Smurf Security Defense Mechanism with Split-protocol

Harold Ramcharan
Department of Computer and Information Sciences
Shaw University
Raleigh, NC, USA
hramcharan@shawu.edu

Bharat Rawal
Department of Computer and Information Sciences
Shaw University
Raleigh, NC, USA
brawal@shawu.edu

***Abstract*—Network intrusion has been a difficult problem to solve due to the rapid growth of the Internet in recent years. Securing computers from harmful attacks are becoming the unprecedented challenging issues for internet users. Every day the recognition of new attacks is becoming a harder problem to crack in the field of Computer Network Security. Currently, Denial of Service (DoS) attacks is affecting the large number of computers in the world on a daily basis. Detecting and preventing computers from DoS attacks is a major research topic for researchers throughout the world. The migratory nature and role changeover abilities of servers in Split-protocol avoid bottleneck on the server side. It also offers the unique ability to avoid server saturation and compromise from DoS attacks. The goal of this paper is to present the idea of Split-protocol as a protection technique against DoS attacks.**

Keywords-Split Protocol; Protocol splitting; DoS; Tribal Flood Network; Bare Machine Computing.

I. INTRODUCTION

Overloaded serves are always at a higher risk for security compromise. The Split-protocol [1] offers a mechanism for server change over without involving clients. For example, as shown in Figure 1, a client on the network sends a request through the Connection Server (CS). This request will then be forwarded to the Data Server (DS), which in turn sends the requested data to the client. The symmetrical structure of CS and DS allows changing roles dynamically.

Should DS1 server crash, DS2 server will take the IP of DS1 and all of its data will be relinquished to DS2. Whenever DS1 is overloaded (CPU is around 96%), DS1 will shutdown as DS* takes over (DS* is back up to DS1, such as DS2, DS3…). By toggling between DS1 and DS*, one can avoid saturation of the server. A detail mechanism is explained in Section II. This mechanism is similar to the mobile defense mechanism [24].

Protocol splitting enables TCP to be split into its connection and data phases, so that these phases are executed on different machines during a single HTTP request [1]. In the basic form of splitting, the state of the TCP connection to the original server is transferred to a Data Server after receiving the HTTP Get request with no client involvement. The Data Server then transfers the data to the client, and connection closing can be handled by either the original server or the Data Server. Many variations on basic TCP/HTTP splitting are possible and have been used to improve Web server performance by use of delegation [1], split mini-clusters [2], and split architectures [3]. The security and addressing issues that arise due to protocol splitting can be solved in a variety of ways. The simplest solution is to deploy the servers in the same subnet or in the same Local Area Network (LAN) if host-specific routes are supported. The latter is used in this paper for testing migration performance by splitting. More generally, splitting can be applied to protocols other than TCP/HTTP by identifying protocol phases that are amenable to splitting. In this paper, we adapt TCP/HTTP splitting to devise a novel technique for Web server migration. It enables an alternate Connection Server to

dynamically take over active TCP connections and pending HTTP requests from the original Connection Server upon receiving a special inter-server message from it. Migration based on splitting can be used to improve Web server reliability with only a small penalty in performance. Additional benefits of splitting such as Data Server anonymity and load sharing can also be achieved with this approach to migration. We first implement Web server migration using split bare PC Web servers [1] that run the server applications with no operating system or kernel support. We, then, conduct preliminary tests to evaluate performance with migration in a test LAN where the split bare PC servers are located on different subnets. Protocol splitting is especially convenient to implement on bare machine computing systems due to their intertwining of protocols and tasks. However, the migration technique based on splitting is general, and can be implemented using conventional servers that require an operating system or kernel to run [4].

The rest of the paper is organized as follows. Section II discusses related work. Section III describes the Web server migration, and its design and implementation. Section IV describes a Smurf attack. Section V discusses possible ways to address these attacks. Section VI presents the design and the implementation of the proposal. Section VII contains the conclusion.

