A DEFENSE MECHANISM FOR WEB SERVICES AGAINST DDOS ATTACKS

Igni Sabasti Prabu S.[1] **Jawahar Senthil Kumar V.**[2]

[1]Research Scholar, Sathyabama University, Chennai.
[2]Asst. Prof, Dept of ECE, Anna University, Chennai.
Email: [1]igni.prabu@gmail.com, [2]veerajawahar@annauniv.edu

ABSTRACT

As Web Services become more and more popular, not only within closed intranets but also for inter-enterprise communications, security is becoming crucial for operating Web Services. One of the worse attacks over web services is distributed denial of service attack. DDoS attack is an attack in which a multitude of compromised systems attack a single target, thereby causing denial of service for users of the targeted system. The flood of incoming messages to the target system essentially forces it to shut down, there by denying service to the system to legitimate users. The proposed paper introduces a robust mechanism to protect both web services and web servers from DDoS attack. This system is designed in such a way that it is difficult for an attacker to disable the service host also overload the server. This model uses both WSDL schema validation and server rate limit (SRL) mechanism to detect and protect services and servers from the attack.

Key words DDoS, WSDL, Server Rate limit

I. INTRODUCTION

A *denial of service* (DoS) attack is defined as an explicit attempt by a malicious user to consume the resources of a server or a network, thereby preventing legitimate users from availing the services provided by the system. There are two major types of DDoS attacks [11]. The attacks of the first types attempt to consume the resources of the victim host. Generally the victim is a web server or proxy connected to the Internet. When the traffic load becomes very high, the victim host starts dropping packets both from the legitimate users and attack sources. The victim also sends messages to all the sources to reduce their sending rates. The legitimate sources slowly down their rates while the attack sources still maintain or increase their sending rates. Eventually, the victim host's resources, such as CPU cycles and memory space get exhausted and the victim is unable to service its legitimate clients. The attacks of the second type target network bandwidth. If the malicious traffics in the network are able to dominate the communication links, then traffics from the legitimate sources are affected. The effects of bandwidth DDoS attacks are usually more severe than the resource consumption attacks.

II. STYLES OF WEB SERVICES

Web services are a set of tools that can be used in a number of ways. The three most common styles of use are RPC, SOA and REST.

Remote Procedure Call (RPC):

RPC Web services present a distributed function (or method) call interface that is familiar to many developers. Typically, the basic unit of RPC Web services is the WSDL operation.

Service Oriented Architecture (SOA):

Web services can also be used to implement architecture according to service-oriented architecture (SOA) concepts, where the basic unit of communication is a message, rather than an operation. This is often referred to as "message-oriented" services. SOA Web services are supported by most major software vendors and industry analysts. Unlike RPC Web services, loose coupling is more likely, because the focus is on the "contract" that WSDL provides, rather than the underlying implementation details.

Representational State Transfer (REST):

REST attempts to describe architectures that use HTTP or similar protocols by constraining the interface to a set of well-known, standard operations (like GET, POST, PUT, DELETE for HTTP). Here, the focus is on interacting with stateless resources, rather than messages or operations. Clean URLs are tightly associated with the REST concept.

III. PROTECTING WEB SERVICES

A host [12] or a service in a network is prevented by a firewall. The tasks of the firewall are to protect the services from attacks and to prevent access to services, which shall not be reachable from the internet. The most widespread firewall concept is packet filtering. Such firewalls are suitable for protecting against DoS attacks exploiting the TCP or IP protocol, like Ping of Death or the TCP/SYN Flood. It is also capable to filter accesses to services using the target IP address and the target TCP port.

Packet filters and HTTP Application Level Gateways only check the TCP, IP and HTTP protocol header, but not the REST messages. A Web Service defines a valid REST message using a Web Service interface description language. Processing of REST messages is time and memory consuming for the Web Service server, so every non-valid message should be rejected by a Web Service firewall. This can be done by validating the REST message in an external application level gateway.

A very simple countermeasure against large valid messages is limiting the REST incoming message. This can even be done by a simple firewall without checking the REST messages itself. On the other hand, this is not very sensible. The amount of memory needed while processing an XML document is usually much larger than the document itself. In order to avoid attacks, the size limit should be low. Unfortunately, this could exclude many valid documents.

A much more sophisticated solution is to restrict the length of single XML elements and also the number of elements inside the REST message. These restrictions can be enforced by validating the REST message against a specially modified XML Schema derived from the Web Service interface description. In the same way, validating the REST message to an XML Schema containing only the allowed operations solves the WSDL Scanning.

