Botnet in DDoS Attacks: Trends and Challenges

Nazrul Hoque, Dhruba K. Bhattacharyya, and Jugal K. Kalita

Abstract—**Threats of distributed denial of service (DDoS) attacks have been increasing day-by-day due to rapid development of computer networks and associated infrastructure, and millions of software applications, large and small, addressing all varieties of tasks. Botnets pose a major threat to network security as they are widely used for many Internet crimes such as DDoS attacks, identity theft, email spamming, and click fraud. Botnet based DDoS attacks are catastrophic to the victim network as they can exhaust both network bandwidth and resources of the victim machine. This survey presents a comprehensive overview of DDoS attacks, their causes, types with a taxonomy, and technical details of various attack launching tools. A detailed discussion of several botnet architectures, tools developed using botnet architectures, and pros and cons analysis are also included. Furthermore, a list of important issues and research challenges is also reported.**

Index Terms—**DDoS attack, botnet, mobile botnet, IP traceback, DDoS prevention.**

I. Introduction

As advances in networking technology help connect every nook and corner of the globe, the openness and scalability are giving birth to a large number of innovative, interesting and useful online applications. With the proliferation of these IT-enabled applications, more and more important and valuable information is flowing across public networks. Networks are usually designed to make efficient use of shared assets among network users [1] and network-based computer systems play a vital role in our personal as well as professional activities. Today, the Internet interconnects billions of computers, tablets and smartphones, providing a global communication, storage and computation infrastructure. Furthermore, the integration of mobile and wireless technologies with the Internet is currently ushering in an impressive array of new devices and applications.

These tremendous technological advances in terms of speed, accuracy, reliability and robustness of the modern Internet has made significant impacts on our day-to-day activities and people now rely on the Internet to share valuable and confidential personal as well as business information with other network users. On the other hand, because of the high reliance on the Internet, some people also make use of the weaknesses of the Internet to paralyze it. For example, one of the major weaknesses of the Internet is speed mismatch between edge routers and core routers. Inappropriate configuration of routers is also another common weakness of the Internet. Due to such weaknesses, networked systems have often become the targets of various attacks, which are launched in order to unlawfully gain access to important and confidential information or to damage useful resources of a business competitor using different attack tools [2], [3]. Although, in the recent past, many significant developments in constructing firewalls and cryptographic systems have taken place, they are not free from limitations. Defense mechanisms that identify intrusions provide another way to protect networked systems from attacks. In spite of all these safeguards, almost every day, new and complex attacks are being created. However, denial of service attacks are still the most frequent and usually the most devastating ones.

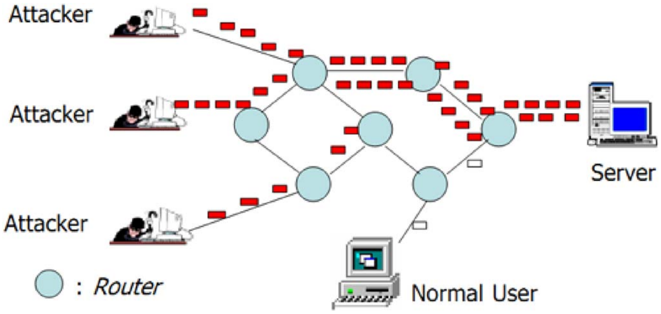


Fig. 1. DDoS attack.

A. Distributed Denial of Service (DDoS) Attack

DDoS is a coordinated attack, generated by using many compromised hosts. An attacker initially identifies the vulnerabilities in a network to install malware programs on multiple machines to bring them under his control. Then the attacker uses these compromised hosts to send attack packets to the victim without the knowledge of the compromised hosts. Depending on the attack packet intensity and the number of hosts used for attacking, commensurate damage occurs in the victim network. If the number of compromised hosts is very large, it may disrupt a network or a Web server in a very short period of time. Some examples of DDoS attacks include smurf, fraggle and SYN flooding.

The aim of a DDoS attacker is to disrupt a network so that it cannot provide any services to legitimate users (Fig. 1). To launch an attack, an attacker generally follows four basic steps. (i) information gathering to scan a network to find vulnerable hosts to use them later to launch an attack, (ii) compromising the hosts to install malware or malicious programs in the compromised hosts (called zombies) so that they can be controlled

Manuscript received January 8, 2015; revised May 8, 2015; accepted July 10, 2015. Date of publication July 16, 2015; date of current version November 18, 2015. This work is supported by MHRD, under FAST proposal and UGC, Government of India under SAP level-II. The authors are thankful to both the funding agencies.

N. Hoque and D. K. Bhattacharyya are with the Department of Computer Science and Engineering, Tezpur University, Tezpur 784028, India (e-mail: nhoq@tezu.ernet.in; dkb@tezu.ernet.in).

J. K. Kalita is with the Department of Computer Science, University of Colorado, Springs, CO 80309 USA (e-mail: jkalita@uccs.edu).

Digital Object Identifier 10.1109/COMST.2015.2457491

TABLE I
COMPARISON WITH EXISTING SURVEYS CONSIDERING THE DISCUSSION OF BOTNET RELATED TOPICS

Work	Compare with prior surveys	Taxonomy of DDoS	Architecture and Topology		Botnet C&C		Botnet Detection		DDoS attack tools	Botnet tools		Issues and challenges	Recomm-endation
			S	M	S	M	S	M		S	M		
[4]	×	×	√	×	√	×	√	×	×	×	×	×	×
[5]	×	×	√	×	√	×	√	×	×	×	×	×	×
[6]	×	×	√	×	√	×	√	×	×	×	×	×	×
[7]	×	×	×	×	×	×	√	×	×	×	×	×	×
[8]	×	×	√	×	√	×	√	×	×	×	×	√	×
[9]	×	×	√	×	√	×	√	×	×	×	×	√	×
Our survey	√	√	√	√	√	√	√	×	√	√	×	√	√
S=Stationary and M=Mobile													

only by the attacker, (iii) launching the attack to command the zombies to send attack packets with specified intensities to the victim, and (iv) cleaning up to remove all records or history files from memory.

A DDoS attacker often aims to attack one or more of the following targets (i) routers, (ii) links, (iii) firewalls and other defense systems, (iv) victim's computer and network infrastructure, (v) victim's OS, (vi) current communications and (vii) victim's applications. Some prominent factors of DDoS attacks are (i) existence of high interdependencies in Internet security, (ii) limited availability of Internet resources, (iii) many conspiring against a few, (iv) not collecting intelligence and resources, (v) use of straightforward and simple routing principles, (vi) mismatch between core and edge network design issues and speed, (vii) laxity in network management, and (viii) the common practice of sharing of resources.

B. DDoS Attack Using Botnet

A botnet is a collection of many malware infected machines called zombies that are controlled by a malicious entity called the botmaster. A botmaster remotely controls the zombies and instructs them to perform malicious activities through commands. The way the bots are controlled depends on the architecture of botnet command and control mechanisms, which may be IRC, HTTP, DNS or P2P-based. These botnets are used to commit cyber crimes such as sending spam mail, launching denial-of-service attacks or stealing personal data such as mail accounts or bank credentials. It is common knowledge that approximately 80% of all email traffic is spam and most such messages are sent through botnets.

C. Organization of the Paper

The rest of the paper is organized as follows. Section II discusses prior work and contributions. The background of DDoS attacks, a generic strategy for launching DDoS attacks and a detailed taxonomy of DDoS attacks are described in Section III. The evolution of botnet technology and their use in launching DDoS attacks are elaborated in Section IV. A comparison of stationary botnets is presented in Section V. A brief introduction to mobile botnets is given in Section VI. In Section VII we present a practical design approach for building both stationary and mobile botnets. DDoS attack detection, prevention and tolerance approaches are discussed in Section VIII. A brief discussion of DDoS attacks in the cloud, and in the context of Big Data and software defined networking is provided in Section IX. Finally, Section X has our concluding remarks and a list of issues and challenges for DDoS attack detection, prevention and tolerance.

II. PRIOR WORK AND CONTRIBUTIONS

In the past few years, several scholarly review articles have been published on botnets [4]–[9], and these attempt to elucidate the technological aspects of botnets, architectural issues, defense mechanisms, botnet-based attacks and the development of botnet-based tools. Feily *et al.* [4] describe techniques for botnet detection in four distinct categories: signature-based, anomaly-based, DNS-based and mining-based. The authors describe major characteristics of botnets. They use a number of parameters such as the rate of real-time detection, false positive rate, protocol and structure independence and the use of encryption mechanisms to compare botnet detection techniques. Zeidanloo *et al.* [6] present a taxonomy of botnet detection techniques and discuss these techniques in terms of two basic categories: setting up of honeynets and developing intrusion detection systems (IDSs). The second category is described in terms of two sub-categories, viz., signature-based and anomaly-based. Further, the authors classify anomaly-based methods into two sub-categories based on deployment, viz., host-based and network-based. Bailey *et al.* [5] present a brief review of botnet research, evolution of botnets and future trends, and modern approaches to botnet technology and defense systems. The authors also summarize propagation mechanisms used by bots, command and control mechanisms and attack classes. A detailed discussion of the relationship between network visibility, botnet invariant behaviors and existing botnet-based techniques is also a part of this article. Zhang *et al.* [7] report several botnet-based attack defense mechanisms. The authors summarize and classify recently introduced botnet flux features and detection techniques. Further, the authors also compare several survey papers on botnets (Table I). Recent research on detection or mitigation of fluxing botnets is also highlighted in this work. Rodriguez *et al.* [8] introduce a taxonomy of botnets in terms of life-cycle behavior. They also provide a comprehensive analysis of the botnet life-cycle to help botnet researchers understand the working principles of botnets. Further, they also discuss how to design detection mechanisms against botnet attacks. Silva *et al.* [9] explain four classes of command and control systems at a high level and their pros and cons. A significant number of host-based and network-based detection techniques are also discussed to explain their effectiveness. In addition, the authors discuss several defense techniques against botnet-based attacks. Further, they enumerate research challenges and highlight future trends in botnet technology.

Although these articles primarily focus on the design of botnets, generation of botnet-based attacks and detection of botnet attacks, most fail to provide a detailed study of recent developments of botnet design, issues related to stationary as well as mobile botnets and cost-effective botnet-based attack detection and the detection of such attacks in the context of sophisticated DDoS attacks. In this survey article, we discuss several stationary and mobile botnet architectures, their topologies and command and control mechanisms. We also discuss many botnet-based tools and the pros and cons of these tools from the perspective of DDoS attack detection. We highlight a list of issues and research challenges in this evolving field. Furthermore, we attempt to provide a practical approach to designing a botnet-based DDoS attack detection technique for both stationary and mobile networks.

III. DDoS Attack: Background, Strategy and Taxonomy

A DDoS attacker requires more diligence and is usually more damaging than a DoS attacker. It is distinguished from other attacks by its ability (i) to deploy its weapons in a coordinated and distributed way over the Internet, and (ii) to create a large collection of malicious traffic patterns by aggregating dispersed forces. In addition to causing damage to a victim either for personal reasons or for material gain, the attacker may also attempt to break the victim's defense system for fun or to show prowess or for cheap popularity.

DDoS attacks can be generated in two different ways: direct attack and reflector attack. In a direct attack, a large number of attack packets are sent to the victim machine directly. In this attack, the attacker spoofs the source IP address so that the response is misdirected and goes elsewhere. In case of an indirect attack, many innocent intermediate nodes known as reflectors are used to generate an attack. An attacker sends packets that require responses to the reflectors with the packets' inscribed source address set to the victim's address. The attack packets can be constructed using TCP, UDP, ICMP or IGMP protocols.

A. DDoS Attack Taxonomy

Over the past decade, the sizes and frequencies of DDoS attacks have increased spectacularly as attackers exploit botnet technology and other recent high-speed Internet access technologies to consume their target's resources. Igure *et al.* [10] provide taxonomies of attacks and vulnerabilities in computer systems. The largest DDoS attack grew 10 times in size from 2005 (10 Gbps) to 2010 (100 Gbps) as reported by Arbor's 8th Annual World-wide Infrastructure Security Report.[1] Due to this dramatic increase of DDoS attacks, network security defenders must focus on protecting the network community from two basic types of DDoS attacks: (i) network/transport layer DDoS attacks and (ii) application layer DDoS attacks. We classify DDoS attacks occurred in network/transport and applications layers into a taxonomy as shown in Fig. 2 and discuss different types of attacks based on attack generation approach, network architectures, victim type, attack rate and attack impact.

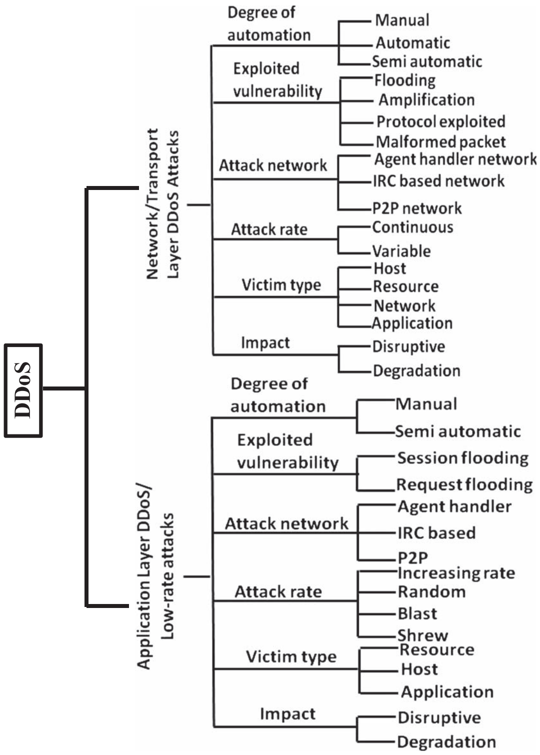


Fig. 2. Taxonomy of DDoS attacks.

1) Network or Transport Layer DDoS Attacks: The main target of this type of attacks is to overwhelm the network infrastructure consisting of servers, routers and switches by sending a large volume of attack traffic. These attacks can be generated by exploiting protocol weaknesses. Network/Transport layer attacks can be further characterized according to degree of automation, exploited vulnerabilities, types of attack networks used, attacks rates generated, victim types and impacts of the attack.

(a) Degree of automation: A DDoS attacker can attempt to execute a network, transport layer and application layer DDoS attack either manually, automatically or semi-automatically. In a manual attack, the attacker gathers vulnerability information on the hosts by scanning the network to install malicious programs. These compromised hosts are later used to send attack packets to the victim machine(s) in response to manual instruction. In case of automatic network/transport layer attack generation, no direct communication between the attacker and the compromised machines exists. Based on pre-specified attack start time, attack type, duration and source IPs of the victim machine(s), the attack initiates

[1] https://www.arbornetworks.com

automatically. Semi-automatic attacks are in between the above two categories of attacks; the addresses of the attack machines, attack type and attack durations are specified manually at the time of attack generation.

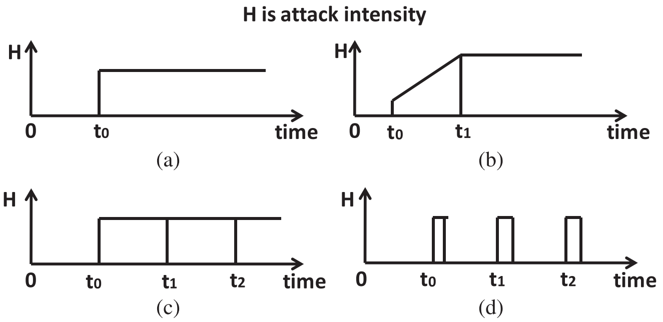
(b) Exploited vulnerability: To launch layer specific DDoS attacks, various types of vulnerabilities in a network are exploited by an attacker, including weaknesses of the protocols used, such as TCP, UDP, ICMP, HTTP, FTP and TELNET. Such vulnerabilities may lead to attacks such as flooding, amplification or malformed packet attack. Semantic and brute-force attacks are launched with a large amount of network traffic to overwhelm the service of a victim machine.

(c) Attack network: In this category of attacks, the attacker can use or hire a network of compromised machines to launch a network/transport layer DDoS attack. The attack network can be of several types such as (i) agent handler, (ii) IRC network, and (iii) P2P network. In case of (i), the attacker has control over the handlers and it uses the agents to send attack traffic to the victim network. The agent-handler architecture is very effective in launching large DDoS attack traffic of various types instantly. The type (ii) network includes a large number of compromised nodes called bots, which are connected to form a network called botnet, which is controlled by a botmaster. In type (iii), each node is able to communicate directly with every other node in the network. In this network, a centralized control mechanism doesn't exist.

(d) Attack rate: A network or transport layer DDoS attack can be launched with various attack rates. It can be either constant rate, increasing rate or variable rate [11]. In a constant rate attack, generally an attacker sends attack traffic to the victim-end at a steady rate with small variations within a limited range. In an increasing rate attack, the incoming attack traffic at the victim-end increases gradually or drastically, and it continues until the bandwidth of the channel(s) saturates. Such an attack type can be detected easily. On the other hand, in case of a variable rate attack such as a pulsing attack, the attack traffic usually consists of several huge bursts of data for short durations repeatedly at regular or irregular intervals. An important characteristic of this attack type is that the amplitudes and heights of the pulses may be constant or random. The fourth type, i.e., a subgroup attack is another variant of the variable rate network/transport layer DDoS attack, which is a combination of pulsing and constant rate attack types. Fig. 3 shows all these attack dynamics.

(e) Victim type: Depending on the type of victim targeted such as a single host, a network of hosts, resources or applications, the attacker can take various actions to launch network or transport layer DDoS attacks.

(f) Impact: In a network or transport layer DDoS attack, depending on the volume of attack traffic used, the severity of the damage on the victim machine or network can vary. The impact due to a DDoS attack either may be disruptive or degradation of performance.



(a) (b) (c) (d)

Fig. 3. DDoS attack rate dynamics.