II. RELATED WORK

When an increasing number of users (or processes) accessing a website is beyond the tolerable threshold, the performance of the web server decreases. This is due to higher CPU utilization rates, thereby resulting in a greater response time. Furthermore, as the response time increases, the ratio of the users accessing the site will also decrease. This higher CPU utilization can occur due to intruders launching deliberate attacks. These unwanted users use unnecessary data and techniques to occupy most of the server's bandwidth, degrading the server performance, thus, rendering the site useless. Kuppusamy and Malathi [6], implemented a particular technique to detect and prevent both individual Denial of Service (DoS) attacks [8], as well as Distributed Denial of Services (DDoS) attacks [6]. DDoS occurs when a multitude of distributed attack is launched against a single site or server, as opposed to a single user staging direct attacks. In response to mitigating the effects of spoofing IP source addresses, as is common in DoS attacks where packets lack a verifiable IP source address, the unicast reverse path forwarding (uRPF) [7] is a valuable tool for this purpose. It requires that a packet be forwarded only when the source addresses are valid and consistent with the IP routing table, or ensuring that the interface that the packet arrives on is matches the same used by the router to reach the source IP of the packet. If the interface does not match, then the packet will be dropped.

In Hop-Count Filtering (HCF) [8], each end-system maintains a mapping IP address aggregates and valid hop counts from the origin to the end system. Packets arriving at destination with significant variation in hop counts are considered unreliable and are either discarded or flagged. Li et al. [9] described SAVE, a mechanism for propagating only valid prefixes along the same paths that data packets will follow. By using the prefix and path information, routers can thus construct the appropriate filtering mechanism along the paths. Bremler-Barr and Levy proposed a Spoofing Prevention Method (SPM) [10], where packets are exchanged using an authentication key affiliated with the source and destination domains. Nowadays, there is an ever growing threat of intruders to launch attacks utilizing both-nets [11]. In this case, since the attacks are carried out through compromised intermediaries, often termed bots, it is difficult to discover the initiator of the attacks. However, current trends indicate that IP spoofing still persists [12] [13]. Man-in-the-Middle attacks (MitM), is a variant of TCP hijacking, as well as DNS poisoning [14] [15], and are carried out by the attacker masquerading as the host at the other end of the communication. IP spoofing attack is a hijacking technique in which an attacker masquerades as a trusted host to hide his identity [21].

III. MIGRATION WITH SPLIT-PROTOCOL

A. Overview

Split protocols require a minimum of two servers, i.e., a Connection Server and a Data Server. The CS establishes the connection via SYNs and ACKs. When the HTTP Get is received by the CS, it sends an ACK to the client, and uses an inter-server packet message referred to as a Delegate Message (DM). The DM1 is used to transfer the TCP state to the DS, which sends the data to the client. In bare PC servers, the TCP state and other attributes of a request are contained in an entry in the TCP table (known as a TCB entry). The CS also handles the TCP ACKs for the data and the connection closing via FINs and ACKs. Typically, the CS has information about the requested file (i.e., its name, size, and other attributes), and the DS has the actual file (the CS may or may not have a copy). When the DS gets DM1, it creates its own TCB entry and starts processing the request. When a DS sends data to the client, it uses the CS's IP address. After the CS receives the FIN-ACK, it sends another inter-server packet DM2 to DS. The receipt of DM2 closes the state of the request in the DS. More details of protocol splitting are given in a Split-protocol technique for Web Server Migration [5]. For Web server migration, inter server packet would be sent with a special massage, indicating that the CS is going to crash, and the TCB entry moved from one CS to another CS (called CS* for convenience), enabling the latter to take over the connection. Migrating server content in this manner and requiring that CS and CS* use the same IP address for two-way communication, poses a new challenge: now CS* must be able to send and receive packets with the IP of CS, which has a different prefix. Furthermore, the client must remain unaware that migration or protocol splitting has occurred. The main focus of this work is to address these issues and migrate (or transfer) a client connection to a new server, when the current connection server detects that it is going down or is being taken down. The means by which the server might detect its imminent failure is beyond the scope of this paper.

B. Design and Implementation of Role Change

Before the CS shuts down, it must send all of its pending requests to its alternate CS*. We assume that CS* is connected to the network, but that it will not process any normal requests (i.e., it is in stand-by mode). Also, CS and CS* are able to communicate with each other. Prior to the connection transfer, inter-server packets are being sent from CS to the DS according to the usual protocol splitting [6] when GET requests arrive. Under large load conditions, it is possible that CS could have many unprocessed requests in its TCP table. In addition to these pending requests, new requests may still continue to be sent by the client during the time between when CS shuts down and CS* takes over. These requests will be lost and will be processed later by CS* when the client retransmits them. Before CS shuts down, it also sends a final inter-server packet to CS* to confirm it is shutting down. Only minimal modifications had to be made to the current split server and inter-server packet format to implement the migration.

