AUGMENTED SPLIT–PROTOCOL; AN ULTIMATE DDoS DEFENDER

Bharat Rawal¹, HaroldRamcharan² and Anthony Tsetse³

¹&²Department of Computer and Information Sciences
Shaw University
Raleigh, NC, USA
³Department of Computer and Information Sciences
State University of New York
Fredonia, NY, USA

ABSTRACT

Distributed Denials of Service (DDoS) attacks have become the daunting problem for businesses, state administrator and computer system users. Prevention and detection of a DDoS attack is a major research topic for researchers throughout the world. As new remedies are developed to prevent or mitigate DDoS attacks, invaders are continually evolving new methods to circumvent these new procedures. In this paper, we describe various DDoS attack mechanisms, categories, scope of DDoS attacks and their existing countermeasures. In response, we propose to introduce DDoS resistant Augmented Split-protocol (ASp). The migratory nature and role changeover ability of servers in Split-protocol architecture will avoid bottleneck at the server side. It also offers the unique ability to avoid server saturation and compromise from DDoS attacks. The goal of this paper is to present the concept and performance of (ASp) as a defensive tool against DDoS attacks.

KEYWORDS

Split Protocol; Protocol splitting; DDoS; Tribal Flood Network; Bare Machine Computing.

1. INTRODUCTION

In a Denial of Service (DoS) attack, an intruder penetrates and depletes a computer system, refuting genuine users from using network services, such as a computer system, web server, or website [1]. While, a Distributed Denial of Service (DDoS) attack is a synchronized, multiple DoS attack that are launched through many negotiated machines. The targeted for the attack are those of the “primary victim,” while all the cooperated systems participating in the attack are referred to as the “secondary victims.” By adding many secondary victims in a DDoS attack, it allows the attacker the extravagance to launch a larger and more upsetting attack while remaining concealed. This happens since the direct source of attacks is launched from the secondary victims systems, thereby masking the true identity of the real invader.

These DDoS attacks frequently affect large network systems by disrupting or shutting down their services, and diminishing service performance while negatively impacting returns. The Split-protocol [2] offers mechanisms to hide the actual network services from the real world and role change over without involving client. For example, as shown in Figure 1a, a client on the network sends a request through the Connection Server (CS). This request will then be forwarded to the
Data Server (DS) through Resource Allocator (RA), which in turn sends the requested data to the client. The symmetrical structure of CS, RA and DS allows changing their roles dynamically. In case of DS1 server crash, DS2 server will take the IP of DS1, relinquishing all its data to DS2. Whenever DS1 is overloaded (CPU is around 96%), DS1 will shut down 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 [38]. Protocol splitting enables TCP to be split into its constituent connection and data phases, allowing for these phases to be executed on different machines during a single HTTP request [2]. Figure 1b shows the protocol transaction for migratory (M) Split-protocol. In its 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, all without client involvement.

**Figure 1a. Split Architecture**

After the DS receives the TCP connection, it then transfers the data to the client, and allows for the connection termination to be handled by either the original CS or the DS. 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 [4], and split architectures [5]. 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 approach for DDoS defense.

**Figure 1b. Augmented Split-protocol Transaction**
2. RELATED WORK

The Global Defense Infrastructure (GDI) proposed by K. Wan and R. Chang in [6] and [7] describe an approach similar to distributed management architecture in securing against DDoS attacks. Alert exchanges make all the infrastructure's members aware of their findings. The Cooperative Intrusion Traceback and Response (CITRA) framework [8, 9] is also comparable to Koutepas, G., Stamatelopoulos, F., & Maglaris, B’s Distributed management architecture[1], and uses the concept of administrative domain communities organized as neighbourhoods which maps out existing DDoS defense strategies mentioned in their literature. Their current DDoS defense mechanisms include “Detection, Response, and Tolerance & Mitigation”[36]. Attack detection aims to detect the presence of an ongoing attack followed by separating malicious traffic from legitimate for eviction. Typical detection methodology stem from signature based, anomaly based, hybrid, and third party attacks. SNORT IDS [10] and Bro [11] are the two most popular used open source signature based detection approaches. A known disadvantage in signature based techniques is that they are only capable of providing protection against known attacks. However, the threat landscape is continuously changing as new attacks are being developed daily, allowing them to go unnoticed.