IV. WORK FLOW ARCHITECTURE

IPTABLES

The iptables net filter implementation in Linux 2.4 module iptables is a user space application program that allows a system administrator to configure the tables provided by the Linux kernel firewall (implemented as different Net filter modules) and the chains and rules it stores. Iptables applies to IPv4, *ip6tables* to IPv6, *arptables* to ARP, and ip*tables* to Ethernet frames. The “iptables” providing excellent logging mechanism based on different categories it can be programmed to log based on the TCP flags, packet size, based on time, etc.,

4.1 Algorithm

- The checkpoint server analyzes the request based on WSDL schema.
- XML elements are parsed and the individual flags set.
- A time based Server Rate Limit (SRL) control is encapsulated.
- The flag value in the database is checked and inserted into the ip block table.
- Protection scripts fetch the ip address and kept in the linux firewall scheduler.
- Scheduler run the specialized scripts to update the ip chains of the web server.

4.2 XSD Schema Validation

The XML Schema Definition (XSD) is the standard schema language for all XML documents and data which enables us to define the structure and type of XML documents. An XML schema defines the elements, attributes and data types that conform to the World Wide Web Consortium (W3C).

4.3 Design of the Checkpoint Service Engine

The Checkpoint engine has its own queuing model to process the incoming request. And the server has a dynamic buffer allocation model using the MemCache technique which acts as a service. The system has the statistical sample of the incoming request rate (which is taken from the actual server and calculate request per minute) based on the request the server decide where to keep the request in FIFO buffer. Initially the system has two buffers Buffer 1 (B1) and Buffer 2 (B2).

For every request the system calls the MemCache (MC) service to check for the interval and the existence of the ip address. If it’s a fresh request then the MC update the cache scheme with ip address and the Packet Arrival Rate (PAT) and keep the request in the B1 buffer. If any further incoming request from the same server within the caching period the system drop the request. Otherwise it updates the caching service and allows the request to get serviced. If the caching time is expired the system removed the cached ip address and the time interval automatically. The MemCache runs as a service on port 8080.

4.4 Security Architecture

1. User sends request to the server (XML format).
2. The request is forwarded to the Checkpoint proxy server.
3. The Checkpoint Proxy Validate the incoming request and register the ip-address and insert the incoming request time into the database.
4. Return the URI to the actual Web server, in case of trusted request.
5. Proxy forwards the URI to the Web Service.
6. Web Server sends requests to the Database Server.
7. Fetching the records from the database.
8. Response from the database to the actual web server.
9. Proxy forwards the response URI to the user agent.
10. The checkpoint proxy redirects the response to the user in the form XML.

As shown in Fig. 1. The user agent (UA) send requests to the web servers to access the services, Where the Checkpoint proxy running in sniffing mode which analyse the incoming request in application layer of the ISO-OSI model and validate the request against the WSDL-Schema. In-case found any malicious tag it block the system from accessing the web services.

4.5 Attack Scenario

Fig. 2 show attack scenario where

1. User sends request to the server (XML format).
2. The request is forwarded to the Checkpoint proxy server.
3. The Checkpoint Proxy Validate the incoming request and register the ip-address and request time into the database.
4. The ip-tables read the ip-address from the database frequently and push the ip-address into the net filter module of the linux kernel.