Alternatively, the DS can also assume the role of CS* (instead of using a separate CS*) if CS sends its pending requests to DS. If DS has some of its previous data transfer requests still to be processed, it will complete them before it begins to act as CS*. Protocol splitting is designed so that the same server can provide services as a CS and/or a DS; so, it is capable of assuming the role of CS* to implement the migration.

CSs in Split-protocol do not reserve resources for all requests it receives; this increases the capacity of servers in handling many folds of higher load than conventional servers. Also, the self-delegating mechanism in the splitting protocol allows the server to deny accepting any additional request to process, and changes his identity (IP) within a single TCP connection. Even if it changes its identity, it will still continue to serve that old request already in the queue (receiving system), until completion or reset. While original server (CS) is recovering, a new server (CS*) who has replaced it, will manage new requests. When it reaches a saturation point, it will also change its identity (IP address), and this time, the original server will handle all new incoming traffic. Toggling the same IP address between multiple servers will minimize the incoming load on Split-servers [5].

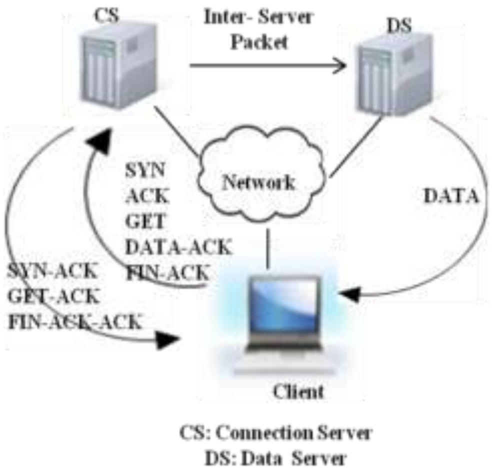

Figure 1. Split Architecture

IV. SMURF OR FRAGGLE

Smurf attacks can be considered one of the most overwhelming of the DoS attacks. In the Smurf [Internet Control Message Protocol (ICMP) Packet Magnification] attack [24], the attacker sends an ICMP echo request (ping) to a broadcast address with a spoofed source address. This source address is that of the victim's IP address. All the machines, when responding to the echo request, will flood the victim's system with their ICMP echo replies. As the flooding continues, this will ultimately results in the victim system crashing or freezing.

Smurf attack targets all available network bandwidth by consuming it with the intention to disrupt system's resources through bandwidth amplification. On a multi-access broadcast network, hundreds of systems could be responding to each packet sent, which could flood and render the victim's system useless even though using a much higher bandwidth type system [22]. The cousin of the Smurf's attack is the Fraggle attack, which uses the UDP echo packets in place of the ICMP echo packets in the same manner [35].

The recent rise in DoS attacks targeting high-profile web sites shows how overpowering these attacks are and how unprotected the Internet is under such attacks [28]. We present a survey of the up-to-date defense strategies against Denial of Service (DoS) attacks that gives hope in this area. We also present the weaknesses of the available methods pointing to the fact that no distinct adopted method has been in place. Also, future trends in DoS defense mechanism are discussed. The primary targets for these attacks are Web servers own by banks, online gaming websites, credit card payment gateways, domain name servers (DNS), E-commerce application tools, and Voice-over-IP (VoIP) services by prohibiting customer's access, or limiting access to resources such as bandwidth or degrading usage of these applications [21] [23].

The commonly employed attacks are:

A. Flooding

Flooding is the most basic method aimed to cripple a network by overwhelming it with large amounts of traffic directed to the victim. This utilizes all of the system resources [23] where the victim, in this case, can be a single PC or a high profile web server. The severity of such an attack depends more on the volume of traffic rather than the contents of the attack traffic.

B. Malware

Malware is malicious software used or programmed by attackers designed to overwhelm the system thereby allowing them unauthorized access. They will then have the capability to perform malicious operations. Known types of Malware includes: viruses, Trojan horse, adware and spyware, root kits, etc. The intended benefits for the perpetrator writing malware can range from financial gain to vengeance or for fun in seeing how fast and effective it can spread [24] [33]. These attacks exploit flaws in software vulnerabilities, such as in windows operating system or web server defects, cause these systems to reboot, crash or impede the system performance [25].