The anomaly based detection method relies on base lining for network behaviour with valid traffic patterns and identifies anomalies whenever they deviate from the predefined or accepted model of behavior. Most of the commonly used DoS detection systems employed are anomaly based [19],[20]. In [12], Gil and Poletto proposed a method called MULTOPS for detecting DoS by examining the packet rates in both the up and down links. According to MULTOPS, under normal operation, packet rates between two hosts are considered proportional. Any steep variant or spiked disproportion in traffic to and from a host or subnet is a possibility of a DoS attack in progress. Blazek et al. Though the majority of DoS detection systems [20] use volume based metrics to identify DDoS attacks; they have been successful in defending against flooding attacks, however low rate flooding attacks usually go undetected as they do not appear to inflict significant disruptions in traffic volume, but on account of the large number of false positives and false negatives, significant damage can be inflicted when attack is carried at slow continuous rate. One method worth mentioning here is the entropy based DDoS detection [21]-[22] which boasts its effectiveness in countering diluted low rate degrading flooding attacks.

Higher CPU utilization rates can occur when intruders launching deliberate attacks on servers, or a higher than the allowable number (threshold) of users simultaneously. These unauthorized users overwhelms the server occupying most of its bandwidth, rendering it useless. Kuppusamy and Malathi [24] implemented a particular technique to detect and prevent (DoS) attacks [25], as well as (DDoS) attacks [24]. DDoS occurs when a multitude of coordinated and distributed attack is launched against a single target, such as a website or server. Spoofing is commonly associated with Dos and DDoS attacks, however, in response to mitigating the effects of spoofing IP source addresses where packets lack a verifiable IP source address, the unicast reverse path forwarding (uRPF) [26] is a valuable tool for this purpose. Bremler-Barr and Levy proposed a Spoofing Prevention Method (SPM) [28], where packets are exchanged using an authentication key
3. COMMON DOS/DDOS ATTACKS AND TECHNIQUES USED

In recent times, high profile business entities have been at the receiving end of DoS/DDoS attacks. The most common applications targeted are gateways, webservers, electronic commerce applications, DNS servers and Voice-over IP servers. The success of these attacks gives credence to how vulnerable and unprotected the internet has become. Considering the economic impact of network downtime on businesses, it becomes imperative that businesses invest a lot money and resources in protecting their IT infrastructure [34][37]. Some of the attacks employed are discussed below

3.1. Smurf and Fraggle

Smurf attacks have gained considerable eminence as a means of performing DDoS/DoS attacks. This approach of performing DoS attacks is based on the use of ICMP packets sent to broadcast network addresses by the attacker [42]. Fraggle attacks are similar to smurf attacks in their operation mechanism. In fraggle attacks however, UDP echo packets are sent instead of ICMP echo packets. In some variants of fraggle attacks, the UDP packet is sent to the intermediary’s port (chargen, port 19 in Unix systems) that supports character generation with the return address spoofed to the victim’s echo service (echo, port 7 in Unix systems) thereby amplifying the requests infinitely [40].

3.2. Flooding

In flooding, the attacker sends large amounts of packets to its victim with intent of consuming up all the victim’s available resources to a point that the victim can no longer process any requests from legitimate clients [23]. It is worth mention that in flooding attacks the volume of the traffic is what matters and not the actual contents of the traffic. Some of the common flooding techniques used are TCP SYN, UDP, SIP and HTTP GET/POST flooding.

3.3. Malware

Malware is malicious software that have been programmed overwhelm a system allowing attackers to gain unauthorized access, and in most cases escalating privileges of the attacker. Once an attacker is able to escalate his privileges on a system, the opportunity for launching an attack is limitless [43]. Malwares normally take advantage of vulnerabilities in Operating systems and application software and can be in the form of Trojan horses, rootkits, viruses, worms etc. The motivation for programming malware may be financial, for fun or to deliberately halt a system [44].

3.4. DoS attacks

DDoS/DoS can be broadly take two forms; Attacks that flood networks resulting in bandwidth degradation and attack that consume resources and eventually crashing services [46]. In figure 5, some methods of attack are shown.
The magnitude of DDoS/DoS attacks increases considerably when an unlimited amount of unknown sources are used. In the case of DDoS the attack occurs in two phase where initially zombies are compromised and recruited and eventually these zombies launch attacks on the victim[41][13]. Buffer and stack overflow vulnerabilities in are commonly exploited by attackers[31].