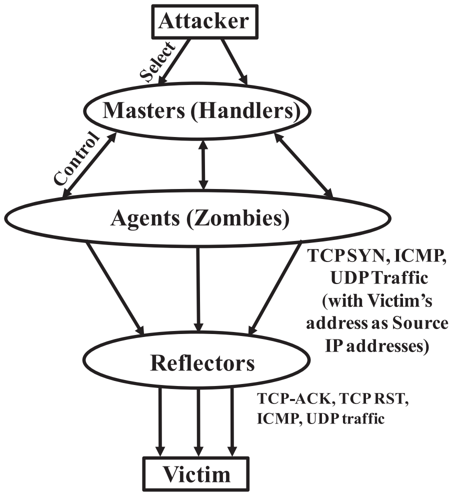

Attacker
Select
Masters (Handlers)
Control
Agents (Zombies)
TCPSYN, ICMP, UDPTraffic (with Victim's address as Source IP addresses)
Reflectors
TCP-ACK, TCP RST, ICMP, UDP traffic
Victim

Fig. 4. Reflector based TCP, UDP and ICMP flooding attacks.

Here are a few examples of network/transport layer attacks. In TCP SYN flooding, attack is generated by sending a large number of TCP SYN connection requests to overwhelm the server. To initiate the attack, the attacker creates packets with random source IP addresses for each packet, with the SYN flag set in each packet to open a new connection to the server from the spoofed IP addresses. On receipt, the victim responds to spoofed IP addresses and waits for confirmation that never arrives. As a consequence, victim's connection table fills up waiting for replies and the server is not able to accept new connection any more. Hence, legitimate users are deprived from accessing the server. Reflector based DDoS flooding attacks generated by TCP, UDP and ICMP protocol is shown in Fig. 4.

In UDP flooding, due to its connectionless nature, the UDP protocol does not require any connection setup procedure to transfer data. To initiate this host-based flooding attack, the attacker sends a UDP packet to a random port on the victim system. In response, the victim system determines what application is waiting on the destination port. If it finds that no application is waiting on the port, it generates an ICMP packet saying *destination unreachable* to the forged source address. If the number of UDP packets delivered to the ports on the victim is large, the system goes down. In ICMP flooding, which is also known as the ping-of-death attack, the attacker sends a large number of ICMP packets continuously to the server without waiting for replies from the server. After a while the server becomes unable to accept any more packets.

2) Application Layer Attacks: Application layer DDoS attacks generally target the HTTP protocol with an objective to exhaust limited resources available to Web services. In comparison to network/transport layer attacks, these attacks consume lower bandwidth. The attacker usually customizes them to target a particular Web application by sending requests to tie up resources deep inside the affected network. To accomplish their malicious designs, such attackers require only limited network connections. Typically, such attacks are not trivial to identify because they look similar to legitimate traffic and the volume of traffic is also not too large. The three main reasons behind the difficulty in detecting this type of DDoS attack [12] are: (i) Obscurity, the use of legitimate TCP and UDP connections during the attack makes it difficult to distinguish it as illegitimate traffic. (ii) Efficiency, the need for fewer connections to launch an attack successfully makes it efficient. (iii) Lethality, it can overwhelm resources of services quickly, resulting in denial of services regardless of the abilities of the hardware of the host.

Like network/transport layer attacks, application layer attacks can also be further classified into multiple subclasses considering the degree of automation, exploited vulnerabilities, types of attack networks used, attack rates generated, victim types and the impacts of the attacks. We introduce each of these subclasses below.

(a) Degree of automation: This class of attacks can be manual as well as semi-automatic, depending upon the degree of automation used when an attack is launched. When a manual DDoS attack is focused on the application layer, after identifying the vulnerable hosts to compromise, the attacker executes the application with command and control processes. On the other hand, in a semi-automatic DDoS attack, after the identification of the victim machines, the attacker manually specifies the attack rate, attack type and attack duration. The attack generation is still automatic using the compromised hosts. Although, an automatic application layer DDoS attack is not commonly found, generation of attack traffic is possible with the help of malicious code that executes automatically.

(b) Exploited vulnerability: Some application layer DDoS attacks are launched by exploiting weaknesses of protocols such as HTTP, FTP and TELNET against a server or a host. Some common examples of this class of DDoS attacks are session flooding and request flooding. In a session flooding attack, the attacker aims to cause malfunction in the victim server by sending sessions at a higher rate than for normal users, whereas in request flooding, the attacker sends a large number of request packets to the victim machine at a rate higher than for normal users to devastate the machine.

(c) Attack network: In an application layer DDoS attack, an attacker can also launch the attack by creating or hiring a network of victim machines. Such attack networks are usually of three types, viz., agent handler networks, IRC networks and P2P networks. Unlike agent handler and IRC architectures, in a P2P network, each node can communicate with any of the nodes in the network. The intention of the attacker is to trick a large number of client computers running P2P software into requesting a file from the intended target of the DDoS attack to overwhelm the target site with traffic.

(d) Attack rate: Like network/transport layer DDoS attacks, the HTTP GET flooding attack traffic also can be of several types depending upon the traffic patterns generated over time. In general we have four distinct types: (i) increasing rate, where the attack rate increases gradually with time, (ii) pseudo-random flooding, where the attacker attempts to flood the victim(s) by varying the traffic intensity at random, (iii) instead of flooding with increasing rates or random changing of the flooding request traffic, shrewd attackers launch assaults by periodically sending bursts of HTTP GET requests, (iv) blast flooding, which is launched by flooding with high volume of attack requests for a long duration of time.

(e) Victim type: During an application layer attack, an attacker may choose any server such as a Web server, mail server of an organization, any host in a network or a popular Website as a victim of the attack. For example, using an HTTP GET or HTTP POST flooding request, an attacker may disrupt the server or can bring down a Website.

(f) Impact: An application level flooding attack may attempt to exhaust the resources of a server so that it is unable to provide services to legitimate clients. Such application layer DDoS attacks are disruptive whereas in a degradation attack, the attacker attempts to bring a server down so that degraded performance results.

Some examples of application layer attacks are discussed below.

- Request-Flooding Attacks: An attacker initiates these attacks by sending legitimate requests at a higher rate such as HTTP GET, HTTP POST and DNS queries to a server in order to overwhelm its session resources.
- Asymmetric Attacks: In these attacks, the attacker attempts to consume server resources such as CPU, memory or disk space by sending high workload requests at normal rates. The main goal of this attack is to consume such resources of a server in order to degrade the performance of services.
- Repeated One-Shot Attacks: In this type of attack, an attacker sends a high workload request to many TCP sessions to degrade the performance of services. This type of attack is also considered an alternate (secret) way of executing request-flooding and asymmetric application-layer attacks.
- Application-Exploit Attacks: To initiate these attacks, an attacker takes advantage of the vulnerabilities that arise during the running of an application on a target system. Here, the attacker attempts to cause a fault in the target's operating system by running applications to gain control of one or more applications, the system or network. Some common examples of such attacks are Cross-site

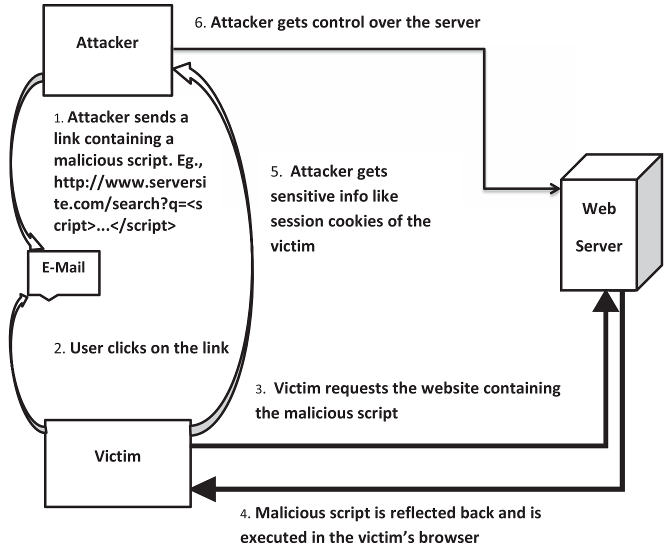

Attacker
6. Attacker gets control over the server
1. Attacker sends a link containing a malicious script. Eg., http://www.serversite.com/search?q=<script>...</script>
5. Attacker gets sensitive info like session cookies of the victim
Web Server
E-Mail
2. User clicks on the link
3. Victim requests the website containing the malicious script
Victim
4. Malicious script is reflected back and is executed in the victim's browser

Fig. 5. XSS attack.

scripting (XSS), SQL injection (SQLIA), buffer overflows, scripting vulnerabilities, cookie poisoning and hidden-field manipulation. Fig. 5 depicts a typical scenario of XSS attack.

In addition to the above attacks, there are several other application-based attacks, which are related to the HTTP protocol. These application-based attacks are of four types, as discussed below.

(i) HTTP attacks: This type of application-based attacks usually occurs in three forms. (i) HTTP malformed attacks, where the attacker attempts to consume or obfuscate the server resources by sending large invalid HTTP packets. An example of this worm is Zafi.B, which uses malformed HTTP GET requests. (ii) HTTP request attacks, which try to flood Web servers with legitimate HTTP requests. Some examples of this attack type are HTTP GETS and POSTS. (iii) HTTP idle attacks, where the attacker opens HTTP connections but remains in idle state without actually sending a complete HTTP request. An example of this attack is slowloris that attempts to involve indefinite dribbling out of a small number of bytes per packet to keep the connection from timing out, but never completes the request.

(ii) DNS attacks: This application-based attack has four distinct sub-categories. (i) DNS query (or answer) malformed packet attacks, DNS query-length buffer overflow and DNS query buffer overflow attacks, where the attacker sends or receives invalid DNS packets that cause degradation or failure of DNS infrastructure. (ii) The second sub-category consists of man-in-the-middle and DNS cache poisoning attacks where, the attacker attempts to intercept DNS queries and place erroneous information within the DNS infrastructure. (iii) The third sub-category consists of DNS amplification attacks, where the idea is that a small spoofed DNS request (e.g., 128 bytes) can cause the generation of a large DNS response (e.g., 1500 bytes) to an unsuspecting target as shown in Fig. 6. (iv) Finally, the fourth sub-category consists of DNS dictionary attacks, which generate a massive number of requests to a DNS server in order to extract information that approximates a full zone transfer. The attacker uses a large dictionary of words to exhaustively scan the namespace of possible host names in the hope of hitting most of DNS records in the victim server.

(iii) VoIP attacks: This category of application-based attacks also has four distinct sub-categories. (i) SIP invite flood attacks, which send bogus SIP INVITEs to overwhelm the session initiation protocol (SIP) registration. (ii) SIP call setup request attacks, which send a high rate of SIP call setup requests to a SIP proxy server in an attempt to disable it. (iii) SIP malformed packet attacks, which send invalid packets to SIP devices in an attempt to disable them. (iv) Real-time transport protocol (RTP) flood/quality of service (QoS) attacks, which are used to transport the voice portion of a call onto a network.

(iv) SMTP attacks: There are two types of SMTP attacks. (i) SMTP error DoS, mailbox DoS (excessive email size) and SMTP mail flooding, which attempt to overwhelm email servers. (ii) SMTP buffer overflow attacks, where different SMTP commands can cause the SMTP server to crash or execute arbitrary byte-code that could lead to a system compromise.

We have already stated in the previous sections that botnet based attack launching technology is more serious in both network/transport layer as well as application layer DDoS attacks. In the recent past, significant developments in attack sophistication using botnet technology have been noticed. We provide some fundamentals of botnet based DDoS attack generation in the next section. We discuss the evolution of botnet technology and its various characteristics in the context of DDoS attack generation in two categories, viz., (i) stationary botnet and (i) mobile botnet.

IV. DDoS Attack Using Stationary Botnets

In recent times, most sophisticated DDoS attacks have been launched using botnet technology. The growth of botnet technology is enabling the generation of various types of DDoS attacks due to the flexibility and power of the technology. The four main reasons behind the preference for this technology for the attackers are: (i) inclusion of a large number of zombie nodes allow generation of a powerful flooding attack quickly, (ii) great difficulty in finding the identity of the actual attacker, (iii) ability to use protocols to bypass security mechanisms, and (iv) difficulty in detecting in real time due to its similarity with the normal traffic. Fig. 7 shows an example of DDoS attack generated using botnet. A brief description of the botnet components is given below.

1) Attacker: The attacker configures the bots to initiate an attack. First, it compromises a machine to install malicious code and gains control of the machine once it joins the control server.

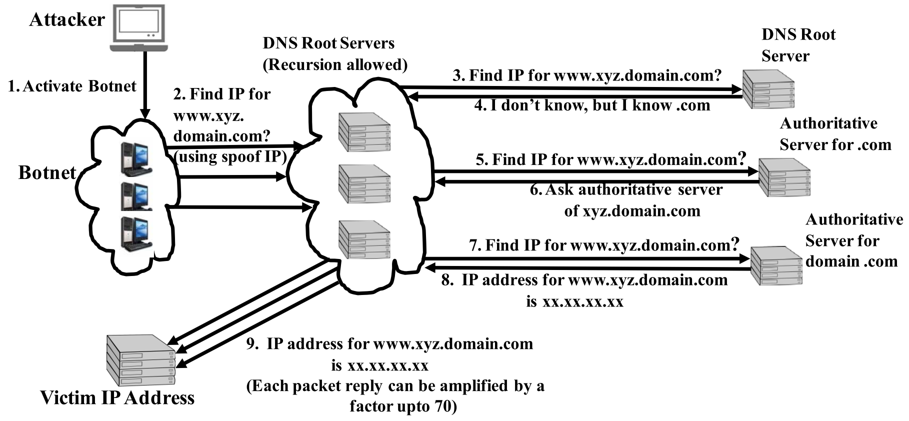

Attacker
1. Activate Botnet
Botnet
2. Find IP for www.xyz. domain.com? (using spoof IP)
DNS Root Servers (Recursion allowed)
DNS Root Server
3. Find IP for www.xyz.domain.com?
4. I don't know, but I know .com
Authoritative Server for .com
5. Find IP for www.xyz.domain.com?
6. Ask authoritative server of xyz.domain.com
Authoritative Server for domain .com
7. Find IP for www.xyz.domain.com?
8. IP address for www.xyz.domain.com is xx.xx.xx.xx
9. IP address for www.xyz.domain.com is xx.xx.xx.xx (Each packet reply can be amplified by a factor upto 70)
Victim IP Address

Fig. 6. DNS amplification attack.

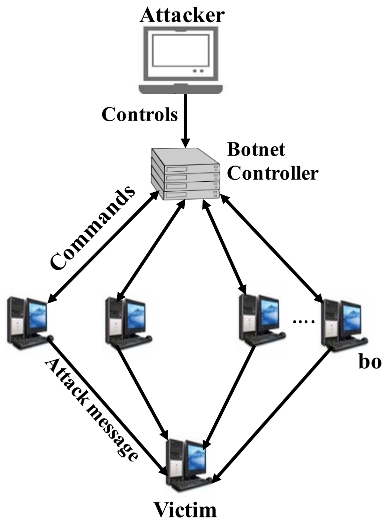


Fig. 7. Botnet attack.

2) Botnet controller: It works as a botnet command and control server that can send/receive communication commands from the connected parties.
3) Compromised host: Once a machine is compromised by installing malicious code, it works as a bot in a botnet.
4) Victim: It is the target host for the attacker that receives a large number of attack packets from the compromised hosts.

Though botnets have become integral to most sophisticated DDoS attacks, they originated from Internet Relay Chat (IRC), a text-based chat system that organizes communication in channels. The purpose of the IRC system was very affirmative and it provided services for message sharing, administrative support along with simple games and other services to chatters. Later bad actors started exploiting its power to execute malicious activities such as launching attacks on networks. In a report by Arbor Networks, it was noted that in one instance six command and control (C&C) servers employed over 25,000 infected Windows machines to attack CMS systems using brute force. The ability to use several thousand infected hosts to execute simultaneous attacks under the control of a botmaster has made this attack approach extremely effective. In 2014, a significant excalation was observed in volumetric attacks using botnets with increased size, frequency and complexity. The largest attack reported to Arbor Networks was 400 Gbps, more than a 100 percent increase from the previous year.[2]

A. Botnet Characteristics

Botnets are generally characterized by the type of C&C system used for communication. The communication between the bot and the botmaster occurs as per the specification of the C&C system. Although in the literature, several types of C&C systems are discussed, *centralized* and *distributed* mechanisms are mostly used for communication. Both types have their own advantages as well as limitations. To address these limitations, an effective botnet technology has been introduced of late, called *peer-to-peer botnet*, one that is more robust and difficult to defend. To counter a botnet based attack, it is essential for the network defender to know the malware code and enhancements that may have been applied to such code. In addition, the botnet architectures, topologies and protocols used for attacks should be carefully analyzed when developing a DDoS defense system. With the successful convergence of Internet and traditional telecommunication services, the threat of botnets over essential basic communication services, including the 3G and 4G wireless networks, has become alarming.

B. Botnet Models

Botnet-based DDoS attacks are generally launched using three basic models [13]: (i) agent handler model (ii) IRC botnet model and (iii) web-based model.