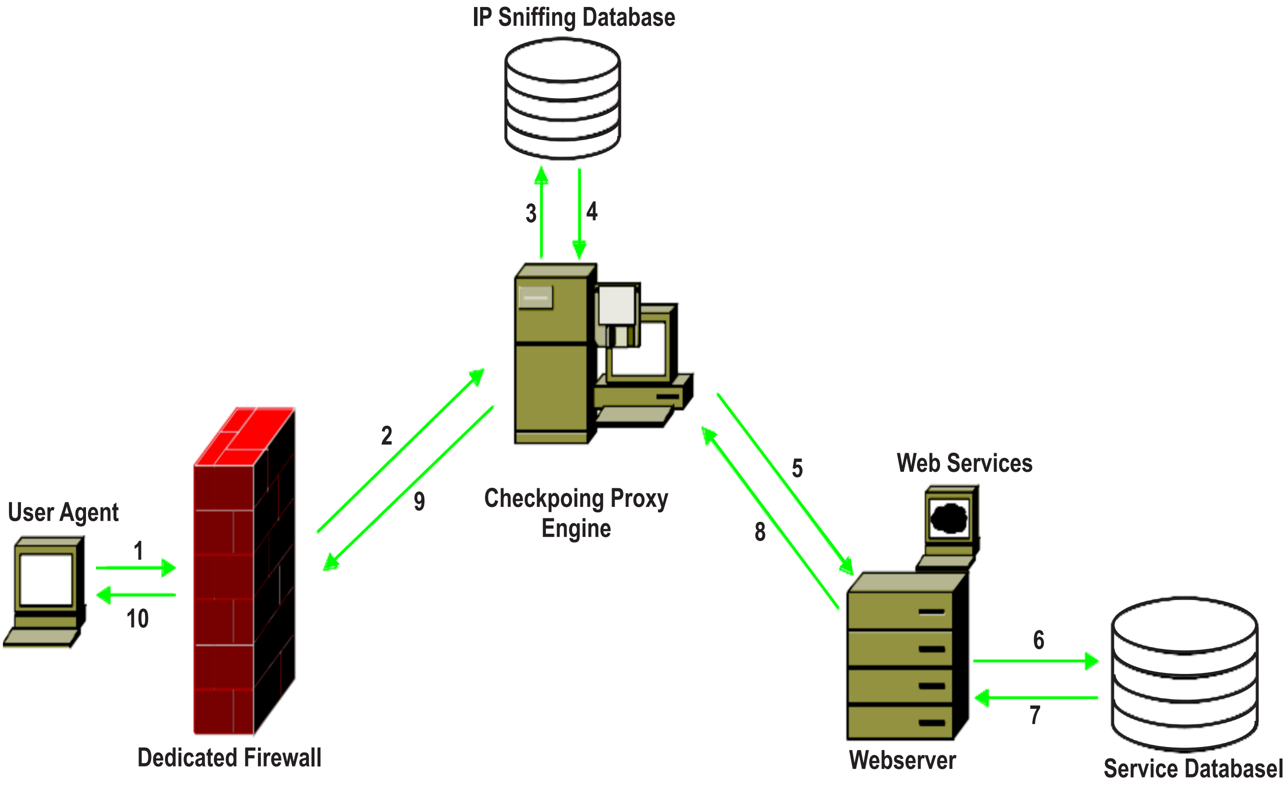


Fig. 1. Security Architecture

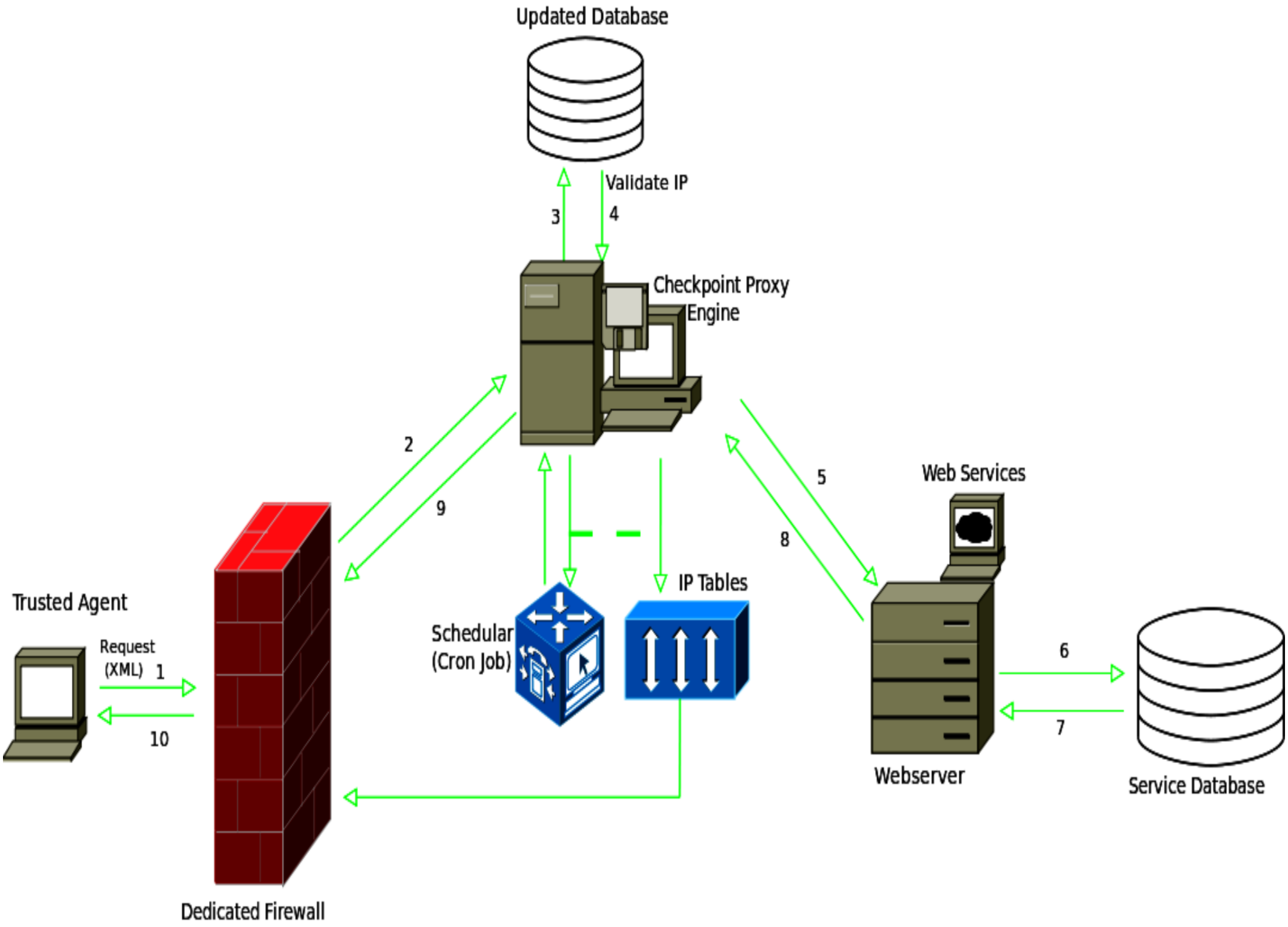
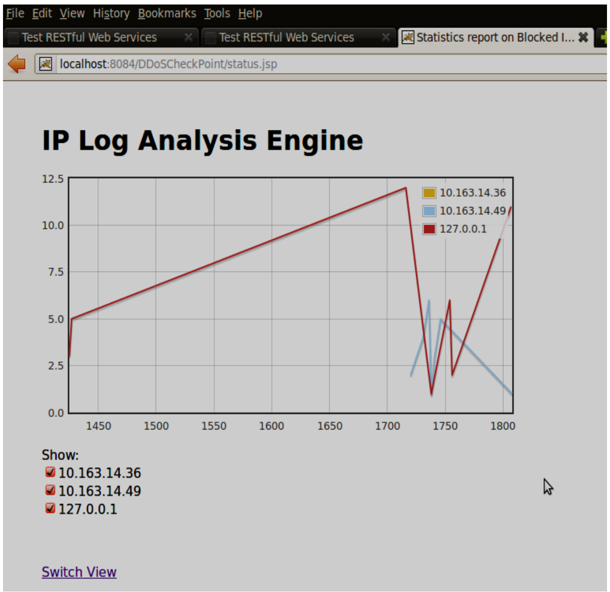


Fig. 2. Attack Architecture

5. The linux scheduler updates the net filter module per-minute push the ip-address into the perimeter level firewall.
6. The linux box blocks the ip-address of the malicious user.
7. Fetching the records from the database.
8. Response from the database to the actual web server.
9. Proxy forwards the response URI to the user agent.
10. The checkpoint proxy redirects the response to the user in the form XML.

V. EXPERIMENTAL RESULTS

The log analysis engine intimates the effective user about the abrupt change in the traffic is shown in Fig. 3. This provides clear view about different ip address


File Edit View History Bookmarks Tools Help
Test RESTful Web Services
Test RESTful Web Services
Statistics report on Blocked I...
localhost:8084/DDoSCheckPoint/status.jsp
12.5
10.0
7.5
5.0
2.5
0.0
1450
1500
1550
1600
1650
1700
1750
1800
10.163.14.36
10.163.14.49
127.0.0.1