Malicious code is used to start agent tools to provide access to the victim’s system once these vulnerabilities are detected and consequently the DDoS agent code is installed.

### 3.5. DoS attack techniques

Figures 5a, 5b, and 5c show some common techniques used for DoS/DDoS attack such as agent setup, agent activation and network communication.
The two most popular types of DDoS attack networks model in current use today are the Agent-Handler model and the Internet Relay Chat (IRC)-based model. The Agent Handler model is shown in Figure 5d, comprising clients, handlers, and agents. The client is the medium through which the attacker communicates within the DDoS system and uses software packages scattered throughout the internet termed handlers which the clients uses to communicate with the agents. This allows the attacker to hide himself among the many clients participating in the attack. Attackers will usually try to install the handler software on a compromised system, and then use these handlers to communicate with agent’s software which is located on a compromised system.
International Journal on Computational Sciences & Applications (IJCSC) Vol.4, No.1, February 2014

from which to launch their attacks [36]. As described in Figure 5e, IRC (Internet Relay Chat) uses a communication channel to connect the client to the agents. An IRC communication channel aids an attacker through the use of “valid” IRC ports for conveying instructions to the agents [36]. In IRC, the attacker easily conceals his presence due to the extremely high volumes of traffic flowing on the servers.

![Figure 5d. DDoS Agent-Handler model[36]](image)

Agent software in an IRC network communicates messages within the IRC channel thus allowing the attacker to easily see the list of the agents as they become operational [36]

![Figure 5e. DDoS IRC-Based Attack Model [36]](image)

5. INSTALLATION DDoS AGENT

Attackers install malicious DDoS agent code either actively or passively onto a secondary victim. In Active DDoS agent installation methods, an attacker probes the network for vulnerabilities, and then executes scripts to gain unauthorized entry into the system, while silently installing the DDoS agent software. Before installing DDoS software, attackers first utilize scanning tools, to identify potential secondary victim systems. These scanning tools allow attackers to select ranges of IP addresses from which to scan. The tool will then proceed to return information such as each
IP address, open TCP and UDP ports, and the underlying OS [10]. In the case of passive DDoS installation methods, the secondary victim accidentally causes the DDoS agent software to be installed, either by opening a corrupted file, or visiting malicious web-sites [36].

6. DOS/DDOS DEFENSE MECHANISMS

Just as in any security setting, it is virtually impossible to completely isolate risks associated with DoS attacks. In this section we describe the Avoid –Detect- Prevent cycle approach to mitigating the risk of DoS attacks.

6.1. Avoid

Avoidance plays a key role in the successful implementation of any efficient defensive strategy. In an attempt to analyze DoS attacks and guide against future occurrence, a lot of technical data has to be obtained (e.g. network topologies, vendor agreements etc.). The data can also be acquired by monitoring traffic at network and host levels. This baseline data would help organizations in determining services that are critical. With this information, it becomes relatively easier for organizations to focus security strategies on service that are likely to have a relatively higher impact on business processes should they be affected by a DoS attack.

6.2. Detect

The heterogeneous nature of modern networks has to a large extent resulted in a corresponding increase in the complexity of networks. To this end it is important that detection systems are able to detect, prevent, and alert personnel of any possible DoS attacks in real time.

Modern Intrusion Detection Prevention Systems (IDPS) come equipped to combat these attacks and maintain state [43]. Detection systems should provide multiple detection mechanism, alerts, response mechanisms [44], and short detection time with low false positive rate [43]. These intrusion detection systems can take several forms such as anomaly detection, signature-based detection, as discussed below [18].

6.2.1. Signature-based detection

Signature based detection is usually used to detect known attacks. In this approach packets are analyzed to see if they conform to a known attack and based on that a decision is made. A database is maintained of known attacks against which network traffic is compared. Even though databases are constantly updated to reflect new threats, it possible for new attacks to be ignored by signature based systems [18] [41][47].