Agent handler model: In this model, there are four participants, viz., (i) the attacker or the master, (ii) handlers, (iii) agents and (iv) the victim, as shown in Fig. 8(a). The first player, i.e., the attacker initially compromises some hosts in a network to bring them under his control. The second set of players, i.e., the handlers are composed of malicious software residing on remote machines that are used by the attacker for a DDoS attack. It is common for an attacker to launch DDoS attacks from a victimized computer (handler) to make it more difficult

[2] http://krebsonsecurity.com/tag/arbor-networks/

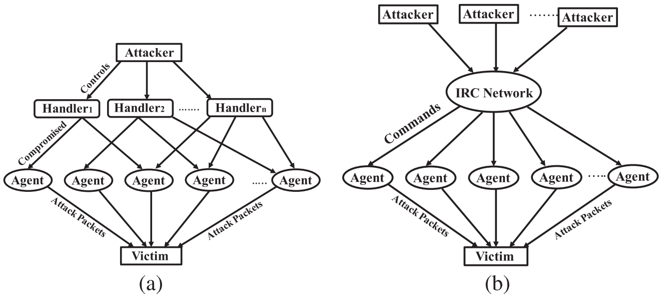


Fig. 8. Agent handler and IRC architecture. (a) Agent handler architecture. (b) IRC architecture.

to trace the attack back to the attacker (client). The third set of players, i.e., the agents, consists of software on compromised machines that actually performs the DDoS attack. It also can be thought of as a subset of the handlers residing on the same systems. A large number of agents are usually involved in a DDoS attack. To launch a DDoS flooding attack, an attacker may exploit the weaknesses of protocols such as TCP, UDP or ICMP. In general, the victimized (hacked) computers are engaged in the flooding attack without knowledge of the owners of the machines. The use of a large number of handlers in the attack helps conceal the malicious use of a handler computer. The fourth player, i.e., the victim, may be a single target machine or a number of target machines.

IRC-based model: The Internet Relay Chat (IRC), a text-based chat system that organizes communication in channels unwittingly facilitated the birth of botnets. IRC-based DDoS attacks are now most popular because of their power to generate a huge volume of attack traffic instantly. Typically, this system installed with a bot can spread very fast and automatically exploit multiple vulnerabilities. After successful installation, a full backdoor generally resides on the system, including an IRC component to establish communication between the computer and a remote IRC server; this backdoor is controlled by the attacker. To communicate with clients, the agents use an IRC communication channel as shown in Fig. 8(b). The agents are communicated through IRC ports, which makes it difficult for the defender to trace DDoS command packets. An effective tool called LOIC (Low Orbit Ion Cannon) [14] has been introduced to exploit the IRC protocol. To launch a DDoS attack via IRC, an attacker simply logs into a malicious IRC server, authenticates and issues commands to many zombies at once or to individual bots within private windows. IRC botnet operators tend to keep their botherds small in size, by rolling out updates and many minor variants of the code to create dozens, if not hundreds of smaller botnets on various servers. Depending on the botnet architecture used, topology and protocol, the effect on the victim network can vary. The ability to avoid a single point of failure, such as capacity to survive when authorities shut down a hostile IRC server, or to bypass the defense mechanism mostly depends upon the size and technology of the botnet used with the IRC model. IRC based attacks may involve many different software code variants and protocols. As in the case of agent-handler DDoS attacks, IRC-based DDoS attacks may also involve TCP, UDP or ICMP protocols.

Web-based model: In the past few years, Web-based reporting and command have emerged as a potential alternative platform for botnet command and control, in spite of the heavy dominance of the IRC-based method. Bots are now not only used to report statistics to a Website, but are also configured and controlled through encrypted communication and sophisticated PHP scripts. Some common advantages of Web-based control over IRC are: (i) friendly interface, which is easy to set up, configure and also easy to rent out, (ii) the availability of improved and meaningful reporting and command utilities, (iii) consumption of less bandwidth due to distributed load, which allows bigger botnets, (iv) the ease in concealing traffic as well as making filtering difficult due to use of port 80, and (v) the possibility of botnet hijacking, which is not usually possible via chat room hijacking.

Another important advantage of Web-based C&C is the ability to control larger and more conspicuous botherds. This has made it more difficult for security professionals to identify and filter out such traffic over TCP port 80. It has also made it extremely difficult for the defender to distinguish botnets from DDoS attacks. We note that malicious actors are constantly improving botnet technology to enhance the effectiveness of DDoS attacks. An emphasis during the evolution of new attacks is to enable the mastermind to distance oneself from the actual attack. In recent times, malicious actors have also attempted to complicate attacks by introducing new layers to the architecture. For example, in a distributed reflector DDoS attack, the attacker takes advantage of uncompromised devices that unwittingly participate in the attack. Another common example is the use of DNS servers as reflectors. In such a case, the DNS server sends several times more traffic to the victim than what was sent to it. So, it has become essential for network defenders to understand or predict trends in the evolution of botnet technology to be prepared to defend their networks effectively in the future. We strongly feel that defenders' in-depth analysis of attack trends including extrapolation to the future will help build appropriate solutions to mitigate them. It seems clear that with the recent use of peer-to-peer C&C systems, the next generation botnets are likely to be hybrid P2P.

C. Botnet Formation Life Cycle

The botnet formation life cycle comprises of five phases, viz., initial infection, secondary infection, connection, malicious activity and maintenance [5]. A graphical representation of the life cycle is depicted in Fig. 9. In Phase 1, the mastermind of the botnet sends malware to infect target machines, whose payloads are bots. In Phase 2, attempt is made to log into an IRC server or another communication medium by the infected machine to set up the botnet. In Phase 3, the owner of the bot is paid by the spammer for getting the access right. The spammer orders the botnet in Phase 4 to send spam or malicious code to many machines in the victim network, and finally, in Phase 5, the maintenance and update activities are initiated.

D. Stationary Botnet Architecture

A DDoS attacker can set up a botnet with various architectures depending on the availability of machines to infect and

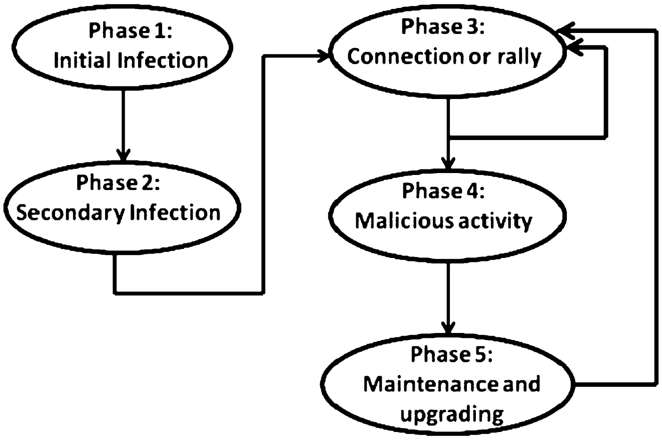


Fig. 9. Botnet life cycle.

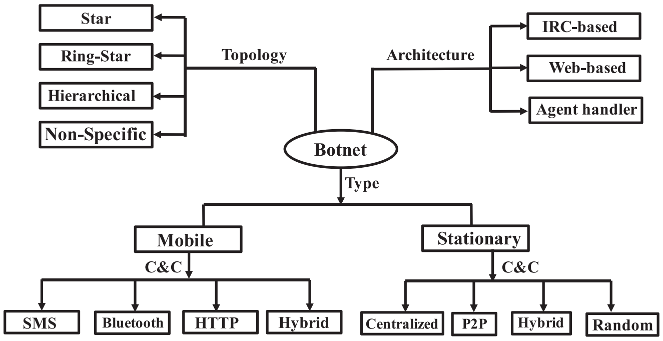


Fig. 10. Taxonomy of botnet.

pre-specified objectives. Although, initially most attackers used centralized structure to communicate among bots and servers, recently more complex communication structures have evolved with different network topologies. Fig. 11 shows three different botnet architectures. We provide a taxonomy of botnets based on their types, models and C&C mechanisms in Fig. 10.

E. Communication Mechanisms

Four types of communication mechanisms are typically used by most attackers [15]. (i) star topology, where the attacker gives more weight to a single centralized C&C resource component to communicate with all bot agents. This central component is responsible for issuing new instructions directly to each bot agent. (ii) ring-star topology, a logical extension of the star topology, uses multiple servers that are connected in a circular form to provide C&C instructions to bot agents. To manage the botnet, communications take place among multiple command systems, and if any of the individual severs fails or is detached permanently from the network, the remaining servers take up the responsibility of controlling the botnet. (iii) hierarchical topology follows the methods used in the compromise and subsequent propagation of the bot agents themselves. Bot agents have the ability to proxy new C&C instructions to previously propagated progeny agents. However, updated command instructions typically suffer latency issues, making it difficult for a botnet operator to use the botnet for real-time activities. (iv) non-specific topology, where no centralized C&C infrastructure exists.

A DDoS attacker uses three communication protocols in a botnet. (i) IRC protocol, which is used to send messages to other bots in the botnet. This protocol was designed for both one-to-many conversations as well as for one-to-one conversations. A major limitation of this protocol is that security devices can easily block IRC traffic. (ii) HTTP protocol, another widely used protocol, which is advantageous because of its ability to bypass the security system during communication. It is usually difficult to identify malicious communication that uses the HTTP protocol because of its similarity with legitimate traffic. (iii) P2P protocols, which is recently gaining tremendous popularity due to its distributed support.

F. Botnet Command and Control Systems

Identifying the C&C system used by an attacker plays a major role in detecting DDoS attacks. Typically, a DDoS attacker uses any of four different C&C server approaches, viz., central, P2P, hybrid and random. Each of these systems has its own advantages as well as limitations. We analyze them in the context of DDoS attack generation.

Central C&C server: A central server is preferred by most DDoS attackers because it provides a simple, low latency, anonymous, real-time and efficient platform for the botmaster. Through this server, a botmaster can communicate directly with the bots. Although this approach has several advantages, it also suffers from two important limitations: (i) it has a single point of failure, i.e., if the server fails, the botnet may fail to perform since no bot can receive any messages; and (ii) the messages sent to the servers by a host can actually be triggered and sent by defenders. An IRC botnet is configured to send text messages from a client to the IRC server or even among the servers themselves using a client-server model; it can function in a distributed environment. An attacker typically follows four basic steps to execute an IRC based botnet attack. (i) Creation, where the attacker may add malicious code or simply modify existing code out of numerous highly configurable bots found on the Internet. (ii) Configuration, where a victim machine automatically connects to a selected host, as long as the bot is installed on that machine. The attacker also may configure the system for restricted access and to protect the channel to the bots for personal or for business purposes. (iii) Infection, where bots propagate directly or indirectly. In direct propagation, vulnerabilities of the services or operating systems are exploited and are usually associated with the use of viruses. On the other hand, indirect propagation uses other programs as proxy to spread bots. It may also use distributed malware through DCC (Direct Client-to-Client) file exchange on IRC or P2P networks to exploit vulnerabilities of target machines. Once the vulnerable systems are compromised, they may be used to spread the infection process saving time for the attacker to add other insecure victims. Some common examples of vulnerable systems are Windows 2000 and XP SP1, where the attacker generally finds unpatched or insecure (e.g., without firewall) hosts. (iv) Control, where the attacker is able to send

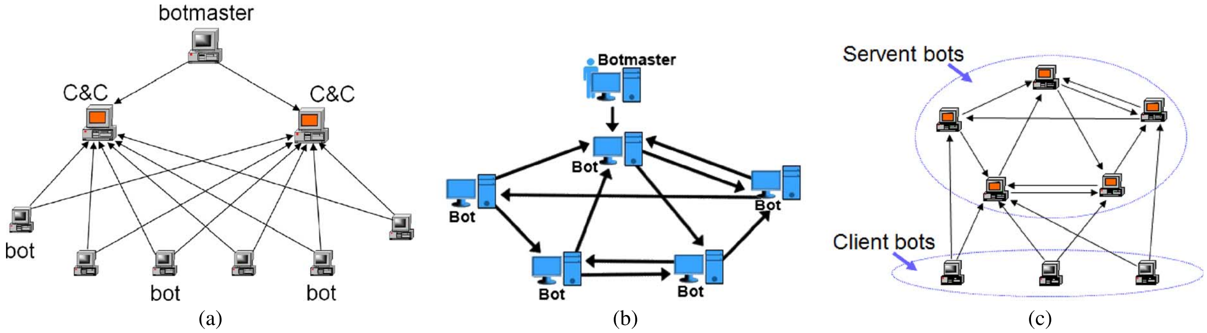


Fig. 11. Botnet architecture. (a) Centralized architecture. (b) P2P architecture. (c) Hybrid architecture.

instructions to a group of bots via an IRC channel to perform malicious tasks.

P2P C&C server: In recent times, P2P has gained tremendous popularity. Most people share resources, programs, documents, movies and games using this system. A significant development during the recent past, made to the original P2P server (originally implemented in 2002) facilitates the mounting of attacks. In P2P-based architecture, a seed list can be maintained with each host, and when a bot receives a message, it forwards it only to the private list of seeds. An important advantage of this system is that the botmaster needs to connect only one of the bots (peers) to send instructions over the network and each host periodically connects to its neighbor host to retrieve instructions from the botmaster. Other significant advantages of P2P C&C architecture are: (i) they are more robust and reliable than a centralized architecture, (ii) they are not easy to shut down, (iii) design complexity is not too high, and (iv) they have high survivability rate. However, an important limitation is that this architecture gives no guarantee of message delivery. The three most commonly found P2P architectures are discussed below.

(a) Unstructured C&C server: This type of P2P architecture does not restrict sending of messages from a host to other hosts. If the system does not maintain a seed list, the bot scans the network to gather information to identify another bot. Initially, the botmaster encrypts his message and passes it to any one of the hosts over the network. This is a simple, but secure structure, and the discovery of a host does not affect other hosts. Its advantages include (i) low design complexity, and (ii) very high survivability. However, it suffers from limitations including (i) no guarantee of message delivery, and (ii) low message latency.

(b) Structured C&C server: This type of P2P network is more efficient than the previous type. For effective routing, it maintains a distributed hash table (DHT) [16]. It creates mapping between the content and its locations. Some example DHTs are CAN [17], Pchord [18], Pastry [19], Tapestry [20] and Kademlia [21].

(c) Superpeer P2P overlay [22]: In this type of P2P network, all peers may not be equal. It automatically selects a small subset of peers as temporary servers to support network functions, such as search and control. Some common applications of this type are Skype, Fasttrack and Gnutella. However, due to its high visibility and vulnerability to targeted attacks, intelligent botnets generally do not prefer this design. Bots belonging to this class usually have a valid IP address and are not under firewalls or DHCP.

Hybrid: A hybrid C&C system is designed to exploit the benefits of both centralized and P2P models. It has two types of bots, i.e., servant bots and client bots. A servant bot contains static and routable IP addresses, and these bots behave both as clients as well as servers whereas client bots contain dynamic and non-routable IP addresses. Hybrid C&C systems can also be located behind firewalls without a global connection to the Internet.

Random C&C system: To provide a simple, yet secure platform, this type of P2P system is designed based on the principle that a single bot knows at the most one other bot. According to this topology, a sender bot or controller initially scans the Internet at random to identify another bot and once found, sends it a message in encrypted form. The simplified design principle makes this system attractive. The detection of a single bot is not enough to compromise the full botnet. Three major limitations of this type are: (i) high message latency, (ii) no guarantee of message delivery, and (iii) frequent detectability of the random probing behavior.

V. Some Stationary Botnets

In recent times, most sophisticated attacks on networks or Web servers have been launched by botnets. Several types of botnets have come into existence during the past few years. In this section, we introduce a list of botnets used for launching various types of attacks (Table II).

1) Agobot [23]: It is a multi-threaded bot written in C++/assembly language by a German programmer known as Axel Ago Gembe. Agobot is a pioneer IRC bot used for attack generation. Some important features of Agobot are: (i) password protected IRC client control interface, (ii) remote update and removal of the installed bots, (iii) execution of programs and commands for DDoS attacks and (iv) port scanning to find and infect other hosts.
2) SDBot [24]: This bot is equipped with several backdoor capabilities and information stealing routines. It can propagate through network shares and exploited vulnerabilities.