C. DoS attacks

Generally, DoS attacks can be categorized into two forms: (1) those that flood services affecting bandwidth, and (2) those that crash services by consuming resources [26]. Figure 2 describes different methods of DoS attacks. These DoS attacks become amplified when sent from unknown & unlimited sources termed Distributed Denial of service attack (DDoS) and usually occur in two phases, the recruitment phase and the actual attack [27] [28].

Method	Types
Protocol Based Attack	ICMP Flood, SYN Flood
Application Based Attack	HTTP Flood , SIP Flood
Distributed Reflector Attack	DNS Amplification Attack
Infrastructure Attack	

Figure 2. Methods of attack [28]

V. DEFENSE MECHANISM

From the standpoint of DoS, it is very difficult to completely remove the risk associated with such attacks. However, risk mitigation can be implemented through Avoid-Detect-Prevent cycle, as described here.

A. Avoid

Avoidance is an essential part of any defensive strategy even though many web sites choose to ignore it. Attacks can best be studied through collecting technical data such as network topology, Internet Service Provider (ISP) vendor agreements, insurance policy coverage, etc. However, from a risk management standpoint regarding DoS defense system, the need to identify and label critical services versus non-critical ones is important. The same apply for the corresponding vendors providing those services on the network. It is also important to have discussion with management, knowledgeable technical staff, service vendors, and law enforcement

B. Design network or system for survivability

This refers to the separation of critical services from non critical ones.

C. Monitoring

Prior to implementing monitoring procedures, special attention should be focused on target resources should an attack occur. Monitoring however, can be performed at two distinct levels; (a) at the network level, and (b) at the host level. Risks from DoS attacks can be reduced through the creation of effective incident response plans, establishing a sound partnership with service providers (vendor), and firewalls as intrusion prevention systems [26].

D. Detect

Modern networks can be very complex and diverse, therefore, an effective detection system is valuable to detect, prevent, and alert personnel of any DoS attacks in real time. Detecting an attack before becoming full scale can be vital to an organization's security posture. Modern Intrusion Detection Prevention Systems (IDPS) come equipped to combat these attacks and maintain state [24]. Detection systems should provide multiple detection mechanism, alerts, response mechanisms [25], and short detection time with low false positive rate [24]. These intrusion detection systems can take several forms such as anomaly detection, signature-based detection, and DoS attack detection, as discussed below [20].

E. Signature-based detection

This is simply searching network traffic and looking for a packet or series of bytes (signatures), which is considered malicious codes and comparing it to a set of attack signatures in order to detect the presence of an attack. A database of known signatures is usually developed by antivirus vendors for detecting known signatures [20]. This technique is also used by Snort (an IDPS), as it can perform real–time packet content searching and matching [19]. Snort and other IDPSs have one major weakness; they may take some time for a new exploit to become known. Later, after this new attack is known, a new signature can be developed and implemented. But, until then, well-defined signatures may go undetected [27].

F. Anomaly-based detection

This detection mechanism focuses on examining network traffic and comparing it with an established baseline [19], and is characterized by a set of pre-programmed thresholds [26]. This includes statistical approaches together with varying techniques such as those adapted from machine learning Models and Algorithms [17]. Neural Networks [31] and Bayesian Learning [32] can also be applied.

G. DoS-attack-specific detection

DoS attack traffic is instituted by the attacker as his objective is to direct maximum traffic to launch a powerful attack and may generate random patterns to make an attack signature undetected.

H. Prevent

Attack prevention measures aim to detect and prevent attacks before becoming full scale. Distributed packet filtering is possible through local routing information in order to prevent severe flooding attacks [24]; thus, a reaction and alert mechanism must be instituted to minimize the loss potential. This response mechanism should be effective in providing early detection automatically, dodging network overloading, and localizing the attack source with trace back techniques [18] [19], or mitigating the propensity of the attack [16] by denying unwanted packets.

I. Reaction

Reaction methods include effective incidence response plan, efficient backup systems, and filtering excessive traffic.