6.2.2. Anomaly-based detection

Anomaly based detection systems examine network traffic and application behavior and compare the traffic against existing ‘normal’ traffic patterns and thresholds. Some anomalous patterns that can be captured include [59]:

i. Misuse of network protocols such as overlapped IP fragments
ii. Uncharacteristic traffic patterns such as more UDP packets compared to TCP
iii. Suspicious patterns in in application payload
Machine Learning algorithms, Neural networks, Bayesian Learning and statistical techniques are some of the most common techniques used in anomaly based detection [12], [13] and [14].

6.3. Prevent

The primary objective of prevention is to detect attacks in the initial stages and prevent them from escalating. This is normally done through the use of distributed packet filtering mechanisms relying on information from local routing with the view of preventing flooding [3][15][16].

6.4. Reaction

Reaction techniques require the use of efficient incidence response and backup systems coupled with filtering of excessive traffic to mitigate the effects of attacks. In addition to the defensive techniques discussed above, several techniques have also been implemented to mitigate DoS and DDoS attacks. In [13], a technique for anomalous pattern for HTTP flood protection is proposed. This technique tunes a level of rate limiting factors using feedback. In using this approach, attacks are efficiently mitigate and legitimate traffic is allowed.

Specht and Lee’s [18] 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.

In a split-protocol implementation based on the Bare Machine computing paradigm (BMC)[37], no operating system is required. Because most DoS/DDoS agents are OS based, it is virtually impossible to run any agent code on the systems that are designed based on BMC paradigm.

7. AUGMENTED SPLIT ARCHITECTURE

Augmented Split-protocols (ASp) require a minimum of three servers, i.e., a Connection Server (CS), Resource Allocator (RA) and Data Server (DS). The CS establishes the connection via SYN and ACKs. When the HTTP GET is received by the CS, it sends an inter server packet to RA, this Inter Server Packet (ISP) contain the detail information about the GET. When the RA gets ISP, it creates its own TCB entry and sends ACK to the client and RA reserve resources for particular GET; also at the same time it sends an inter-server packet message to DS (referred to as a Delegate Message DM1). 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). In this architecture, CS does not reserve any resources for received GET request. However, it forwards GET to RA and intern RA reserve resources and state of the request (TCB Table). When CS sends or receives FIN, or FIN-ACK it sends information to the RA through the ISP, and RA deletes the TCB records belongs to the specific request. Retransmission and packet losses are also managed by RA. In this architecture, CS and DS does not reserve any resources for specific GET request. In this mechanism, RA is master and DSs servers as slaves they only follow the instruction from RA. RA knows the distribution of data on various DSs accordingly it sends DM1. The CS also handles the TCP ACKs for the data and the connection closing via FINs and ACKs. Typically, the RA has information about the requested file (i.e., its name, size, and other attributes), and the DS has the actual file (the RA may or may not have a copy). When the DS gets DM1, it 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 RA. The receipt of DM2 closes the state of the request in the RA. Furthermore, when CS reaches a threshold value, it migrates to a new server. It enables an alternate Connection Server (called CS* for convenience) 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 \[38\] 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 \[39\]. More details of migration and role changeover are given in a Split-protocol technique for Web Server Migration \[38\]. 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*, 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.

8. DESIGN AND IMPLEMENTATION

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 6 shows a high level design structure of 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 \[5\] because there is no process information contained in the entry.

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 [4]. By implementing limited ACKs, the linear performance improvement continues up to 4 DSs [5]. 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.

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 reduce to the point that the network cannot be used. Most of the time, the attacker identifies the primary 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 a 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 a strong defensive mechanism against Smurf attacks.

9. EXPERIMENTAL SETUP

The experiments were conducted using a prototype server cluster consisting of Dell Optiplex GX260 PCs with Intel Pentium 4, 2.8GHz Processor, 1GB RAM, and an Intel 1G NIC on the motherboard. All systems were connected to a Linksys 16-port 1 Gbps Ethernet switch. Bare PC Web clients capable of generating 5700 requests/Sec were used to create the server workload.

While attacking the server, there was not any other traffic going to the server, which was not connected directly to the Internet. The experiment was done without any network intrusion prevention and detection system or any firewall installed, so that all packets from the client machine that reached the server were captured. From the wire shark, it was possible to see that no packets were lost during the capture in the server. The experiment was repeated several times, by varying LOIC/ parameters. The first three experiments tested the TCP option with 10, 100 and 1000 parallel connections. The fourth experiment tested the attack over the HTTP protocol with 100 parallel connections. A second experiment was conducted using HOIC with varying number of thread 1, 2, 3, 4, 5 and 30 keeping the same security setting as the LOIC experiment.