TABLE II
COMPARISON OF STATIONARY BOTNETS

Name of botnet	Year	No of machines used	Architecture	Attack type	Protocol used	Platform	Reference
GTbot	1986		Centralized	DDoS	IRC	Windows	[34]
EggDrop	1993		Centralized		IRC	Windows/Linux	[45]
Agobot	2002		Centralized	DDoS	IRC/ HTTP	Windows	[23]
SDBot			Centralized	DoS	IRC	Windows	[24]
RBot	2003		Centralized	DoS	IRC	Windows/Linux	[25]
Sinit	2003		P2P		IRC/HTTP	Windows/ Linux	[35]
Bagle	2004	230,000	Centralized/P2P	Spam sending	IRC	Windows	[36]
Phatbot	2004		P2P		HTTP/IRC	Windows	[36]
SpamThru	2006	12,000	P2P	Sending spam	IRC	Windows	[29]
Nugache	2006	160,000	P2P		IRC	Windows	[31]
Rxbot	2006		Centralized		IRC	Windows	[37]
Spybot			Centralized	DoS	IRC		[26]
Rustock	2006	150,000	Centralized	Sending spam/ DDoS	HTTP	Windows	[32]
MegaD	2007	500,000	P2P	DDoS		Windows	[28]
Srizbi	2007	400,000	Centralize	DoS/DDoS	HTTP	Windows/Linux	[29]
Storm	2007	160,000	P2P	DDoS	IRC	Windows	[31]
Conficker	2008	10.5 millions	P2P	Buffer overflow	HTTP	Windows	[27]
Torpig	2008	180,000	Centralized	Phishing/ man-in-the-middle	IRC	Windows	[30]
Grum	2008		Centralized	Sending spam	IRC/ HTTP	Windows	[32]
Asprox	2008	15,000	Centralized	SQL injection	HTTP	Windows	[38]
Bobax	2008	185,000	Centralized		HTTP/UDP	Windows	[36]
Kraken	2008	400,000	Centralized		HTTP	Windows	[40]
Waledac	2009	90,000	P2P	Sending spam	IRC	Windows/Linux	[41]
Cutwail	2009		P2P	DDoS	IRC	Windows	[33]
Donbot	2009	125,000	Centralized	DDoS	TCP	Windows	[42]
Festi	2010	25,000	Centralized	DoS/email spam	HTTP	Windows	[43]
TDL-4	2011	4.5 million	P2P	DDoS/DoS		Windows	[44]

3) RBot [25]: This Ruby IRC bot creates a large family of backdoors - remote administration utility programs. Once the backdoors are successfully installed, it allows a remote user to access and control it over a network or the Internet. A remote user with malicious intentions may be able to control an infected computer, usually without the knowledge or consent of the system's legitimate user(s) and uses the backdoors remotely to perform a variety of actions, such as stealing data, executing commands on the affected machine or accessing other machines on a local network.
4) Spybot [26]: Spybot is a successor of SDBot. It connects to a designated IRC server and receives commands from the botmaster through designated channels.
5) Conficker [27]: It is one of the most powerful and most effective computer worms that has infected millions of users in government and business in over 200 countries since 2003. It uses flaws in the Windows software and mounts dictionary attacks on administrator passwords to propagate while forming a botnet. It is not trivial to counter because it combines several advanced malware techniques.
6) MegaD [28]: It is a mass spamming botnet, initially identified in 2007. An important design advantage of MegaD bot is that it supports interaction with four types of C&C servers, viz., master servers (MS), drop servers (DS), template servers (TS) and SMTP Servers (SS).
7) Srizbi [29]: It is composed of a collection of computers (zombies) infected by the Srizbi Trojan Horse; which are at the command of the controller of the botnet called botnet herder. The operation of the Srizbi botnet depends on a number of servers through which the utilization of the individual bots in the botnet is controlled. These servers are redundant copies of each other, which protects the botnet from being crippled in case of system failure.
8) Torpig [30]: Torpig is another effective tool used to steal sensitive information such as bank accounts and credit-card data from its victims. It is one of the most dangerous Trojans horses to infect the Internet. To extract additional, sensitive information from the victim machines, it uses Phishing attacks.
9) Storm [31]: Storm has a backdoor component which allows remote clandestine access to infected systems. It harvests email addresses found on infected computers, delivers a downloader/dropper component to update itself or download additional malware. Additionally, it installs a rootkit to hide the presence of itself.
10) Grum [32]: It is a spam email sender botnet. Grum design is based upon two types of control servers for its operation. One type is used to push configuration updates to the infected computers and the other is used to instruct the botnet what spam emails to send.
11) Cutwail [33]: This is a simple, but effective spam email sender botnet. It uses a Trojan component called *pushdo* to install the bot on the victim machine. The bots connect directly to the C&C server, and receive instructions about emails to be sent. Once delivered, the bots report back statistics on email delivery and error messages to the spammer.
12) Rustock [32]: Rustock is a rootkit-enabled backdoor Trojan that is used to assist in distribution of spam emails. It can transmit more than 25,000 spam messages per hour from an infected machine.
13) GTbot [34]: GTbot uses legitimate IRC as a C&C server to launch flooding attacks with huge volumes of text messages. GTBot accesses an IRC channel with a huge

volume of traffic and then attempts to join the target channels and flood them with endless repetitive data. GTbot can cause normal users to become disconnected or their IRC client to freeze, because it cannot process the rapidly scrolling flood of garbage data fast enough. It is able to flood often up to 150 kbps of data through the IRC server and often incurs the owner, who usually gets free or cheap service, penalties for extra bandwidth consumption.

14) Sinit [35]: It is another backdoor Trojan that allows users with malicious intention to access a machine and connect it to a distributed botnet. Sinit uses P2P technology during communication.
15) Bagle [36]: This botnet was first introduced in early 2004. It is mainly used in proxy-to-relay e-mail spam and it is more impressive than Rustock as it can perform very effectively with fewer infected machines. The size of Bagle is estimated to be between 180,000 and 280,000 computers and it pumps out 8.31 billion spam emails daily. In comparison, Rustock's size is between 470,000 and 690,000 computers and generates 13.82 billion daily junk mails.
16) Phatbot [36]: Phatbot is another descendent of Agobot that uses the P2P botnet architecture. It uses IRC channels during communication. It allows a remote attacker to compromise the victim machine and link them to P2P networks. It uses the network to send a large amount of spam e-mail messages or to flood Web sites with data in an attempt to knock them offline.
17) SpamThru [29]: This botnet uses a custom P2P protocol to share information with other peers including IP addresses, ports and software versions of the control server and template servers. All the peers know about each other. The botnet is usually maintained by a central server. However, if the control server is shut down, the spammer can update the rest of the peers with the location of a new control server, as long as the spammer controls at least one peer.
18) Nugache [31]: It is another customization of the Trojan worm. Nugache uses P2P communications without any C&C server. This feature makes it undetectable and at the same time also provides a new level of resiliency for the botnet.
19) Rxbot [37]: It is a Windows based worm that uses IRC C&C server. This botnet has been used for many malicious activities such as DDoS attack, spam mail, fraud click and identity theft.
20) Asprox [38]: It is a botnet that emerged in 2007. It is used for sending phishing scams and to launch SQL injection attacks on Websites. According to BogusBiter, the detection of phishing Websites is done by other tools [39].
21) Bobax [36]: It is a very large spanning botnet that uses plaintext HTTP for communication with the C&C server. This worm exploits the DCOM and LSASS vulnerabilities on Windows systems.
22) Kraken [40]: It is another spam Trojan, used to spread spam from an infected machine. It uses encrypted communication with C&C and communicates using UDP and TCP protocols.
23) Waledac [41]: Waledac is a sophisticated worm that uses social engineering and certain client side vulnerabilities in order to propagate. The worm has the ability to download and execute binaries, act as a network proxy, send spam, mine infected computers for data such as email addresses and passwords, and perform DoS attacks.
24) Donbot [42]: This botnet is especially used to send pharmaceutical and stock-based e-mail spam. It includes roughly 125,000 individual computers to send 800 million spam messages in a day.
25) Festi [43]: This botnet is mostly used as an email spammer and for launching DoS attacks. It uses the client server based C&C mechanism, where a set of servers is dedicated to manage the botnet.
26) TDL-4 [44]: It is a very effective new generation botnet. It was able to infect almost more than 4.5 million machines within the first three months of 2011. It attempts to infect the master boot record of the target machine, which makes it extremely difficult to identify.

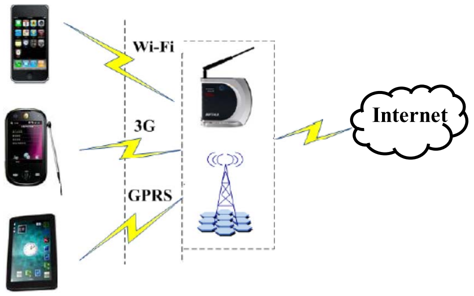


Fig. 12. Mobile botnet architecture.

VI. MOBILE BOTNETS

A mobile botnet is composed of a collection of compromised smartphones, remotely controlled by a botmaster via C&C channels. The ability of mobile devices to communicate with Internet services through various techniques such as High Speed Downlink Packet Access (HSDPA), Evolution Data Optimized or Enhanced Voice Data Only (EVDO), Universal Mobile Telecommunication System (UMTS), Enhanced Data Rates for GSM Evolution (EDGE) and General Packet Radio Service (GPRS) has attracted intruders to exploit them as a platform for launching DDoS attacks. Most attackers use the mobile platform during initial stage of launching a DDoS attack due to reasons such as limited battery power, limits on network traffic and lack of a fixed IP address when smartphones are connected to a network. [46]. An example mobile botnet architecture is shown in Fig. 12.

A. Mobile Botnet Characteristics

Due to the constraints as stated above, mobile environments are typically less secure and their specific characteristics pose several challenges to mobile botnet and malware detection [47].

Mobile malwares are the real threats for smartphones because they can force the assignment of a new IP address from the cellular network by resetting the data connection [48]. Some inherent challenges of mobile botnets are discussed below.

1) Developed with long-term intentions: Mobile botnets are developed with long-term intentions and the botmaster attempts to keep the botnet safe and undiscovered by applying different strategies.
2) Flexibility: Attackers update the botnets and bots frequently and change the codes periodically. In addition, botnet control strategies are changed frequently and new methods are developed to recover and restore the detected bots, within a short time.
3) Use standard protocols: Most attackers prefer to use standard protocols to establish their communication infrastructure. The latest generation of botnets use the standard HTTP protocol to impersonate normal Web traffic and bypass current network security systems.
4) Work in silent mode: A mobile botnet developer always avoids unnecessary or suspicious use of CPU, memory or other computer resources, which may help identify their presence.
5) Social engineering: Most recent mobile botnet designers employ a social engineering approach to propagate in the network.
6) Resource limitations: Mobile device resources such as CPU, memory and battery life are limited. Therefore, it is difficult to deploy existing botnet detection solutions for mobile botnets.
7) Device-specific characteristics: Some characteristics are speecific to mobile devices such as mobility, strict personalization and different types of connectivity, technology convergence and a variety of capabilities.
8) Diversity in infection: A mobile botnet may use different communication media (e.g., SMS, MMS or Bluetooth) along with the Internet to spread. This diversity makes it difficult to detect infection processes using most current security systems.
9) Distributed security management: Typically a mobile botnet lacks central security management to track and monitor security threats and update the security policies on mobile devices.

B. Command and Control Mechanisms in Mobile Botnets

In a mobile botnet, the botmaster is responsible for controlling channels for compromised nodes. If we can block the channel for the botmaster, the botnet will not be able to function. Typically three types of C&C mechanisms are used in a mobile botnet.

1) Internet-based (IP-based) C&C: IP-based C&C is similar to the P2P-based mechanism used in traditional network botnets.
2) GSM-based (SMS-based) C&C: In this C&C mechanism, a botmaster uses a phone to control the botnet.
3) Local wireless C&C: In this mechanism, the botmaster injects a command and allows it to travel through the net.

A C&C channel is usually responsible for circulating commands from the botmaster to the mobile bot. Since this channel is very crucial in the mobile attack process, designers need to be very careful. Typically, a mobile botnet uses four different channels during communication.

1) SMS C&C channel: In mobile communication, one convenient way of sending or receiving textual information between two communicating parties is SMS. In SMS-based C&C, the following features are usually supported.
   a) It propagates through SMS or MMS functions.
   b) The server of the service provider stores the messages while the recipient's mobile is switched off.
   c) It can hide malicious content.
   d) It can support multiple send and receive channel options.

   In order to stop detecting commands sent as SMS by a remote user, each mobile bot intercepts all incoming SMS messages before they reach the inbox. SMS messages containing the specific passcode are added while other SMS messages are allowed to pass through the inbox.
2) Bluetooth C&C channel: A mobile phone-based botnet that uses bluetooth to propagate control messages is almost similar to a Internet-based P2P botnet.
3) HTTP C&C channel: The Bluetooth and SMS channels are usually not enough to retrieve information from the server. An additional channel commonly used to transmit information for the C&C channel, one that uses a common application protocol is HTTP. It provides services to communicate information to the control server. It is a bidirectional communication channel that can forward information between a mobile bot and the control server.
4) Hybrid C&C channel: This channel attempts to overcome the limitations of a single point of failure. It is composed of three components: (i) a propagation vector (ii) C&C channels and (iii) a mobile botnet topology. The hybrid design uses the efficiency of multiple C&C channels in order to satisfy multiple objectives: no single point of failure must exist in the topology, the cost of command dissemination must be low, network activities must be limited and battery consumption per bot must be low.

With rapid development of cellular networks and Internet access capabilities of smartphones using WiFi, GPRS, 3G and 4G, construction of mobile botnets has become the trend. Many security experts contend that large scale attacks can be launched from mobile networks. In recent times, networks of mobile devices or mobile botnets have become significantly involved in launching DDoS attacks. For example, Android.DDoS.1.origin is an Android Trojan that is used to mount DDoS attacks from a mobile botnet. This Trojan creates an application icon that can be used by a normal user. The Trojan connects to a remote server and transmits the phone number of the compromised device to criminals and then waits for further SMS commands. It can be used to attack a specified server or send an SMS.

TABLE III
COMPARISON OF MOBILE BOTNETS

Name of botnet	Year	Platform	Architecture	Attack type	Protocol used	Reference
Zeus	2007	Windows/Android	Centralized	Mobile Banking Attacks	HTTP	[50]
Tiger	2010	Android	P2P	Theft of Private Data	HTTP	[50]
DroidDream	2011	Android	P2P	Download of Malicious Applications	HTTP	[50]
Andbot	2011	Android	Centralized	Theft of Private Data	HTTP	[49]
Geinimi	2011	Android	P2P	Illegal Transactions	IRC/HTTP	[51]
MDK	2012	Android	P2P	Theft of Private Data	HTTP/IRC	[52]
MisoSMS	2013	Android	P2P	Stealing of SMS messages	SMTP	[53]

TABLE IV
COMPARISON BETWEEN MOBILE BOTNETS AND STATIONARY BOTNETS

Features	Mobile botnet	Stationary botnet
Uses of IP address	Private	Public
Battery power	Limited	Unlimited
Bandwidth	Limited	High bandwidth
Collection of	Mobile devices	Stationary devices
Firewall protection	No	Impose
Central security management	No	Usually yes
C & C protocol	SMS, MMS, Bluetooth	IRC, HTTP, P2P

C. Some Mobile Botnets

In this section we present several mobile botnets and discuss their characteristics.

1) Andbot: This mobile botnet is known for three important features: stealth, resilience and low-cost. It uses an effective C&C system called URL Flux [49]. It uses a centralized C&C topology.
2) Waledac: This is a Web-based mobile botnet that uses HTTP for communication through channels. The use of P2P technology has made Waledac effective as a spam mailer. Each infected mobile device communicates with others to exchange lists of active proxy servers. This is done through MMS messages communicated among the infected devices on the mobile network.
3) Ikee.B: It is a simple botnet that operates on jailbroken iPhones with almost the same capabilities as computer-based botnets. Some major characteristics of this botnet are the abilities to scan the IP range of iPhone networks, look for vulnerable iPhones on a global scale and self-propagation.
4) Geinimi: This is a Trojan malware that can compromise a significant amount of personal data on a user's phone and send it to remote servers. The major objective is to first construct an Android botnet. After installation on a user's phone, the malware can receive commands from a remote server that allows the owner of that server to control the phone.
5) Zeus: Zeus or Zitmo is able to infect a large variety of mobile operating systems such as Symbian, Windows Mobile, BlackBerry and Android, mainly by using social engineering approaches. It attempts to infect a mobile phone by sending an SMS that contains a fake URL to dupe users to download a security certificate which is, in fact, the Zitmo bot. It also attempts to intercept messages sent by banks to customers and authenticate illegal transactions by stealing mobile Transaction Authentication Numbers (TAC).
6) DroidDream: DroidDream is a silent botnet which does not make any unusual or suspicious use of the CPU, memory or other resources, which may uncover its activities. It is activated at night (11pm to 8 am) when mobile users are usually asleep. It attempts to gain root privileges on infected mobiles and tries to steal confidential information by installing a second application.
7) Tiger: This is a fully SMS controlled bot that detects C&C messages and attempts to uncover them. It can capture not only private SMS data but also can record voice call conversations and even background sounds.
8) MisoSMS: This is one of the largest mobile botnets that leverages modern botnet techniques and infrastructure. It infects Android systems and secretly steals user's personal SMS messages and send them to a C&C system.

A mobile botnet consists of a command and control center that governs a network of compromised mobile devices such as smartphones and tablets. Comparisons between mobile botnets and their stationary predecessors are shown in Tables III and IV, respectively. The comparison shows a clear distinction between these two types of botnets in terms of IP addresses used, power backup, available bandwidth, firewall protection and the existence of centralized management.

VII. BOTNET DESIGN: ISSUES AND CHALLENGES

During our literature review, we have observed that the requirements for designing an effective DDoS attack launching tool using a stationary botnet are not the same as those for mobile botnet technology. A mobile botnet based attack launching tool demands a set of special requirements due to its limited battery backup, limited bandwidth, lack of firewall protection or lack of central security management, in comparison to a stationary botnet based tool. So, the attacker as well as the defender needs to be aware of these while developing a tool to attack or defend a DDoS attack. An attacker considers the issues shown in Fig. 13 when designing a botnet architecture. The issues most relevant for stationary botnet design are shown

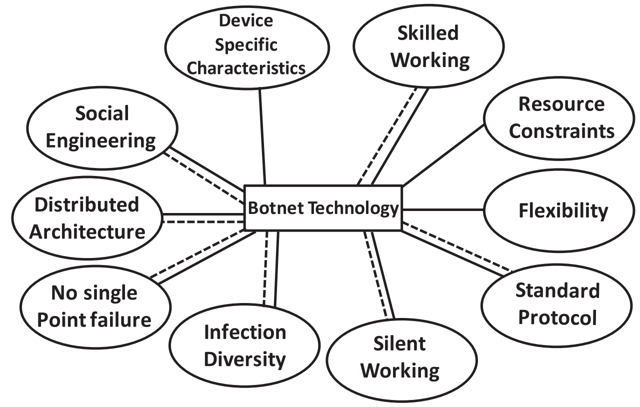


Fig. 13. Botnet design issues.

connected with dotted lines, whereas the relevant issues for mobile botnet design are shown connected with solid lines. Comparing centralized and P2P botnet communication, P2P botnet communication has advantages over centralized networks. In general, P2P communication is too complex to disrupt by the defender. Also, the failure of a single bot does not influence the entire network much. Further, passing commands through intermediate peers in a P2P network makes it more difficult for analysts to identify bot controllers and to determine the size of a network. Furthermore, another observation is that increasing the size of a botnet is not always effective if the goal is to inflict maximal damage because it also increases the visibility of the botnet. A botnet with a relatively small size, i.e., say with the number of bots within the *range* $\geq$ 15,000, but $\leq$ 20,000 can also be much effective in damaging a victim's Website or server if their combined bandwidth can be utilized properly with appropriately skilled coding. With the growing development of botnet technology and the increasing use of smartphones, mobile botnets have emerged as an effective platform for attackers to launch cellular network attacks such as SMS spam, DDoS attack and click fraud. However, in contrast to stationary botnets, a mobile botnet's design is influenced by factors such as device specific resource constraints (e.g., battery life) and flexibility. So, the coder for mobile botnets must be careful in handling these factors. Also, smartphones are more vulnerable to the attacker because the attacker can get access easily to a mobile phone through an SMS command and control system or via bluetooth [54]. Further, the P2P-based topology for mobile botnets allows botmasters and bots to publish and search for commands in a P2P fashion, making detection and disruption much harder. Furthermore, attackers prefer the use of HTTP based communication in mobile botnet communication, which generally helps hide the activities of the communication.