There are several mitigation techniques implemented for DoS and DDoS attacks. Avi Chesla [30] introduces an anomalous pattern for an HTTP flood protection. In this procedure, mitigation is controlled through a feedback mechanism that tunes a level of rate limiting factors. This is required for mitigating the attack effectively while allowing legitimate traffic to pass. A reliable trigger for an automated response system may be difficult to implement. Specht and Lee's [30] mitigation technique is based on similarities and patterns in different DDoS attacks. DDoS attack tools are normally designed to be friendly with different Operating

Systems (OS). Any OS system (such as UNIX, Linux, Solaris, or Windows) may have DDoS agents or handler code designed to work on it. Normally, a handler code is intended to support an OS that would be positioned on a server or terminal at either a corporate or ISP site. Most of the proposed mitigation mechanisms are also OS dependent. Split-protocol implementation on Bare Machine Computing (BMC) paradigm [4] does not use any kind of operating system. So, practically it is impossible to attack any BMC based system. On BMC, any DDoS agent or handler code designed for an OS cannot run. BMC codes are self-content. In addition, extra codes on existing applications or processes would not be allowed to run on the BMC system.

VI. DESIGN AND IMPLEMENTION

Split-protocol client server architecture design and implementation differ from traditional client server designs. As the traditional client server architecture is modified in this approach, we have designed and implemented a client server based on a bare PC, where there is no traditional OS or kernel running on the machine. This made our design simpler and easier to make modifications to conventional protocol implementations. Figure 3 shows a high level design structure of a client server architecture in a bare PC design. Each client and a server consist of a TCP state table (TCB), which consists of the state of each request. Each TCB entry is made unique by using a hash table with key values of IP address and a port number. The CS and DS TCB table entries are referred by IP3 and Port#. The Port# in each case is the port number of the request initiated by a client. Similarly, the TCB entry in the client is referenced by IP1 and Port#.

The TCB tables form the key system component in the client server designs. A given entry in this table maintains complete state and data information for a given request. This entry requires about 160 bytes of relevant information and another 160 bytes of trace information that can be used for traces, error, log, and miscellaneous control. This entry information is independent of its computer and can be easily migrated to another PC to run at a remote location. This approach is not the same as process migration [5], as there is no process information contained in the entry. The inter-server packet is based on this entry to be shipped to a DS when a GET message arrives from the client. Notice that the client uses IP1 and Port# to address the TCB entry. That means, when DS sends data or other packets, then it must use IP1 as its source address and its own MAC address in the packet. However, a client must be aware of IP1 and IP2 addresses to communicate to two servers for different purposes. The client knows IP1 through its own request and by resolving the server's domain name.

The client does not know IP2 address to communicate during the data transmission. We solved this problem by including the IP2 address in the HTTP header using a special field in the header format. In this design, a client could get data from any unknown DS and it can learn the Data Server's IP address from its first received data (i.e., header). This mechanism simplifies the design and implementation of Split-protocol client server architecture. This technique also allows the CS to distribute its load to DSs based on their CPU utilization without implementing a complex load balancing technique [1]. With implementing limited ACKs, the linear performance improvement continues up to 4 DSs [3]. This is also expected as CS poses no bottleneck for 4 DSs. For limited ACKs, the number of DSs connected to a single CS can be estimated to be 13 by extrapolating the CS CPU time and the number of DSs.

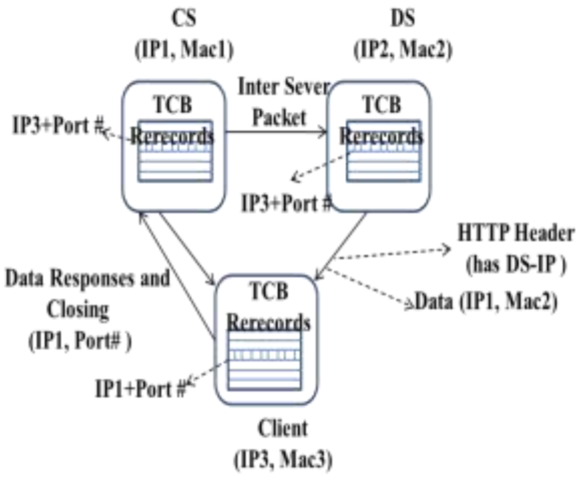

Figure 3. Design Structure

VII. CONCLUSION

Connection server in Split-protocol technique does not reserve any resource for all requests it receives, therefore it can handle many connection requests. In our empirical data, it suggests that CS only reserve 1% of CPU cycles compared to 95% for DS. Since there are many DSs in the system, they can handle very large loads without compromising services.

Also, the self-delegating mechanism in the split-protocol allows the server to deny accepting any additional request to process, and changes his identity within a single TCP connection. As shown in Figure 4, toggling the same IP address between multiple servers minimizes the incoming load on Split-servers. In multiple ways, both the Smurf attack and the Fraggle attack involves the attacker, the intermediary, and the victim.