10. PERFORMANCE MEASUREMENT

Figure 8, describes protocol transaction time for 4k resource file size on WAN subnet. We have compared transaction times with No-Split, Split system and M-Split system. The transaction time depends on the distance between client and server on a given network topology. In Split architecture, the server component (CS, RA, and DS) is located at different subnets. For convenience, we have placed CS at the same distance but varied DS by plus or minus one hop. We have noted there is a delay 976 microseconds when DS is placed one hop further than the CS. Furthermore, we have observed that when DS is placed closer to the client in comparison to CS distance, the transaction time was lesser by 674 microseconds. For larger file, multiple DSs involved to get faster transmission of data [3].

![Figure 8. Protocol Transaction Time](image-url)
As shown in figure 9, we have studied CPU utilization of three systems (Single system, Split system (CS-DS) and M-Split system (CS-RA-DS) at 6000 requests /Sec. CPU of Single system is almost at saturation point with 95%, Split system 45% and M-Split system is only at 20% CPU utilization. For availability, point of view M-Split system is freely available. In addition, for bigger resource file size of 128K single server can just handle up to 735 requests/second and CPU utilization reaches 95%, whereas CPU utilization of CS in Split system is 5% and in M-Split just 1%.

Figure 10 shows the total CPU utilization of all components of the systems for 4K resource file size. Overall CPU utilization of M-Split system is 87% and Split system 88% and Single system 95%.

Figure 9. CPU Utilization three systems at 6000 requests /Sec.

Figure 10 shows the total CPU utilization of all components of the systems for 4K resource file size. Overall CPU utilization of M-Split system is 87% and Split system 88% and Single system 95%.

Figure 10. CPU Utilization overall systems at 6000 requests /Sec.
Figure 11 illustrates the CPU utilization for varying LOIC/ parameters for TCP option with 10, 100 and 1000 parallel connections with five clients. The CPU utilization of CS is less than 5% and DS utilization was 10% and attack over the HTTP protocol with 100 parallel connections CS utilization around 5% and DS utilization is around 70%. And we found there is no effect of UDP protocol 10,000 threads CS CPU utilization was around 40% for the DS were around 1% only. This behavior is same as genuine clients, and we do not see the effect of DDoS attack even though LOIC clients are connected on the same LAN. Figure 12 shows the utilization CS/DS under HOIC attack with five clients, there is also no effect up to five threads; however, for 30 threads CPU is 96%, which was expected. The both experiments with LOIC and HOIC, CS and DS were performing normal servers as if there is no DDoS attack.

11. CONCLUSION

In multiple ways, the DDoS attack involves the attacker, the intermediary, and the victim. Connection server in Split-protocol architecture does not reserve any resource for all requests it
receives, so it can handle many connection requests. In our experiment, we have noted that for a large resource file, CS CPU utilization was only 1% as compared to 95% of the single server. Since there are many DSs in the system, they can handle very large loads without compromising services. Furthermore, the self-delegating mechanism in the split-protocol allows the server to deny any additional request to process and changes his identity within a single TCP connection. As shown in Figure 1a, toggling the same IP address between multiple servers minimizes the incoming load on Split-servers. As shown in the figure 7, client only communicates with the CS, and only handles SYN and does not reserve any resource for connection requests, therefore, logically CS appears very large. CS is capable of handling many fold more requests than the number of requests generated by genuine clients or DDoS attackers.

ACKNOWLEDGEMENTS

The authors would like to thank Dr. Ramesh Karne, Dr. A. L. Wijesinha and IT department at Shaw University, just everyone!

Authors

Dr. Bharat Rawal, has conducted research in the area of computer networks, including wireless networks, Split-protocol designs and analyzes, and network performance evaluations, HPC and Network security. He was the author and co-author in several papers in networking and security area. Currently, he has focused on solving a big integers and data compression in Split-protocol infrastructure. He is now server as CIS program coordinator and teaches computer science courses at Shaw University.

REFERENCES


S. Specht, and R. Lee “Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures,”