Now, we enumerate issues and challenges for botnet design. We believe that future attackers are grappling with how to design effective attack launching tools to inflict maximum damage. Our intention is to help improve the know-how of network security researchers and practitioners about the design trends in attacks tools, but our purpose is not to educate anyone in the design of attack launching tools to enable them to counter DDoS attack mitigation techniques or methods. It is only possible for a defender to protect a network by filtering malicious traffic when the defender has knowledge of the various ways an attacker can attempt to intrude the network.

(1) *Source IP spoofing:* It is an effective technique used widely by DDoS attackers. Although many researchers deem source IP spoofing to be of low relevance and low usefulness in the context of current botnet-based DDoS attacks, many attackers still prefer to use it due to its cost effectiveness. It is costly to manage or to hire a botnet and maintain its ownership. Even though ingress and egress filters are considered very effective in filtering traffic with invalid IP addresses, attackers still manage to bypass such protection mechanisms by using appropriate source IP spoofing schemes. Thus, providing a full-proof solution against source IP spoofing still remains an important research issue.

(2) *Degree of randomization:* Most attackers believe that a high degree of randomization of the header fields such as port addresses (source and destination) and sequence number along with partial or complete spoofing of source IP addresses, are enough for mounting a successful DDoS attack. But it is not true! It actually becomes easier for a defender to distil anomalous traffic from legitimate traffic when a high degree of randomization is used. A believable and effective tool should generate traffic with addresses within a feasible range. Any traffic with an arbitrary address (beyond a safe range) may lead to an obvious anomaly. So, we believe that sophisticated attackers will develop a tool able to generate attack traffic with careful and partial source IP spoofing, and randomization of other head fields without violating the feasible range. Network defenders must be prepared to counter such efforts of attackers.

(3) *Isolation vs. combination:* Most flooding tools such as Agabot UDP flooding generate attack traffic by exploiting either packet size randomization or source IP spoofing or randomization of other header fields. None of these tools are designed by carefully combining all these features. So, we believe that attackers will develop an attack tool that combines all such features and is able to generate flooding attacks including a wide range of protocols. Therefore, network defenders must think ahead now to develop methods to detect such efforts at feature combination.

(4) *Realistic TCP SYN flooding:* Improper balance between SYN and ACK packets and unusual service requests are the major sources of identification of TCP SYN flooding attacks. For each round of flooding, a sophisticated attack tool is bound to generate the numbers of SYN and ACK packets with balanced probability so as to avoid easy identification as anomalous traffic. The expert attacker is unlikely to generate any service requests for unusual IP protocol types, other than the most commonly used TCP or UDP. The use of other protocols will definitely lead to anomalies in the traffic. So, proper understanding of the protocols and the associated typical services in the context of a network are surely important for an attacker to develop an effective attack tool. Network defenders must be aware of such expertise in attack generation.

(5) *Removal of unique characteristics:* A knowledgeable attacker is likely to avoid generating any traffic with unique characteristics that stand out (such as the use of unusual or unrealistic spoofed addresses for source IPs and ports, or other

parameters, e.g., packet size, service type, granularity in delay setting), because it will help identify such traffic as anomalies easily. Therefore, an advanced attack tool will probably have a mechanism to filter out such traffic with unique characteristics before flooding a network. This is an issue a defense mechanism must be prepared to handle.

(6) *Low cost and limited bandwidth attack:* An attacker aiming to launch a DDoS attack using mobile botnet technology will have to be dependent on limited bandwidth and battery backup. So, to develop an attack tool based on a mobile botnet, a sophisticated attacker will have to be able to generate all variations of attack classes without violating these constraints. Obviously, a defender should also be able to look for such variations.

Next we discuss various DDoS detection, traceback, prevention and tolerance approaches and methods introduced in the past two years.

VIII. Defense Against DDoS Attacks

Due to increasing sophistication of attack behaviors, real-time detection of DDoS attacks is a challenging task. In the last two decades, network security researchers and practitioners have developed many approaches to detect DDoS attacks. Some approaches are centralized, others are distributed. Robinson *et al.* [55] claim that five issues should be considered in designing an effective DDoS defense system.

1) Since DDoS attacks detection is essentially a distributed problem, to detect DDoS attacks, an effective distributed solution is ideal.
2) A DDoS defense system should not harm legitimate users' activities.
3) A defense system must have adequate security mechanism against not only external, but also internal threats that may facilitate DDoS attacks from inside a network.
4) A defense system should be a scalable one with strong economic deployment incentives.
5) Defense should be designed in such a way that it may not be a complete solution for all problems but it can be a part of a complete solution that may be built over time.

Considering these five requirements of a DDoS defense system and various methods of DDoS attacks generation, Robinson *et al.* [55] suggest four criteria to classify DDoS attack traffic.

1) Site of action: Sites of action include source point of an attack, victim-end, victim's surrounding network or intermediate networks.
2) Discrimination of legitimate traffic: Any defense system should pose minimum effect on legitimate traffic. Some methods drop packets based on probabilistic analysis whereas some others drop packets radically at certain points.
3) Ease of deployment and interaction: Deployment of a defense system requires collaboration among different entities of a network to detect attacks.

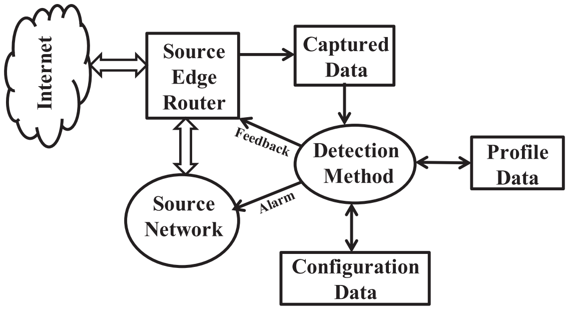


Fig. 14. Source-end DDoS detection architecture.

4) Effectiveness of defense mechanism: Effective DDoS defense should be able to detect any attack traffic with high detection accuracy and low false alarm rate.

A. Detection Approaches and Methods

Intrusion is an attempt to bypass or violate the security mechanisms of a system and an intrusion detection system uses enhanced processes to identify intrusions. Intrusion detection systems are designed to detect anomalous traffic in a network. To design an effective DDoS defense mechanism, the designers consider various security issues in a network. The main purpose of a detection system is to provide security to a system by detecting anomalous traffic that can disrupt system services.

1) Detection Approaches: In this section, we discuss four deployment points for DDoS defense mechanisms, viz., source-end, victim-end, intermediate and distributed [56].

(a) *Source-end*: A source-end DDoS defense system is very effective in stopping attacks as close to the source as possible. It reduces network traffic congestion and saves network resources. Placing a defense system in the source network is better than placing it further downstream. In this approach, network attack traffic can be stopped before reaching the target network, and it also reduces the chance of collision by filtering attack traffic before it aggregates with other attack traffic flow in the network. Moreover, source-end detection is not only easy to traceback but it provides high detection accuracy due to low network traffic flow aggregation at the source end. A source-end detection architecture is shown in Fig. 14.

The main advantage of source-end DDoS attack detection is that it can protect a network near the initial point of attack generation, reducing the chance of severe damage to the victim network. But, a major drawback of a source-end detection approach is collateral damage [57] to the legitimate traffic. Mirkovic *et al.* [58] propose a source-end DDoS defense system called D-WARD to defend DDoS attacks at the source-end. They state that the source-end is the only deployment point that can achieve good response selectiveness in case of high-volume, high-spoofing flooding attacks, which will make it a key building block for distributed defense systems.

(b) *Victim-end*: Most DDoS defense mechanisms are deployed at the victim-end for effective detection and defense of a system. Victim-end detection systems detect attacks either in a reactive or proactive manner. Source-end defense systems detect attacks close to the source of the original attack and

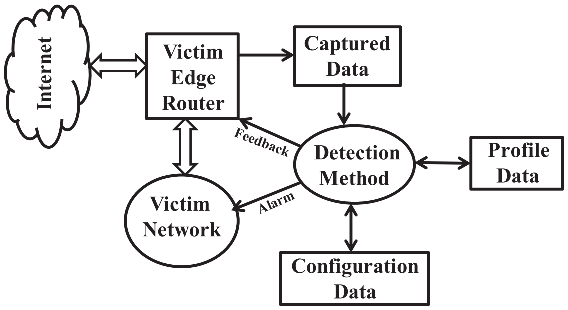


Fig. 15. Victim-end DDoS detection architecture.

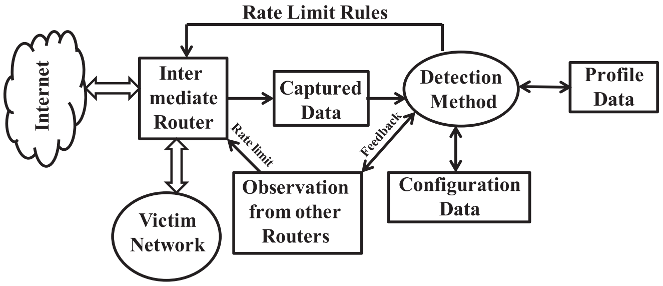


Fig. 16. Intermediate DDoS detection architecture.

hence the attack traffic cannot damage other parts of a network. In contrast, a victim-end defense approach cannot provide complete protection to DDoS attacks because high rate attack traffic may have already damaged the victim network. So, in many high rate attacks, victim-end based DDoS defense is not adequate. Besides, a victim-end defense system can drop all incoming traffic when the traffic rate is very high and as a consequence, legitimate traffic will to be able to pass to the network through a congested link between other parts of the network and the victim. A victim-end detection architecture is shown in Fig. 15.

(c) *Intermediate*: Network-based defense mechanisms are deployed mainly on the routers of the network system. The main advantage of a network-based defense mechanism is that proper actions can be taken in routers against malicious traffic before the malicious traffic is forwarded to the victim machine. Suspicious network traffic filtering in edge or core routers of a victim machine is another advantage of network-based defense. Some well known network-based defense mechanisms are (i) Router-based packet filtering [59] (ii) Detecting and filtering malicious routers [60], and (iii) Pushback [57]. The main drawback of network-based defense mechanism is overhead due to large network size. These detection mechanisms are not very effective and efficient for real-time defense against DDoS attacks. An intermediate detection architecture is shown in Fig. 16.

(d) *Distributed defense*: Distributed defense systems detect and protect a network in multiple phases. A source-end or a victim-end defense mechanism is not adequate in combating all types of DDoS attacks in an enterprise. Therefore, network security defenders design distributed defense mechanisms to detect DDoS attacks at multiple locations and prevent attacks using traceback of the source of attacks and traffic rate controlling mechanisms.

2) Detection Methods: Literature survey shows a significant number of publications on anomaly detection systems [61]–[69]. A large number of hardware-based network infrastructure security mechanisms are presented in [70]. People use different methods such as statistical, machine learning, soft computing and knowledge-based, for anomaly detection. A brief discussion of different DDoS detection methods and their effectiveness is presented here.

(a) *Statistical*: Many statistical approaches have been used for detection of anomalies in a network. Such systems use statistical methods such as entropy, principal components analysis, hidden Markov models, mutual information, correlation and covariance. Li *et. al.* [71] propose an entropy based DDoS attack detection method that calculates the distribution pattern of the attributes in network packet headers. Cumulative entropy is calculated to monitor network traffic behavior for a period of time instead of classifying the traffic as abnormal after initially detecting as abnormal in the first phase. In the second phase, an anomaly pattern is detected based on time instead of a threshold value set apriori. If abnormal behavior is continues for a certain period of time, only then the pattern is marked abnormal.

Feinstein *et. al.* [72] develop a method by computing entropy and frequency sorted distribution of packet attributes. In this method, entropy of the source addresses is computed for a packet window of size, say 1000, to determine the randomness or uniformity of the addresses. The amount of randomness is different for normal and attack conditions. In normal conditions, the entropy of the source addresses is less than in attack conditions. A low rate DDoS attack detection and traceback method using an information theoretic approach is proposed by Xiang *et al.* [49]. They calculate the difference between legitimate traffic and attack traffic using a generalized entropy metric and an information distance metric for detection of low rate DDoS attacks. Using experimental studies, they claim that the generalized entropy can detect an attack earlier than the traditional Shannon metric. The proposed information distance metric gives better result than Kullback-Leibler divergence approach.

Dynamic entropy can also be used to detect specific types of malicious traffic. A novel dynamic entropy-based model is proposed by Qi *et. al.* [73] to detect DoS attacks. This model uses netflow conversation correlation from different perspectives in a group of correlated events like request and reply. They compare dynamic and static entropy change rates in anomaly detection and find that the dynamic entropy method is more sensitive and more suitable for anomaly detection.

An information theory based DDoS attack detection algorithm is presented by Yu *et al.* [74] to classify attack traffic from the legitimate. During botnet attacks, the attacker uses controlled function(s), called zombies, to send malicious packets to the victim and hence the attack flow shows properties which are not followed by a legitimate flow in a short period of time. The method calculates distance between packet distribution behavior and suspicious network traffic flows and confirms DDoS attack flow if the distance is less than a predefined threshold; otherwise, the flow is marked legitimate.

TABLE V
COMPARISON OF EXISTING STATISTICAL METHODS

Work	Method	Centralized/ Distributed	Detection point	Effectiveness	Remarks
Mirkovic et al. [79]	D-WARD	Centralized	Source	High detection rate	D-WARD provides effective defence and in a few seconds it can detect many common forms of DDoS attacks.
Akella [80]	Sample-and-hold	Distributed	Source & Victim	High detection accuracy	Detects subtle and irregular changes in traffic patterns
Peng et al. [81]	CUSUM/ SIM	Centralized	Victim	100% accuracy	Detects all types of DDoS attacks, low computational complexity
Chen et al. [82]	DCD	Distributed	Victim & Intermediate	100% accuarcy in TCP SYN attack detection	Low rate UDP attack detection rate is low
Oke et al. [83]	RNN	Centralized	Victim	More accurate and powerful in DoS attack detection	Combines maximum likelihood detection method with RNN
Chen et al. [84]	SAR/ t-test	Centralized	Victim	False positives and false negatives are very low	Detects DDoS attacks based on two-sample t-test by incorporating the statistics of SYN arrival rate
Yu et al. [85]	Entropy/ IP Traceback	Distributed	Intermediate	Effective and efficient for DDoS detection	Detects DDoS attack based on entropy variation between normal and attack traffic
Fallah et al. [86]	IP Traceback/ TPDF	Centralized	Source	TDPF is effective in different attack scenarios	

Jin *et al.* [75] propose a model using Multivariate Correlation Analysis (MCA) for detecting SYN flooding attacks. This method is very simple, but can effectively differentiate between normal and attack traffic in real-time. They use correlation analysis on multiple features of normal network traffic and generate a normal profile. When testing network traffic, the method generates a test profile using the same correlation analysis and if the test profile deviates from the normal profile beyond a predefined threshold value, the test profile is marked attack traffic. This method can also detect subtle attacks, which are difficult to differentiate from normal behavior. Experimental results show high detection accuracy and real-time effectiveness for DDoS attack detection.

Yu *et al.* [76] propose a discrimination algorithm to distinguish DDoS attacks from flash crowds. They use flow correlation coefficient as the similarity metric for suspicious flows. They observe that similarity among DDoS attack flows is higher than that in flash crowd flows in a community network. An application layer DDoS attack monitoring method is proposed by Xie *et al.* [77] using the concept of document popularity. They capture spatial-temporal patterns of a normal flash crowd using an access matrix, and apply principal components analysis, and use the independent components to extract a multidimensional access matrix. During attack detection, first they obtain the dynamics of the access matrix using a hidden semi-Markov model and then detect attacks. Xie *et al.* [78] introduce a new scheme that detects application-layer-based DDoS attacks in early stages. They use a hidden semi-Markov model to describe browser behavior during application layer attacks. The browser behavior of a Web user is related to two factors: the structure of a Website and the way the user accesses Web pages. A new on-line algorithm called the M-algorithm is proposed to detect anomalies, reducing memory requirement and improving computational efficiency. A comparison of statistical DDoS attack detection methods is given in Table V.

(b) *Machine learning*: Machine learning enables a system to learn without being explicitly programmed [87]. Machine learning algorithms are mainly divided into two categories: (i) supervised learning and (ii) unsupervised learning. In supervised learning, the learning algorithm gains knowledge from labeled data and later uses the acquired knowledge to predict the class labels of previously unknown data. In unsupervised learning, the learner finds natural organization or structure in the data, without any labeled input. Such an algorithm typically works based on similarity or distance computation. Many machine learning algorithms have been used to detect anomalies present in network traffic [88].

Shon *et al.* [89] propose a hybrid machine learning approach called *Enhanced SVM* to detect network anomalies. They claim that among the variety of anomaly detection approaches, Support Vector Machines (SVM) are the best machine learning algorithms to classify abnormal behaviors. In order to provide unsupervised learning with a low false alarm rate, an enhanced SVM method can be used, one that combines both one-class SVM and soft-margin SVM methods. A DDoS attack detection method using a Naive Bayes classifier is proposed by Vijayasarathy *et al.* [90]. They develop a real-time DoS attack detection system that can detect TCP and UDP protocol specific attacks. They analyze the incoming network traffic based on windowing, i.e., splitting input traffic into traffic subsets. In order to obtain a reasonable estimate and control over the reaction time of the system for attacks and better models from larger training datasets, windowing is essential. In the training phase, the system takes stream information and traffic statistics as input and trains them using the Naive Bayes algorithm. They use 10-fold cross validation to estimate the accuracy for detection of attacks during their experiment.