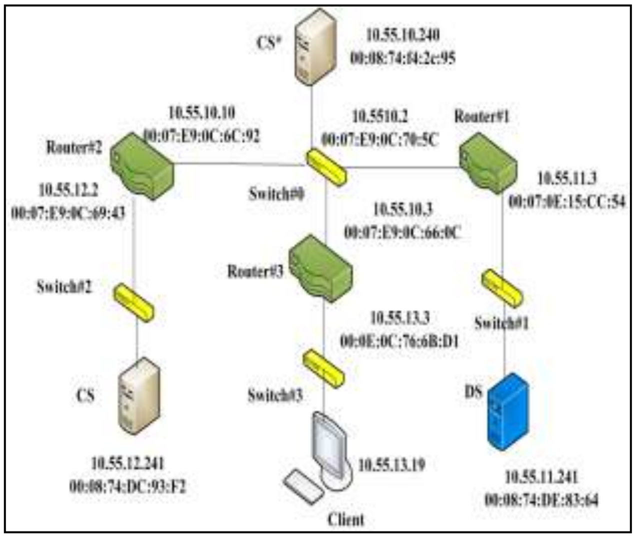

Figure 4. Role Change Over

Normally, both the intermediary and victim of this attack may suffer degraded network performance either on their internal network or on their connection to the Internet. Performance may be degraded to the point that the network cannot be used. Most of the time, the attacker identifies the underlying operating system from data structure of communication packets, which can further maximize the attack. Protocol-splitting, in our study, hides the underlying operating system thereby making it more difficult for Smurf attacker to circumvent. Furthermore, implementing protocol-splitting on BMC makes it harder to run a DDoS agent or handler code designed to work on operating systems. The anonymous nature of Data Server and migratory capability within single connections of Split-protocol architecture offers strong defensive mechanism against Smurf attacks.

REFERENCES

[1] B. Rawal, R. Karne, and A. L. Wijesinha. "Splitting HTTP Requests on Two Servers," The Third International Conference on Communication Systems and Networks: COMPSNETS 2011, January 2011, Bangalore, India.

[2] B. Rawal, R. Karne, and A. L. Wijesinha. " Mini Web Server Clusters for HTTP Request Split," 13th International Conference on High performance Computing and Communication, HPCC-2011, Banff, Canada, Sept 2-4, 2011.

[3] B. Rawal, R. Karne, and A. L. Wijesinha. "Split Protocol Client/Server Architecture," The 17th IEEE Symposium on Computers and Communications - ISCC 2012, July 1-4, 2012, Cappadocia, Turkey.

[4] L. He, R. K. Karne, and A. L. Wijesinha, "The Design and Performance of a Bare PC Web Server," International Journal of Computers and Their Applications, IJCA, vol. 15, no. 2, June 2008, pp. 100-112.

[5] B. Rawal, R. Karne, and A. L. Wijesinha, H. Ramcharan and Songjie Liang. "A Split-protocol Technique for Web Server Migration," The 2012 International workshop on Core Network Architecture and protocols for Internet (ICNA-2012) October 8-11, 2012, Las Vegas, Nevada, USA .

[6] K. Kuppusamy and S. Malathi, "An Effective Prevention of Attacks using GI Time Frequency Algorithm under DDoS", IJNSA journal, vol. 3, no. 6, November 2011, pp. 249-257.

[7] Team Cymru Inc "Bogon route server project", http: //www.cymru.com/BGP/bogon-rs.htm.[Retrieved:July, 2013].

[8] K. Park and H Lee, "On the Effectiveness of Probabilistic Packet Marking for IP Trackback under Denial of Service Attack," Network Systems Lab, Department of Computer Sciences, Purdue University, West Lafayette.

[9] C. Labovitz, D. McPherson and F. Jahanian, "Infrastructure attack detection and mitigation," ACMSIGCOMM 2005 conference, August 2005.

[10] J. Li, J. Mirkovic, M. Wang, P. Reiher and L. Zhang, "SAVE: Source Address Validity Enforcement protocol," In IEEE INFOCOM, vol.6, no.2, June 2002, pp. 81-95.

[11] S. Kandula, D. Katabi, M. Jacob and A. Berger, "Surviving Organized DDoS Attacks that Mimic Flash Crowds," NSDI'05 Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, 2005, vol.2, pp 287 – 300.