IP Log Analysis Engine

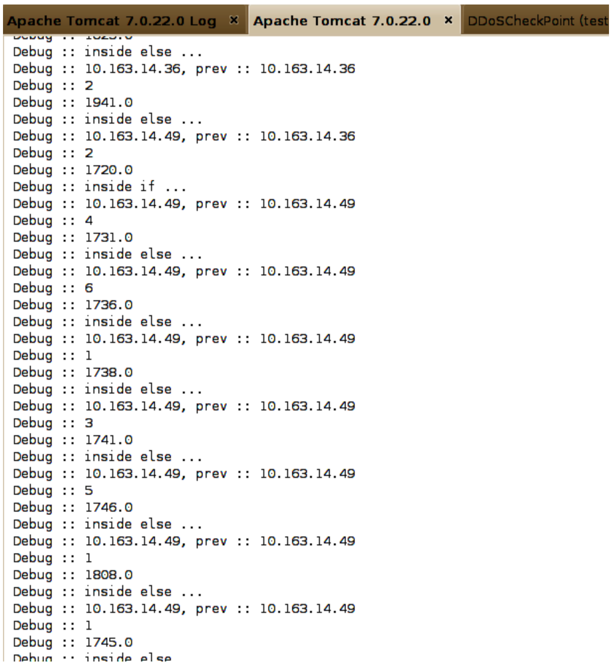
Show:
☑ 10.163.14.36
☑ 10.163.14.49
☑ 127.0.0.1

Switch View

Fig. 3 Log analysis engine

Fig. 3 shows there's no change in the log, so does point the normal status. In-case of abrupt change the log analysis engine fetches the data from the database periodically and draws the appropriate graph dynamically.

The server log allows the developers to know the status of the service engine. This is useful for server debugging and recovering process in-case of server compromised.



```
Debug :: inside else ...
Debug :: 10.163.14.36, prev :: 10.163.14.36
Debug :: 2
Debug :: 1941.0
Debug :: inside else ...
Debug :: 10.163.14.49, prev :: 10.163.14.36
Debug :: 2
Debug :: 1720.0
Debug :: inside if ...
Debug :: 10.163.14.49, prev :: 10.163.14.49
Debug :: 4
Debug :: 1731.0
Debug :: inside else ...
Debug :: 10.163.14.49, prev :: 10.163.14.49
Debug :: 6
Debug :: 1736.0
Debug :: inside else ...
Debug :: 10.163.14.49, prev :: 10.163.14.49
Debug :: 1
Debug :: 1738.0
Debug :: inside else ...
Debug :: 10.163.14.49, prev :: 10.163.14.49
Debug :: 3
Debug :: 1741.0
Debug :: inside else ...
Debug :: 10.163.14.49, prev :: 10.163.14.49
Debug :: 5
Debug :: 1746.0
Debug :: inside else ...
Debug :: 10.163.14.49, prev :: 10.163.14.49
Debug :: 1
Debug :: 1808.0
Debug :: inside else ...
Debug :: 10.163.14.49, prev :: 10.163.14.49
Debug :: 1
Debug :: 1745.0
Debug :: inside else
```

Fig. 4 Server log

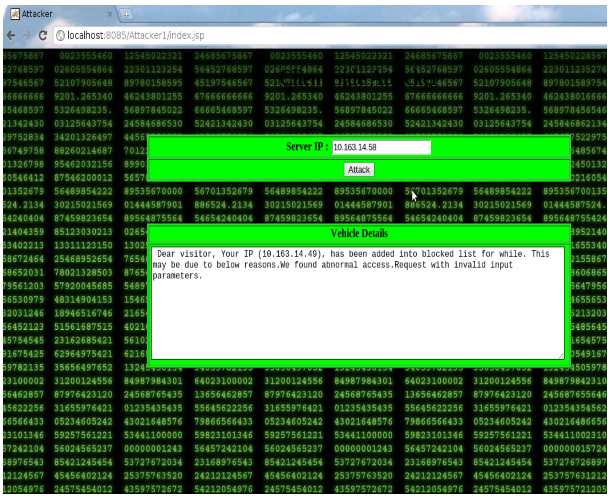
Fig. 4 The server log enables both the server administrator and the security administrator to know the status of the web service and the web server load. Initially the log shows the server start time with other security attributes.

The attacker log sends to the attacker

Fig. 5 shows the log send to the attacker that the ip has been blocked. The reasons for blocking the ip address may be due to abnormal access to the web server or due to invalid input parameters which are detected using XML schema validation.

VI. CONCLUSION

In this paper we have a proposed a method to prevent the web server and web services. This paper opens up new possibilities to defend the various Distributed Denial of Service Attacks. XML schema validation detects the malicious requests which target the web services running on any port of the server.



Server IP : 10.163.14.58

Vehicle Details

Dear visitor, Your IP (10.163.14.49), has been added into blocked list for while. This may be due to below reasons.We found abnormal access.Request with invalid input parameters.