Gaddam *et al.* [91] propose a novel method for supervised anomaly detection by cascading K-means clustering and ID3 decision tree learning. They combine K-means and ID3 learning using two rules: (i) the nearest neighbor rule and (ii) the nearest consensus rule. These two rules are used to obtain a final decision on classification of attack traffic. In this method, network traffic samples are first grouped into *N* clusters using K-means clustering and then ID3 decision tree algorithm is trained on individual cluster instances. The K-Means method ensures that each training instance is associated with only one cluster. However, if there are any subgroups or overlaps within a cluster, the ID3 decision tree trained on that cluster refines the decision boundaries by partitioning the instances with a set of if-then rules over the feature space.

TABLE VI
COMPARISON OF MACHINE LEARNING METHODS USED FOR DDOS DETECTION

Work	Method	Accuracy rate	False alarm rate	Effectiveness	Remarks
Vijayasarathy et al. [90]	Naive bayes	96.2-99.5%	0.4-0.5%	Prototype Independent	Tested with limited number of datasets
Seo et al. [96]	SVM/ TRA	89.8%	1.14%	Early detection of DDoS attacks	Uses multiple SVMs to reduce high false positive rate compared to single SVM
Gavrilis et al. [97]	RBF	100%	0%	100% DDoS detection rate with 0% false alarm	It is a passive monitoring system requiring very few computing resources
Shon et al. [98]	SVM/ GA	100%	0%	Shows fast processing time and better correction rate	The machine learning framework consist of GA for feature selection and SVM for packet classification
Ektefa et al. [99]	C4.5/ SVM	93.87/93.84%	1.28%	Average detection rate for C4.5 and SVM is 84.05% and 83.76%	C4.5 algorithm is better than SVM in detecting network intrusions and false alarm rate for the KDD CUP 99 dataset
Erman et al. [100]	Expectation Maximization	91%	2.03%	The time required to classify connections can be reduced with the unsupervised clustering technique	The unsupervised machine learning approach achieved better result

Su [92] proposes a method for DDoS attack detection using a weighted KNN classifier. In this method, a weight value is computed for each feature and an optimal subset of features is used for classification. They obtain 97.42% detection accuracy for known attacks and 78% accuracy for unknown attacks. Ramamoorthi *et al.* [93] propose an anomaly detection system to detect DDoS attack using Enhanced Support Vector Machines (ESVMs) with string kernels. This method can detect DDoS attacks on both network and transport layers. Classification accuracy for ESVM with string kernels is higher than one class SVMs, Binary SVMs and SVMs with string kernels.

A real time DDoS attack detection using fuzzy estimators is proposed by Shiaeles *et al.* [94]. In the first phase of this method, actual detection of DDoS attacks is performed and in the second phase, offending IP addresses are detected. A notable capability of this method is that it can not only detect DDoS attacks, but also identifies malicious IPs before the victim service suffers from exhaustion of resources due to the attack. In empirical evaluation, they find 80% success rate for the method during attack detection.

Erman *et al.* [95] propose offline/realtime traffic classification using semi-supervised learning. Due to continuous evolution of applications and their dynamic nature, classification of network traffic based on application type is difficult. Flow based classification is comparatively more efficient than packet content based classification due to the unpredictable nature of features; this motivated them to classify traffic by exploiting distinctive flow characteristics of applications when they communicate on a network. They propose an effective semi-supervised classification method to accommodate both known and unknown applications. The classifier uses only a few labeled and many unlabeled flows to train the instances. The significance of this classifier is that it considers two pragmatic classification issues, viz., longevity of classifiers and the need for retraining of classifiers. Experimental evaluation on empirical Internet traffic traces that span a 6-month period shows that: (1) a significantly high classification accuracy on both flow and byte (i.e., greater than 90%) can be achieved. A comparison of machine learning approaches in the context of DDoS detection is given in Table VI.

3) Botnet Attack Detection: Approaches and Methods: For about a decade, botnets attacks have been growing rapidly on the Internet and this has created many security problems for defense mechanisms used by network administrators. Many solutions have been proposed to detect botnet attacks based on attack behavior. People use traffic statistics [101], nature of communication protocols used [102], general analysis of network behavior [103], graphical representations of behaviors, actions in honypots, and collaborative feedback in large networks to detect botnet attacks. The majority of current DDoS attacks including mimicking attacks are performed by botnets, and it is possible to distinguish legitimate cyber behavior from botnets attacks using different detection methods [104]. In this section we provide a brief discussion on botnet detection, tracking and defense methods, their effectiveness and pros and cons.

Botnet detection methods are generally classified into two categories: analysis of passive traffic and capture of traffic using honynet [105]. A honeynet collects data using tools like honeysnap [106], ngrep [107], and tcpdump [108] and analyzes the collected data to identify bots. A honeynet is used to identify zombies or control hosts in a botnet. A subnet of a honeynet pretends to be compromised by Trojans and observes the behavior of attackers. Freiling *et al.* [109] propose a method to detect botnet-based DDoS attacks using honeypot and active responders to collect bot binaries. These binary executables are then run on the honeypot, allowing access to the IRC server. As a result, the honeynet is able to collect useful information to facilitate attack detection. A botnet detection mechanism is proposed by Choi *et al.* [110] that monitors DNS traffic generated by distributed bots. Karasaridis *et al.* [111] propose a mechanism to detect, track and characterize botnets on a wide-scale Tier-1 ISP network. They use a passive mechanism for network data analysis which is invisible to the botnets and gives less than 2% false alarm rate during botnet detection.

P2P botnet detection: P2P botnet attacks are very difficult to detect and mitigate due to decentralized attack generation. To handle the difficulties of P2P botnet detection, researchers have proposed a variety of solutions. For effective classification of P2P botnet traffic, relevant features can be extracted using correlation measures [112] or mutual information [2]. P2P protocol based botnet detection methods are classified into two categories, (i) detection based on feature code (DoPF) and (ii) detection based on network stream (DoNS) [113]. Performance of DoPF is lower than DoNS because DoNS detects botnet attacks on network and transport layers using a set of features of network data streams. The DoNS model consists of three modules: the first module detect the P2P nodes, the next module computes clustering of P2P nodes and finally the detection module detects the nodes for botnet attacks. The P2P node detection module collects network traffic streams and analyzes the streams for detection. The P2P node clustering module analyzes network streams of P2P-nodes using symmetry, quantity and frequency of data exchange between each pair of P2P nodes and discovers clusters using K-means algorithm. From the computed clusters, the detection module detects P2P botnets from similar suspicious actions of P2P nodes.

Huang *et al.* [18] propose a method using contact chains for botnet detection and tracing. In this method, contact information for peers with suspicious behaviors are traced and a contact chain is established to correlate contacts among the peers. If the length of the contact chain is more than a predefined threshold, the peers that form a contact chain are assumed to be part of a P2P botnet. The method can detect botnets quickly and can block the propagation of bots in a dynamic P2P environment. The main drawback of the method is that the identities of machines on a tracing chain can be faked by botmasters.

An automatic botnet detection system, proposed by Jelasity *et al.* [114], uses network monitoring with Traffic Dispersion Graphs (TDGs). In this method, botnets are detected by monitoring the flow traffic behavior of a P2P overlay network. Nagaraja *et al.* [115] also use structured overlay topologies to detect botnets. They devise techniques to localize botnet membership from unique communication patterns used for command and control. They propose an algorithm called BotGrep to identify botnet hosts and links within network traffic traces. It detects botnets using a graph search algorithm over a structure overlay network and separates the subgraphs that exhibit distinct topological patterns from each other or the rest of the graph. It is expected that the maxing rate for the subgraphs of P2P traffic is faster than that for subgraphs corresponding to the rest of the traffic. Though this method does not give high detection accuracy, it produces low false positive rate.

A novel botnet detection method is proposed by Zhang *et al.* [7] to identify stealthy P2P botnets even though malicious activities may not be observable. In this method, statistical fingerprints are derived to profile different types of P2P traffic and these profiles are used to discriminate the P2P botnet traffic from legitimate P2P traffic. Experimental results show that this method can detect stealthy P2P bots with a high detection rate and a low false positive rate.

IRC-based botnet detection: Although IRC-based communication is fairly old, due to its simple command and control mechanism it is still used by bots. To detect botnets based on the IRC protocol, many methods have been proposed. Lu and Ghorbani [116] propose an algorithm to detect and characterize botnets in a large enterprise WiFi network. First, they apply the K-means clustering algorithm on payload signatures to classify network traffic into different applications. The IRC applications are then analyzed using temporal-frequent characteristics of flows to discriminate malicious IRC channels created by bots from legitimate IRC traffic.

Ma *et al.* [117] detect botnets from characteristics of packet size sequences of TCP conversations between zombies and their C&C servers. An approximate periodicity known as quasi-periodicity is observed in IRC botnet conversations. They use a new data structure called Conversation Content Sequence (CCS) to describe packet sequence segments, reflecting quasi-periodicity of the IRC botnet traffic.

Mazzariello [118] proposes a technique to detect botnets using IRC user behavior. The method initially captures raw network traffic and analyzes traffic using descriptive parameters. A classifier is used to separate normal network traffic from botnet-related network traffic. Experimental results show high classification accuracy for this method, but further analysis is required to know whether the obtained results are biased due to the employed classifiers or the analyzed datasets.

Botnet detection-based on C&C: Many command and control channels are used to communicate messages among the nodes of a botnets using communication protocols such as IRC and HTTP. It is very difficult to detect botnet C&C traffic because in botnet communication normal protocols are used to generate traffic and traffic volume is low. Besides, the communication among bots may be encrypted. However, botnet communication traffic is likely to have repetitive characteristics or spatial-temporal correlations due to pre-programmed response activities to control commands. Lee *et al.* [119] propose a method to detect malicious HTTP botnets using degree of periodic repeatability. They compute a degree of repeatability for an individual user's activities and if this value is much lower than other users, mark the user as a bot.

DISCLOUSER is a large scale, wide area botnet detection system proposed by Bilge *et al.* [120]. The system detects botnet C&C servers based on NetFlow analysis using a supervised machine learning algorithm. A set of network traffic features is used by the method to reliably distinguish C&C channels from benign traffic using NetFlow records such as flow size, client access patterns and temporal behavior.

BotSniffer is another popular botnet command and control channel detection method proposed by Gu *et al.* [121]. The authors first compute spatial-temporal correlations and similarities among these botnet C&Cs and discriminate C&C traffic from normal traffic using heuristics. They also propose anomaly based detection algorithms to detect both IRC and HTTP based C&Cs. The BotSniffer system combines these anomaly detection systems and evaluated using real-world network traces by its authors. The system gives high detection accuracy in detecting botnet C&Cs with a low false positive rate.

Allison *et al.* [122] develop a method to prevent Short Message Service (SMS) flooding. The method stores called and calling party identification information in a database and performs a lookup operation in a database that stores identification of systems that have previously taken part in SMS flooding. If the lookup operation determines a match entry in the database, that matched entry information is used to detect a potential SMS message flooding attack. If the method detects an entry that initiates SMS flooding attack, it discards the next several consecutive messages coming from that entry.

Botnets such as Conficker, Kraken, and Torpig use DNS domain fluxing as C&C mechanism to control bots. Yadav *et al.* [123] develop a method to identify domain fluxes in DNS traffic by looking for patterns that exist in domain names that are generated algorithmically. The authors apply statistical measures to classify malicious domains (generated algorithmically) from DNS traffic. Many botnets use domain generation algorithms (DGAs) to build complex C&C infrastructures for DDoS attack generation. Schiavoni *et al.* [124] propose a method called Phoenix to identify the DGA-based botnets using IP-based features and find representatives of botnets from the groups of DGA-generated domains. The Phoenix method consists of three modules, viz., a discovery module, a detection module and a module for intelligence and insights. The discovery module identifies DGA-generated domains whereas the detection module detects the domains whose names are automatically generated. The Intelligence and Insights module aggregates the outcomes of the previous two modules and extracts meaningful information from the observed data.

B. IP Traceback

The traceback mechanism finds the true source of forged IP packets that was used in attack generation. Especially in the case of DDoS attacks, it is very common to send attack packets to the victim machine using spoofed IP addresses using zombies or reflectors. Many solutions for IP traceback have been proposed [125], [126]. These are broadly divided into two categories based on detection strategies. In the first category, routers send their identities to the destinations of certain packets, either by encoding the information with the rarely used bits of the IP header or by creating new raw packets. The main drawback of this mechanism is that they are focused only on DDoS attacks and therefore cannot handle attacks with smaller packets. The second category of solutions is centralized and depends on packet logging in a network. The main problems of this type of solution are high computational overhead and scalability issues.

IP traceback can be performed manually to find the original source of an attack to reduce the effect of the attack on the victim network. IP traceback is performed either from the victim-end or from intermediate routers to the original source-end. A hop by hop trace mechanism is used from router to router. Therefore, co-operation among networks is required to trace attack packets back to their true sources. Since manual traceback is tedious, many mechanisms have been proposed to automate this process. An effective traceback mechanism should have the following properties.

1) The original source of attack should be detectable with the help of a single packet.
2) IP traceback should incur low computational overhead.
3) Low level of ISP involvement is ideal.
4) No additional memory should be used in routers or switches.
5) High level of protection should be obtainable from traceback.
6) There should be low network overhead during traceback.
7) Correct traceback analysis should obtain low false positive rate.
8) The deployment of the traceback system should be limited.

C. IP Traceback in DDoS Prevention

Link testing schemes: In such a scheme, the victim tests each of its incoming links as a probable input link for DDoS traffic and contacts the upstream router which is closest to the victim. This router then interactively traces back to its upstream routers until it finds the source of any potential attack. This recursive procedure is performed on every upstream router to reach the original source. The main advantages of link test schemes are that they can reliably detect flooding attacks, the network overhead is low and the distribution is very efficient. However, the major drawback is that the scheme generates additional traffic and consumes large network resources. There are two types of link testing mechanisms: (i) input debugging and (ii) controlled flooding.

Input debugging: In this mechanism, packets are filtered on every router at some egress ports to determine from which ingress ports they have arrived. During input debugging, the victim must recognize an attack first and then construct an attack signature that describes common features contained in attack packets. The victim then communicates with the upstream router to install input debugging filters on egress ports. Such a filter reveals the associated input ports and hence the upstream routers that originate the traffic. This process is repeated recursively until the originating source is reached. This mechanism is efficient in finding the true attack source because of its distributed nature. But, it has many cons such as high management cost, high network/router overhead, need for significant amount of time during communication to upstreams routers in a large network and need for expert and skilled network operators, without whom traceback will be slow and impossible to complete.

Controlled flooding: A mechanism is proposed by Burch and Cheswick [127] to test links by flooding them with large bursts of network traffic and then observing the effect from attackers. This traceback mechanism is an automatic process that does not require any support from network operators. It floods each incoming link on the router closest to the victim using a pre-generated map of Internet topology containing a few selected hosts. Any packets (including the packets sent by attackers) traveling across the loaded links must have high packet dropping probability. As a consequence, the victim can infer attack links from changes in the rate packets arrive from attackers. This basic procedure is used recursively on the upstream routers until the source is reached. This is a very skillful, practical and effective traceback mechanism. The main drawbacks of this mechanism are that it has high management overhead, needs coordination among routers or switches or even IPSs, and requires skilled network administrators.

Packet marking schemes: Packet marking is one of the best ways to trace sources of flooding attacks. Routers mark forwarding packets either probabilistically or deterministically, with their own addresses. During an attack, the victim uses the marked information in the packet to traceback the attack source. Packet marking schemes are categorized into two classes, viz., probabilistic and deterministic.

The *Probabilistic Packet Marking (PPM)* approach was first proposed by Savage *et al.* [128]. It does not require apriori knowledge of the whole network for an attack tree. It can be used during an attack or even after an attack has occurred. In this scheme, the IP header uses only a single entry to store the marking information. Each router on the path from the source to the destination writes down the unique identifier in the entry in the packet header with some probability. By writing into the entry, routers overwrite any previous entry that was present there. The victim can reconstruct the path from the source to itself on receiving a large number of packets. The main advantages of this mechanism over other schemes are that it does not require any additional network traffic like ICMP traceback, router storage for logging or packet size increase. In this approach, each router performs an information injection event for every forwarding packet. To inject information into a packet, it uses the 16-bit identification field in the IP header. Out of 16 bits, it uses 5 bits for maintaining hop count information and remaining bits for the message that the router wants to send to the destination of the packet. If the message is too big, fragmentation is used to make it smaller in size with some bits to indicate the fragment offset and data fragment. During a DoS attack, a victim can reconstruct the message with the help of a hash function interleaved in the original message it received from the router.

In *Deterministic Packet Marking (DPM)*, each outgoing packet is marked by the router with its own unique identifier. This mechanism is similar to the IP record-route option and it uses the mark information during reconstruction of attack path at the victim. Savage *et al.* [128] calculate the optimal value for the marking probability to be $1/d$, where d is the length of the path. The randomize-and-link approach proposed by Goodrich *et al.* [129] is an improvement of the probabilistic packet marking scheme from security and practicality points of view. The core concept is that each router fragments its message M into several words and these words are included randomly in the b reusable bits together with a large checksum. Though the approach is efficient, it wastes b precious bits. The checksum codes significantly reduce the ability of an adversary to inject false messages that collide with legitimate ones. The main strength of this approach is that it can easily recognize 8-bit or longer fragment messages from hundreds of routers even when attackers inject packets to slow the approach. Moreover, this approach does not require any prior knowledge of the whole network.

