMP Traceback Messages,' Internet Draft, Internet Eng. Task Force, 2003; work in progress.

[25] Mankin, A., Massey, D., Wu, C.L., Wu S.F and Zhang, L., (2001), "On Design and Evaluation of 'Intention-Driven' ICMP Traceback," *Proc. IEEE Int'l Conf. Computer Comm. and Networks*, IEEE CS Press, 2001. pp. 159–165.

[26] Snoeren, A.C., et al., (2002), "Single-Packet IP Traceback," *IEEE/ACM Trans. Networking*, vol. 10, no. 6, 2002, pp. 721–734.

[27] Baba, T., and Matsuda, S., (2002). "Tracing Network Attacks to Their Sources," *IEEE Internet Computing*, vol. 6, no. 3, 2002, pp. 20–26.

[28] Adler, M, (2002),'Tradeoffs in Probabilistic Packet Marking for IP Traceback,' *Proc. 34th ACM Symp. Theory of Computing*, ACM Press, 2002, pp. 407–418.

[29] Peng,T., Leckie, C., and Kotagiri, R., (2002), 'Adjusted Probabilistic Packet Marking for IP Traceback', Networking 2002.

[30] 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

[31] 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

[32] Aiello, M and Dustdar, S, (2006), 'Service-Oriented Computing: Service Foundations', Dagstuhl Seminar Proceedings 05462, http://drops.dagstuhl.de/opus/volltexte/2006/528

[33] Amazon.com, (2008), 'Amazon Elastic Compute Cloud', http://aws.amazon.com/ec2