[12] D. Moore, G. Voelker and S. Savage "Inferring Internet Denial-of-Service activity," In proceedings of 10th Usenix Security Symposium, August 2001, pp.9-22.

[13] R. Pang, V. Yegneswaran, P. Barford, V. Paxson and L. Peterson, "Characteristics of internet background radiation," In Proceedings of ACM Internet Measurement Conference, October 2004.

[14] M. Dalal, "Improving TCP's robustness to blind in-window attacks," Internet- Draft, May 2005, work in progress.

[15] R. Beverly and S. Bauer. "The Spoofer Project: Inferring the extent of Internet source address filtering on the internet," In Proceedings of Usenix Steps to Reducing Unwanted Traffic on the Internet Workshop SRUTI'05, 2005, pp.53-59.

[16] P. Reiher, J. Mirkovic and G. Prier," Attacking DDoS at the source," In Proceedings of the IEEE International Conference on Network Protocols10, Paris, France, November 2002.

[17] T. Shon, Y. Kim, C. Lee, and J. Moon, "A machine learning frame work for network anomaly detection using SVM and GA," IEEE Workshop and Information Assurance and Security US Military Academy West Point NY, 2005.

[18] H. Burch and B. Cheswick, "Tracing anonymous packets to their approximate source," In Proceedings of the USENIX Large Installation Systems Administration Conference, New Orleans, USA, December 2000, pp. 319–327.

[19] M. Roesch, "Snort - lightweight intrusion detection for networks" http://www.snort.org [retrieved: July, 2013].

[20] Y. Xu and R. Guerin, "On the robustness of router-based denial-of-service (dos) defense systems," SIGCOMM Comput. Commun. Rev., vol. 35, no. 3, pp. 47–60, 2005.

[21] K. Kuppusamy and S. Malathi, "Prevention of Attacks under DDoS Using Target Customer Behavior "IJCSI International Journal of Computer Science Issues, vol. 9, Issue 5, no. 2, September 2012.

[22] S. Ratnaparkhi and A. Bhangee, "Protecting Against Distributed Denial of Service Attacks and its Classification: A Network Security Issue," IJCSI International Journal of Computer Science Issues, vol. 3, issue 1, Jan 2013.

[23] J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms," ACM SIGCOMM Computer Communications Review, volume 34, no. 2, April 2004, pp. 39-53

[24] P Tao, C Leckie and K Ramamohanarao. "Survey of Network-based Defense Mechanisms Countering the DoS and DDoS Problems," ACM Computing Surveys (CSUR), vol. 39, issue 1, article no. 3, August 2007.

[25] D. Slee, "Common Denial of Service Attacks," Jul 10, 2007. http://www.infosecwriters.com/texts.php?op=display&id=589

[Retrieved: July, 2013]

[26] F. Gong, “Detection Techniques: Part III Denial of Service Detection,” McAfee Network Security Technologies Group Jan 03.

[27] H. Alefiya, J. Heidemann, and C. Papadopoulos, "A framework for classifying denial of service attacks," 2003 conference on Applications, technologies, architectures, and protocols for Computer Communications. ACM, 2003.

[29] P. Jain, J Jain and Z Gupta “Mitigation of Denial of Service (DoS) Attack,” International Journal of Computational Engineering & Management IJCEM 11 (2011).

[30] A Chesla, "Generated anomaly pattern for HTTP flood protection." U.S. Patent no. 7,617,170. 10 Nov. 2009.

[31] S. M. Specht and R. B. Lee. "Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures." In ISCA PDCS, pp. 543-550. 2004.

[32] A. Chonka, S. Jaipal and Z. Wanlei, "Chaos theory based detection against network mimicking DDoS attacks." Communications Letters, IEEE 13.9 (2009): pp 717-719.

[33] N Abouzakhar, A Gani, G Manson, M Abuitbel and D King, "Bayesian learning networks approach to cybercrime detection." proceedings of the 2003 Postgraduate Networking Conference (PGNET 2003), Liverpool, United Kingdom. 2003.

[34] S. Kumar, “Smurf-based distributed denial of service (ddos) attack amplification in internet,” In Internet Monitoring and Protection, ICIMP 2007. IEEE Second International Conference July 2007, San Jose, California.

[35] http://www.javvin.com/networksecurity/SmurfAttack.html. [Retrieved: July, 2013].