Xiang *et al.* [130] propose an optimized version of DPM called Flexible DPM (FDPM) that provides a defense system with the ability to find real sources of attacking packets. Compared to link testing, packet logging, ICMP traceback, PPM and DPM, FDPM provides more flexible features to trace IP packets and gives better performance. In some situations, the method uses the type of service field of the IP header to store the mark information. It exploits two fields in the IP header, one is the fragment ID and other is the reverse flag. The sender of a packet assigns a value called the identifying value to the ID field that helps assemble all fragments of a datagram. Compared to DPM, FDPM is simpler and more flexible during path reconstruction. FDPM is very effective, especially in terms of low false positive rates as well as the small number of packets needed to reconstruct one source, the maximum number of sources that can be traced in one traceback process, and the maximum forwarding rate of traceback-enabled routers. Other pros of FDPM includes easy implementation, low processing cost, low bandwidth overhead, suitability for attacks other than (D)DoS, scalability and lack of inherent security flaws.

Alwis *et al.* [131] propose a network topology based packet marking approach known as TBPM. This approach is different in many ways from traditional packet marking approaches. It embeds network topology information in a data packet to be marked. The main problem of traditional packet marking methods is that they mark the identity of the edge router through which a packet enters a network. However, during flooding attacks the edge router may be unreachable to the node under attack. A node can defuse the attack close to its source with the help of information about the route that the packet has traversed through the network. This is practically possible even when the edge router is unreachable and therefore, this approach can restore functionality of the internal network in the presence of a DoS attack in the edge routers. Space efficiency in the form of constant marking field and processing efficiency in the form of minimum router support are two main advantages of this method. However, high false positive rate, high number of required packets, poor capability for packet tracing and inflexible marking rate that cannot adapt to the load of participating router are the major drawbacks of this method.

Packet logging: In this approach, routers store packet information so that they can trace an attack long after the attack has completed. A router may use data mining techniques on the packet logged data to determine the path that the packets traversed. Many variations of packet logging methods have been proposed by researchers. Snoeren *et al.* propose [126] a hash-based IP traceback that records the packet digest in a specific efficient data structure. This method needs a significant amount of memory space to store the logged information. To overcome this problem, Bloom [132] proposes a filter called the bloom filter to minimize storage overhead significantly. The main advantages of this method are: (i) storage of packet log information historically for future investigation (ii) easy for traceback, and (iii) excellent distribution capability. However, it requires high storage space to store historic data, high network overhead and high management overhead.

ICMP traceback messages: In this mechanism, the router generates ICMP traceback messages that include the content of a forwarded packet along with information about adjacent routers and sends it to the destination. When flooding like attacks occur, the victim uses these ICMP messages to construct attack graphs back to the attacker. The traceback message helps the victim find the original source of attack. However, this mechanism relies on an input debugging capability which is not enabled in many router architectures. As a result, it may be difficult to establish a connection between a participating router and a non-participating router. ICMP traceback is effective in terms of network overhead as it incurs low management cost. Moreover, it has strong distribution capability and effectively detects attack paths during flooding attacks. However, it generates high additional network traffic and creates many false ICMP messages. ICMP messages can be distinguished easily and hence may be filtered or rate limited differently from normal traffic. The main problems with this method are

TABLE VII
COMPARISON OF DDOS PREVENTION TECHNIQUES

Technique	Advantage	Disadvantage
IP Traceback	Identifies the true sources of attacks	Router overhead
Ingress Filtering	Prevents spoofed IP packets Combats against network abuse by means of packet trace	Needs global development Attacks with real IP addresses cannot prevent
Route based Packet Filtering	Works well with static routing	Problem when dynamic routing is used Needs wide implementation to be effective
Probabilistic Packet Marking	It is very simple and supports incremental deployment	Path reconstruction process requires high computational work
Deterministic Packet Marking	Simple to implement and less computational overhead	Enough packets must be collected to reconstruct the attack path
History based	Does not require cooperation of whole Internet community Gives priority to frequent packets in case of congestion or attack	Ineffective when the attacks come from real IP addresses Requires an offline database to keep track of IP addresses Depend on information collected
Capability based	Provides destination a way to control the traffic it desires Incremental deployment	Attacks against request packets can not prevented High computational complexity and space requirement
Secure Overlay Service (SOS)	Works well for communication of predefined source nodes	Solution has limited scope, e.g., not applicable to Web servers Require introduction of a new routing protocol that itself may be another security issue
SAVE	Filters malicious addressed packets Incremental deployment	During the transition period valid packets can be dropped

high computational overhead in a large network, detection of multiple attack paths, difficulty in IP traceback due to stateless nature of Internet routing and lack of source accountability in TCP/IP, and inefficient manual IP traceback.

IP traceback using entropy: Yu *et al.* [1] propose a novel DDoS traceback mechanism using entropy variations between normal and DDoS attack traffic which is memory nonintensive, robust against packet pollution, efficiently scalable, and does not depend on specific attack traffic patterns. In this mechanism, they use entropy variations to measure randomness of flows at a router within a given time interval. If the entropy value of a flow is greater than a user defined threshold, trace the IP via the upstream router.

In addition to the IP traceback mechanism, some other mechanisms such as Ingress/Egress filtering, rate control and pushback mechanisms are also used for DDoS prevention. A brief discussion of these methods are given here.

Ingress/egress filtering: An Ingress/Egress filtering mechanism is effective in preventing DDoS attacks because it makes attack generation difficult for attackers using spoofed IP addresses. IP spoofing is used to generate some attacks such as Smurf (ping flood), making it difficult to traceback the original source of attacks. A firewall that connects a network to the Internet has two types of interfaces, one connected to the internal network and other connected to the Internet. Using ingress filtering on the external interfaces, the firewall drops all packets with source IP addresses within its internal network because such packets are being clearly spoofed. If such packets are allowed into the network, the attacker can masquerade as a host within the same network. On the other hand, egress filtering is applied on the internal interface on packets that are heading out of the network and if the source address of a packet does not belong to its local network, it is dropped by the firewall. This stops an attacker from using hosts within that network as DDoS agents. These two solutions can prevent all attacks that are generated based on IP spoofing. Moreover, they can traceback the original source of attack as the attacking hosts are forced to use their true IP addresses. However, ingress/egress filtering cannot protect bandwidth based DDoS attacks.

Rate control: Rate limiting mechanisms may be used to prevent DDoS attacks based on pre-defined attack prevention criteria. This approach limits the rate of packet arrival from a source if the packet matches the criteria of DDoS attacks. The main advantage of this mechanism is that it does not harm legitimate flows at all. Furthermore, it does not incur extra overhead on the network and does not create a situation of denial of service attack by itself. If it is known that the attack detection mechanism can come up with many false positives, it is better to go for rate limiting rather than packet filtering.

Pushback mechanism: The pushback mechanism is used to provide tolerance for a network or a particular Website in the presence of DDoS attacks. After detecting a DDoS attack, a filtering mechanism is used to drop malicious packets. By filtering or blocking malicious packets locally on a router, we can protect the particular Website or the local networks, but the attacker can still achieve his goal by flooding network links. Thus the best way is to push the filter back to the attack source. The closer the filter is to the source, the more effective it is. Ioannidis and Bellovin [57] propose a pushback scheme that propagates the pushback signal in the network recursively. It uses control signals to describe the traffic aggregation which has caused network congestion. A general comparison among different DDoS prevention techniques is provided in Table VII. The comparison highlights the advantages as well as disadvantages of the approaches in prevention of DDoS attacks. Some possible important prevention approaches for DDoS attacks are mentioned below.

- Dynamically change the IP address of a victim to evade attackers.
- Link bandwidth maintenance between the edge router and victim machine.
- Use the core router/edge router to determine suspicious IP address block list at the network boundary to drop bogus IP traffic.
- Perform periodic checkpoint about malicious activities in a network.
- Verify protocol type IP packets and their arrival rates.

D. DDoS Tolerance: Approaches and Methods

The main objective of a tolerance mechanisms is to limit the damage caused by DDoS attacks even if it is difficult to differentiate between the behaviors of normal and malicious traffic. Tolerance mechanisms do not depend on a specific detection mechanism or may not even know that attacks are happening in a network or a system. A tolerance system always tries to provide services to legitimate users of a network under attack situations. Mishra *et al.* [133] classify intrusion tolerance and mitigation techniques into two classes, viz., fault tolerance and Quality of Services (QoS). Fault tolerance can be applied at three places, viz., (i) at hardware level (ii) at software level and (iii) at system level.

1) Fault Tolerance: Fault tolerance is a property that enables a system to continue operating properly in the presence of malfunction in the system. Fault tolerance provides survivability to a system to continue its services without being completely shut down. Fault tolerance is an important property for any Intrusion Detection System because there is no IDS that can detect all kinds of network attacks to provide complete security to a system. So, an IDS should provide mechanisms that can enable the system to tolerate network attacks so that the system can operate without being completely damaged. Faults in a system can occur either accidentally or may be malicious. Faults should be detected as soon as possible and fault tolerance should provide mechanisms to recover the system as quickly as possible. Some fault tolerant architectures for DDoS attacks defense are presented here.

System Fault Tolerance Agent (SFTA) [134]: SFTA is an intelligent agent based fault tolerance mechanism used for network intrusion detection systems. It consists of three types of agents, (i) System Sentinel Agent (SSA) (ii) System Fault Evaluation Agent (SFEA) and (iii) System Replication Agent (SRA). SSA is the main agent in the system that monitors the agents and hosts to detect if they are active. It also monitors unauthorized actions and malicious activities on the system. SFEA is responsible for analysis of information collected by SSA. It keeps knowledge about availability of resources (memory, disk space, etc.) in the hosts and enables instant recovery of agents with proper recovery actions. It can request alteration of the strategy for replication of a group of agents, migration or creation of new agents. Finally, SRA is responsible for agent management such as recovery of agents, replication of agents, and adding or removing an agent from a group.

SITAR [135]: It is a scalable intrusion tolerance architecture designed for distributed services. It provides a generic class of protection services to the network and protects a target system by means of redundancy and diversity. It also deals with external or unknown attacks, and zombies that lead to unpredictable behavior during attacks. The intrusion tolerant architecture enables building intrusion tolerant services out of the existing intrusion vulnerable server.

SCIT [136]: Self Cleaning Intrusion Tolerance (SCIT) is a recoverable intrusion tolerance system that uses a periodic recovery policy and simultaneously maintains service availability using redundant servers. It is composed of an SCIT controller and redundant servers as shown in Fig. 17. The SCIT server has four different states, viz., active, grace period, cleansing period and live spare period. In active state, the server is online and accepts requests from the outer world. When the server is in grace period, it does not communicate with the outer world and only processes tasks for the requests that were received during the active period. In the cleansing period, the server is offline and it recovers the system configuration files, service files, and so on. Finally, the server waits to be active in the live spare period.

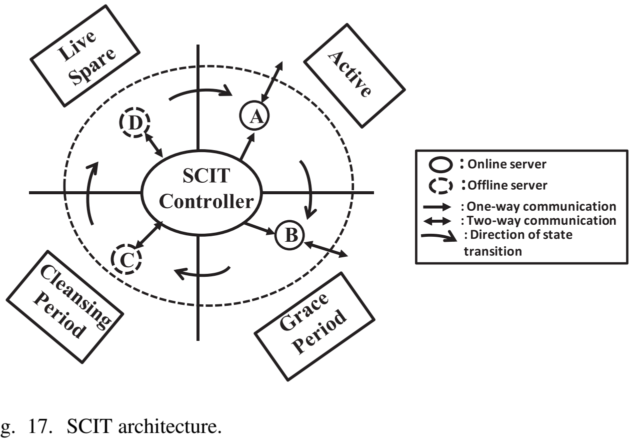


Fig. 17. SCIT architecture.

OASIS [137]: Organically Assured and Survivable Information System (OASIS) is a network defense system designed by DARPA to provide defense capabilities against sophisticated adversaries to allow sustained operation of mission critical functions in the face of known and future cyber attacks. The system consists of potentially vulnerable components and operates with low cost.

Fireflies [138]: Fireflies is a scalable protocol designed for intrusion tolerance network overlays. It provides current view of the alive nodes in a network as well as a pseudo-random mesh for communication. Fireflies considers three states of members: correct, stopped and Byzantine. Correct members execute protocols such as Gossip and Ping, stopped members do not execute protocols whereas Byzantine members are not related to protocols. Gossip is a simple group communication protocol where each member picks a random member from its view and exchanges state information. Pinging is used by a member to detect failures of other members.

MAFTIA [137]: Malicious- and Accidental-Fault Tolerance for Internet Applications (MAFTIA) investigates intrusion tolerance for constructing large-scale dependable distributed applications. The project has a comprehensive approach that handles both accidental and malicious faults. MAFTIA follows three main lines of actions: (i) an architectural framework and conceptual model for intrusion tolerance (ii) design mechanisms and protocols for fault tolerance and (iii) formal validation and assessment of intrusion tolerance.

2) QoS: Quality of Service is an important consideration for any intrusion tolerance system. An effective intrusion tolerance system should provide services without degradation of quality for certain types of applications and traffic. In a DDoS defense system, tolerance is essential because flooding attacks can exhaust the resources of a server in a very short time. The tolerance mechanism should be able to provide services to legitimate users without compromising quality in the presence of attacks. We discuss a few quality of service improvement techniques.

Integrated services: The integrated services approach uses well-defined network architecture to provide quality of service to a network. In this mechanism, each router uses special protocols like Resource Reservation Protocol (RSVP) and reserves resources to provide guaranteed services. It is a flow-based mechanism used to transmit video and sound to the receiver without interruption.

Differentiated services: Differentiated Services architecture specifies a simple, scalable and coarse-grained mechanism for managing network traffic that provides quality of service in a network. It operates on the principle of traffic classification and provides low latency on transmission of voice and video traffic.

Real-time service: Many applications require real-time data services and provide quality of service for both transaction timeliness and data freshness in real-time data analysis. The Real-time Application QoS Monitoring Framework (RAQMON) [139] provides quality of service statistics in real time by end-devices and applications. Many real-time applications can report application-level QoS statistics in real time using the RAQMON Framework.

Class-based queuing: Class-based queuing mechanism is used by a network scheduler that allows traffic to share bandwidth equally among different groups of users. A group is based on parameters such as priority, interface, or originating program. This mechanism can provide effective quality of service against starvation by assigning proper bandwidth to each queue.

Proactive server roaming: In this technique, servers change their locations frequently to defend against undetectable and unpredictable attacks. Only legitimate users can track the server during communication. Proactive roaming servers are used to mitigate DDoS attacks.

E. Discussion

In this section, we have discussed a significant number of detection, prevention and tolerance approaches for DDoS attacks in source-end, victim-end and in intermediate locations. It is crucial to detect a DDoS attack near the original attack source and a victim-end defense mechanism cannot protect high rate DDoS attacks in real time. Moreover, a system is always vulnerable to zero-day attacks because such attacks may evade the detection mechanism of the system. Hence, a collaborative intrusion tolerance mechanism is ideal to protect a system from unknown attacks.

IX. DDoS and Botnet in Emerging Contexts

DDoS attacks are a growing security threat to cloud servers and software defined networking. To defend against DDoS attacks, Huawei is the first vendor to apply Big Data technology to the detection and prevention of covert DDoS attacks disguised as normal access requests. A brief discussion of DDoS attacks in the context of cloud computing, Big Data and software defined networking is given here.

A. DDoS and Botnet in Cloud Computing

Security in the cloud is a major concern for cloud service providers. In cloud computing, resources are shared among millions of users so that resources and services can be access as needed by users. Due to the shared architecture of resources, DDoS attacks may have drastic impact on cloud computing compared to single tenant architectures. Confidentiality, integrity and availability are the three major requirements cloud computing environments must provide and DDoS attack happens to be a major threat to availability [140]. A hacker may run a malicious application to generate DDoS attack against the cloud itself or against another user in the cloud. Flooding DDoS, Web-based DDoS, traditional botnet DDoS and mobile botnet DDoS attacks can be generated in cloud servers to disrupt services provided by them. For example, a Zeus bot (Zbot) variant was spotted taking advantage of Amazon EC2's cloud-based services for its C&C functionalities. A botnet can generate DDoS attacks in the cloud either by hosting the C&C on the cloud or by creating bots (i.e, botcloud) on the cloud [141]. Securing cloud servers from DDoS attacks as well other vulnerabilities is a must to provide reliable services to consumers of cloud services. Cloud service providers use various security mechanisms to keep their servers secure from DDoS attacks.

Zhao *et al.* [142] discuss how Google's cloud-based C2DM service for the Android platform can potentially be attacked by a cloud-based push-style mobile botnet where push notification service is used as a C&C channel. This attack mechanism uses what is called a C2DM botnet. In this botnet, there is no direct communication between the botmaster and the bots, but C2DM services are exploited as a relay. Using C2DM services the botmaster's commands to the bots and botnet traffic can be hidden in the C2DM traffic of other application.

B. DDoS and Botnet in Big Data Analytics

Big Data analytics encompass the processes used to collect, organize and analyze large volumes of data to uncover hidden patterns, unknown correlations and other useful information used in decision making. On February 24, 2014, Huawei, a global Information and Communications Technology (ICT) solutions provider announced the launch of its next-generation anti-DDoS solution called AntiDDoS8000 that can defend several hundred Gigabits-per-second DDoS threats using Big Data analytics and other mechanisms.[3] To provide a high level of security to organizations, up-to-date defenders need to use Big Data analytics combined with anti-DDoS mechanisms. In general, Big Data analytics methods are very useful in analyzing huge amounts of network traffic generated by sophisticated botnets. A Big Data analytics framework using random forests is proposed by Singh *et al.* [143] to detect peer-to-peer botnets. The paper discusses the implementation of a scalable intrusion detection system to handle vary large network traces using Big Data technologies.

[3] http://enterprise.huawei.com/ilink/enenterprise/news-event/news/news-list/HW_328116

C. DDoS and Botnet in Software Defined Networking (SDN)

DDoS attacks can also occur in SDNs causing the SDN controller to become unreachable. In botnet based DDoS attacks, a large number of spoofed packets are sent to a host in a network and due to their spoofed IP addresses, the switch of an SDN cannot find match for the incoming packets. As a result, these incoming packets are forwarded to the SDN controller exhausting its resources. Finally, the SDN controller become unreachable for newly arrive incoming packets, thus breaking the SDN architecture.