Fig. 5. Blocked IP Address

VII. REFERENCES

[1] Jaydip Sen A Robust Mechanism For Defending Distributed Denial of Service Attacks on Web Servers: International Journal of Network Security & Its Applications (IJNSA), Vol.3, No.2, March 2011.

[2] Michael Walfish DDoS Defense by Offense, : ACM Transactions on Computer Systems, Vol.28, No. 1, Article 3, Publication date: March 2010.

[3] Mitigating Application-Level Denial of Service Attacks on Web Servers: A Client-Transparent Approach :ACM Transactions on the Web, Vol. 2, No. 3, Article 15, Publication date: July 2008.

[4] Kaziim Sarikaya, Duygu Sarikaya A Novel Client-Based Approach for Signing and Checking Web Forms by Using XML Against DoS Attacks, (ii) WAS2010, 8-10 November, 2010, Paris, France. Copyright 2010 ACM 978-1-4503-0421-4/10/11.

[5] http://www.ics.uci.edu/~fielding/pubs/dissertation/ top.html

[6] http://www.cisco.com/web/about/ac123/ac147/ archived_issues/ipj_9-4/ syn_flooding_attacks.html

[7] http://www.ajaxonomy.com/2008/xml/ web-services-part- 1-soap-vs-rest

[8] Ramanathan, A.: WesDes: A Tool for Distributed Denial of Service Attack Detection. Thesis at Texas A&M University, August 2002.

[9] Forristal, J.: Fireproofing against DoS Attacks. URL: http://www.networkcomputing.com/1225/1225f3.html, Network Computing.

[10] Sen, J.: A Novel Mechanism for Detection of Distributed Denial of Service Attacks. In Proceedings of the 1st International Conference on Computer Science and Information Technology (CCSIT 2011), pp. 247 – 257, January 2 – 4, 2011, Bangalore, India.

[11] Peng, T., Leckie, C., Ramamohanarao, K.: Survey of Network-Based Defense Mechanisms Countering the DoS and DDoS Problems. ACM Computing Surveys, Vol. 39, No. 1, April 2007.

[12] Nils Gruschka, Norbert Luttenberger Protecting Web Services from DoS Attacks by SOAP Message Validation.

[12] Leiwo, J., Aura, T., Nikandar, P.: Towards Network Denial of Service Resistant Protocols. In Proceedings of IFIP SEC 2000, Beijing, China, pp. 301 – 310, August 2000.

[13] Ferguson, P., Senie, D.: Network Ingress Filtering: Defending Denial of Service Attack which Employ IP Source Address Spoofing. RFC 2827, May 2000.

[14] Belovin, S., Leech, M., Taylor, T.: ICMP Traceback Messages. Internet draft, October 2001. URL: http://www.ietf.org/internet-drafts/draft-ietf-itrace-01.txt.

[15] Kiran, U., Tupakula, Varadharajan, V.: Analysis of Traceback Techniques. In Proceedings of the ACSW, January 2006.Cisco Systems Inc, "Characterizing and tracing packet floods using Cisco routers." URL: http://www.cisco.com/warp/public/707/22.html.

[16] Law, T.K.T., Lui, J.C.S., Yau, D.K.Y.: You Can Run, but You Can't Hide: An Effective Statistical Methodology to Trace Back DDoS Attackers. IEEE Transactions on Parallel and Distributed Systems, pp. 799-813, September 2005.

[17] Burch, H., Cheswick, B.: Tracing Anonymous Packets to Their Approximate Source. In Proceedings of the 14th Systems Administration Conference, USENIX LISA, December 2000, pp.319-327.

[21] Yau, D.K.Y., Lui, J.C.S., Liang, F.: Defending against Distributed Denial-of-Service Attacks with Max-Min Fair Sever-Centric Router Throttles. In Proceedings of the 10th International Workshop on Quality of Service, 2002.

[22] Cai, M., Chen, Y., Kwok, Y.K., Hwang, K.: A Scalable Set-Union Counting Approach to Pushing Back DDoS Attacks. USC GridSec Technical Report TR-2004-21, October 2004.

[25] Zou, C.C., Duffield, N., Towsley, D., Gong, W.: Adaptive Defense against Various Network Attacks. IEEE Journal on Selected Areas of Communication, High-Speed Network Security-Architecture, Algorithms, and Implementation, October 2006.

[26] Dwork, C., Naor, M.: Pricing via Processing or Combating Junk Mail. In Proceedings of the Crypto'92: 12th Annual International Cryptology Conference, LNCS Springer-Verlag, Vol 740, pp. 139 – 147, Santa Barbara, CA, August 1992.