An SDN can be adapted to provide effective ways to detect and prevent DDoS attacks by analyzing network traffic patterns [144]. Lim *et al.* [145] propose an SDN-oriented DDoS blocking scheme for botnet-based attacks using a standard OpenFlow interface. Brocade Communications Systems provides efficient, scalable and real-time SDN analytics for mitigating DDoS attacks to protect cloud data centers from various security threats.[4]

X. Concluding Remarks

We start this paper with a discussion of DDoS attacks and present a classification of DDoS attacks and characteristics of each class of DDoS attacks. We then introduce botnet technology, the primary facilitator of modern DDoS attacks, and recent trends in the launching of various types of DDoS attacks using botnets. A discussion of mobile botnets and traditional PC-based botnets is given, followed by brief comparison between the two. We provide a detailed discussion of botnet-based DDoS defense approaches and methods. Though, in the past two decades, a good number of defense solutions have been introduced to counter DDoS attacks with increasing sophistication, still there are several important issues and research challenges which are open and yet to be addressed. Some of the prominent issues and research challenges are reported next to push the envelop further.

- Existing methods have been designed to be effective in detecting either low-rate or high-rate DDoS attacks, but usually not for both. So, developing a robust method that can detect both these types of attacks in real time remains a problem that needs attention.
- The performance of most methods is dependent on network conditions and their performance is also highly influenced by multiple user parameters. Hence, developing a defense solution free from these limitations as far as possible should be an important research initiative.
- Due to lack of unbiased evaluation frameworks, including benchmark datasets, it is difficult to properly evaluate the performance of the methods being developed. So, creating an unbiased framework for appropriate evaluation of a defense solution happens to be an important issue for investigation.
- Developing a generic solution, if possible, to defend all or most types of DDoS attacks, irrespective of protocol used is yet another research challenge.
- Most prevention methods are effective only for known attacks. Although efforts have been made to make many approaches adaptive and dynamic, it is still elusive to find a prevention mechanism that can protect network resources from unknown attacks in near real-time without such compromising for services towards for users.
- Developing a traceback mechanism that ensures:

  (i) integrates multiple traceback mechanism with customization support, and
  (ii) is cost effective, by allowing reuse of information extracted during detection, and
  (iii) makes no compromise of QoS

  remains an unsolved task.

[4] http://www.brocade.com/downloads/documents/flyers/brocade-interop-sdn-ddos-mitigation-flyer.pdf

References

[1] C.-H. Yu, K. Doppler, C. B. Ribeiro, and O. Tirkkonen, "Resource sharing optimization for device-to-device communication underlaying cellular networks," *IEEE Trans. Wireless Commun.*, vol. 10, no. 8, pp. 2752–2763, Aug. 2011.

[2] N. Hoque, D. Bhattacharyya, and J. Kalita, "MIFS-ND: A mutual information-based feature selection method," *Expert Syst. Appl.*, vol. 41, no. 14, pp. 6371–6385, Oct. 2014.

[3] D. K. Bhattacharyya and J. K. Kalita, *Network Anomaly Detection: A Machine Learning Perspective*. Boca Raton, FL, USA: CRC Press, 2013.

[4] M. Feily, A. Shahrestani, and S. Ramadass, "A survey of botnet and botnet detection," in *Proc. IEEE 3rd Int. Conf. Emerging Security Inf., Syst. Technol. SECURWARE*, 2009, pp. 268–273.

[5] M. Bailey, E. Cooke, F. Jahanian, Y. Xu, and M. Karir, "A survey of botnet technology and defenses," in *Proc. IEEE CATCH*, 2009, pp. 299–304.

[6] H. R. Zeidanloo, M. J. Z. Shooshtari, P. V. Amoli, M. Safari, and M. Zamani, "A taxonomy of botnet detection techniques," in *Proc. IEEE 3rd ICCSIT*, 2010, vol. 2, pp. 158–162.

[7] L. Zhang, S. Yu, D. Wu, and P. Watters, "A survey on latest botnet attack and defense," in *Proc. IEEE 10th Int. Conf. TrustCom*, 2011, pp. 53–60.

[8] R. A. Rodríguez-Gómez, G. Macíó-Fernóndez, and P. García-Teodoro, "Survey and taxonomy of botnet research through life-cycle," *ACM CSUR*, vol. 45, no. 4, p. 45, Aug. 2013.

[9] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles, "Botnets: A survey," *Comput. Netw.*, vol. 57, no. 2, pp. 378–403, Feb. 2013.

[10] V. Igure and R. Williams, "Taxonomies of attacks and vulnerabilities in computer systems," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 1, pp. 6–19, 1st Quart. 2008.

[11] J. Yuan and K. Mills, "Monitoring the macroscopic effect of DDoS flooding attacks," *IEEE Trans. Depend. Secure Comput.*, vol. 2, no. 4, pp. 324–335, Oct.–Dec. 2005.

[12] M. Fakrul Alam, "Application layer ddos a practical approach & mitigation techniques," bdHUB Limit., Dhaka, Bangladesh, 2014.

[13] E. Alomari, S. Manickam, B. Gupta, S. Karuppayah, and R. Alfaris, "Botnet-based distributed denial of service (DDoS) attacks on web servers: Classification and art," *Int. J. Comput. Appl.*, vol. 49, no. 7, pp. 24–32, Jul. 2012.

[14] M. Sauter, "LOIC will tear us apart the impact of tool design and media portrayals in the success of activist DDoS attacks," *Amer. Behavioral Sci.*, vol. 57, no. 7, pp. 983–1007, 2013.

[15] G. Ollmann, "Botnet communication topologies," *Retrieved Sep.*, vol. 30, pp. 1–7, 2009.

[16] T. Li *et al.*, "ZHT: A light-weight reliable persistent dynamic scalable zero-hop distributed hash table," in *Proc IEEE IPDPS*, 2013, pp. 775–787.

[17] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proc. ACM*, 2001, vol. 31, no. 4, pp. 161–172.

[18] H. Huang, Y. Zheng, H. Chen, and R. Wang, "Pchord: A distributed hash table for P2P network," *Front. Electr. Electron. Eng. China*, vol. 5, no. 1, pp. 49–58, 2010.

[19] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware 2001*. Berlin, Germany: Springer-Verlag, 2001, pp. 329–350.

[20] B. Y. Zhao *et al.*, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 1, pp. 41–53, Jan. 2004.

[21] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Peer-to-Peer Systems*. Berlin, Germany: Springer-Verlag, 2002, pp. 53–65.

[22] B. Beverly Yang and H. Garcia-Molina, "Designing a super-peer network," in *Proc. IEEE 19th Int. Conf. Data Eng.*, 2003, pp. 49–60.

[23] T. Micro, "Worm agobot," 2004. [Online]. Available: http://about-threats.trendmicro.com/ArchiveMalware.aspx?language=us&name=WORMAGOBOT

[24] T. Microhgj, "Worm sdbot," 2003. [Online]. Available: http://about-threats.trendmicro.com/ArchiveMalware.aspx?language=us&name=WORMSDBOT

[25] rbot, "The Ruby IRC bot." [Online]. Available: http://ruby-rbot.org/

[26] C. Li, W. Jiang, and X. Zou, "Botnet: Survey and case study," in *Proc. IEEE 4th Int. Conf. ICICIC*, 2009, pp. 1184–1187.

[27] "Experts bicker over conficker numbers," 2001. [Online]. Available: http://news.techworld.com

[28] J. Hruska, "New mega-D menace muscles storm worm aside," 2008. [Online]. Available: http://arstechnica.com

[29] G. Keizer, "Top botnets control 1M hijacked computers," 2008. [Online]. Available: http://www.computerworld.com

[30] B. Stone-Gross *et al.*, "Analysis of a botnet takeover," *IEEE Security Privacy*, vol. 9, no. 1, pp. 64–72, Jan./Feb. 2011.

[31] S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich, "Analysis of the storm and nugache trojans: P2P is here," *USENIX; login*, vol. 32, no. 6, pp. 18–27, 2007.

[32] J. P. John, A. Moshchuk, S. D. Gribble, and A. Krishnamurthy, "Studying spamming botnets using botlab" in *Proc. NSDI*, 2009, vol. 9, pp. 291–306.

[33] McAfee, "Cutall" 2015. [Online]. Available: http://vil.nai.com/vil/content/v_144515.htm

[34] G. Macesanu, T. Codas, C. Suliman, and B. Tarnauca, "Development of GTBoT, a high performance and modular indoor robot," in *Proc. IEEE IEEE Int. Conf. AQTR*, 2010, vol. 1, pp. 1–6.

[35] H. R. Zeidanloo and A. A. Manaf, "Botnet command and control mechanisms," in *Proc. IEEE 2nd Int. Conf. ICCEE*, 2009, vol. 1, pp. 564–568.

[36] T.-F. Yen and M. K. Reiter, "Traffic aggregation for malware detection," in *Detection of Intrusions and Malware, and Vulnerability Assessment*. Berlin, Germany: Springer-Verlag, 2008, pp. 207–227.

[37] P. Bacher, T. Holz, M. Kotter, and G. Wicherski, "Know your enemy: Tracking botnets," The Honeynets Project, Ann Arbor, MI, USA, 2005.

[38] R. Borgaonkar, "An analysis of the asprox botnet," in *Proc. IEEE 4th Int. Conf. SECURWARE*, 2010, pp. 148–153.

[39] M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: A literature survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2091–2121, 4th Quart. 2013.

[40] K. Higgins, "New massive botnet twice the size of storm," in *Retrieved May*. San Francisco, CA, USA: DarkReading, vol. 13, 2008.

[41] L. Trolle Borup, "Peer-to-peer botnets: A case study on waledac," Ph.D. dissertation, Techn. Univ. Denmark, Denmark, DTU, DK-2800 Kgs. Lyngby, 2009.

[42] J. Stewart, "Spam botnets to watch in 2009," SecureWorks, Atlanta, GA, USA, 2009. [Online]. Available: http://www.secureworks.com/research/threats/botnets2009

[43] M. Intelligence, "Annual security report," Symantec Corp., Atlanta, GA, USA, 2010.

[44] S. Greengard, "The war against botnets," *Commun. ACM*, vol. 55, no. 2, pp. 16–18, Feb. 2012.

[45] "Eggheads.org-eggdrop development," 1993. [Online]. Available: http://eggheads.org/

[46] W. Shuai, C. Xiang, L. Peng, and L. Dan, "S-URL flux: A novel c&c protocol for mobile botnets," in *Trustworthy Computing and Services*. Berlin, Germany: Springer-Verlag, 2013, pp. 412–419.

[47] M. La Polla, F. Martinelli, and D. Sgandurra, "A survey on security for mobile devices," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 446–471, 1st Quart. 2013.

[48] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild," in *Proc. 1st ACM Workshop Security Privacy Smartphones Mobile Devices*, 2011, pp. 3–14.

[49] C. Xiang, F. Binxing, Y. Lihua, L. Xiaoyi, and Z. Tianning, "Andbot: Towards advanced mobile botnets," in *Proc. 4th USENIX Conf. Large-Scale Exploits Emergent Threats*, USENIX Association, 2011, pp. 11–11.

[50] M. Eslahi, R. Salleh, and N. B. Anuar, "Mobots: A new generation of botnets on mobile devices and networks," in *Proc. IEEE ISCAIE*, 2012, pp. 262–266.

[51] H. Pieterse and M. S. Olivier, "Android botnets on the rise: Trends and characteristics," in *Proc. IEEE ISSA*, 2012, pp. 1–5.

[52] F. Fiu, "MDK: The largest mobile botnet in china," 2013. [Online]. Available: http://www.symantec.com/connect/blogs/mdk-largest-mobile-botnet-china

[53] A. Freed, "Misosms malware sends your text messages to attackers," 2013. [Online]. Available: http://www.tripwire.com/state-of-security/top-security-\ignorespacesstories/misosms-malware-sends-text-messages-china/

[54] Y. Zeng, K. G. Shin, and X. Hu, "Design of SMS commanded-and-controlled and P2P-structured mobile botnets," in *Proc. 5th ACM Conf. Security Privacy Wireless Mobile Netw.*, 2012, pp. 137–148.

[55] M. Robinson, J. Mirkovic, M. Schnaider, S. Michel, and P. Reiher, "Challenges and principles of DDoS defense," in *Proc. ACM SIGCOMM*, 2003, pp. 1–8.

[56] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2046–2069, 4th Quart. 2013.

[57] J. Ioannidis and S. M. Bellovin, "Implementing pushback: Router-based defense against DDoS attacks," in *Proc. Netw. Distrib. Syst. Security Symp.*, 2002, pp. 1–12.

[58] J. Mirkovic and P. Reiher, "D-WARD: A source-end defense against flooding denial-of-service attacks," *IEEE Trans. Depend. Secure Comput.*, vol. 2, no. 3, pp. 216–232, Jul.–Sep. 2005.

[59] K. Park and H. Lee, "On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, 2001, vol. 31, no. 4, pp. 15–26.

[60] A. T. Mizrak, S. Savage, and K. Marzullo, "Detecting compromised routers via packet forwarding behavior," *IEEE Netw.*, vol. 22, no. 2, pp. 34–39, Mar./Apr. 2008.

[61] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Mació-Fernóndez, and E. Vózquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Comput. Security*, vol. 28, no. 1, pp. 18–28, Feb./Mar. 2009.

[62] M. Thottan and C. Ji, "Anomaly detection in IP networks," *IEEE Trans. Signal Process.*, vol. 51, no. 8, pp. 2191–2204, Aug. 2003.

[63] W. Lee and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM TiSSEC*, vol. 3, no. 4, pp. 227–261, Nov. 2000.

[64] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz, "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks," *Expert Syst. Appl.*, vol. 29, no. 4, pp. 713–722, Nov. 2005.

[65] P. Li, M. Salour, and X. Su, "A survey of internet worm detection and containment," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 1, pp. 20–35, 1st Quart. 2008.

[66] A. Sperotto *et al.*, "An overview of IP flow-based intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 3, pp. 343–356, 3rd Quart. 2010.

[67] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336, 1st Quart. 2014.

[68] K. Pelechrinis, M. Iliofotou, and S. V. Krishnamurthy, "Denial of service attacks in wireless networks: The case of jammers," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 2, pp. 245–257, 2nd Quart. 2011.

[69] M. H. Bhuyan, H. J. Kashyap, D. K. Bhattacharyya, and J. K. Kalita, "Detecting distributed denial of service attacks: Methods, tools and future directions," *The Comput. J.*, vol. 57, no. 4, pp. 537–556, 2013.

[70] H. Chen, Y. Chen, and D. H. Summerville, "A survey on the application of FPGAS for network infrastructure security," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 4, pp. 541–561, 4th Quart. 2011.

[71] L. Li, J. Zhou, and N. Xiao, "DDoS attack detection algorithms based on entropy computing," in *Information and Communications Security*. Berlin, Germany: Springer-Verlag, 2007, pp. 452–466.

[72] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to DDoS attack detection and response," in *Proc. IEEE DARPA Inf. Surv. Conf. Expo.*, 2003, vol. 1, pp. 303–314.

[73] Z. Jian-Qi, F. Feng, Y. Ke-xin, and L. Yan-Heng, "Dynamic entropy based DoS attack detection method," *Comput. Electr. Eng.*, vol. 39, no. 7, pp. 2243–2251, Oct. 2013.

[74] S. Yu, W. Zhou, and R. Doss, "Information theory based detection against network behavior mimicking ddos attacks," *IEEE Commun. Lett.*, vol. 12, no. 4, pp. 318–321, Apr. 2008.

[75] S. Jin and D. S. Yeung, "A covariance analysis model for DDoS attack detection," in *Proc. IEEE Int. Conf. Commun.*, 2004, vol. 4, pp. 1882–1886.

[76] S. Yu *et al.*, "Discriminating DDoS attacks from flash crowds using flow correlation coefficient," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 6, pp. 1073–1080, Jun. 2012.

[77] X. Yi and Y. Shunzheng, "Monitoring the application-layer DDoS attacks for popular websites," *IEEE Trans. Netw.*, vol. 17, no. 1, pp. 15–25, Feb. 2009.

[78] Y. Xie and S.-Z. Yu, "A large-scale hidden semi-markov model for anomaly detection on user browsing behaviors," *IEEE Trans. Netw.*, vol. 17, no. 1, pp. 54–65, Feb. 2009.

[79] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source," in *Proc. 10th IEEE Int. Conf. Netw. Protocols*, 2002, pp. 312–321.

[80] A. Akella, A. Bharambe, M. Reiter, and S. Seshan, "Detecting DDoS attacks on ISP networks," in *Proc. 22nd ACM SIGMOD Workshop Manage. Process. Data Streams*, Citeseer, 2003, pp. 1–3.

[81] T. Peng, C. Leckie, and K. Ramamohanarao, "Proactively detecting distributed denial of service attacks using source IP address monitoring," in *NETWORKING 2004. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications*. Berlin, Germany: Springer-Verlag, 2004, pp. 771–782.

[82] Y. Chen, K. Hwang, and W.-S. Ku, "Collaborative detection of DDoS attacks over multiple network domains," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 12, pp. 1649–1662, Dec. 2007.

[83] G. Öke and G. Loukas, "A denial of service detector based on maximum likelihood detection and the random neural network," *The Comput. J.*, vol. 50, no. 6, pp. 717–727, 2007.

[84] C.-L. Chen, "A new detection method for distributed denial-of-service attack traffic based on statistical test," *J. UCS*, vol. 15, no. 2, pp. 488–504, 2009.

[85] S. Yu, W. Zhou, R. Doss, and W. Jia, "Traceback of DDoS attacks using entropy variations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 3, pp. 412–425, Mar. 2011.

[86] M. S. Fallah and N. Kahani, "TDPF: A traceback-based distributed packet filter to mitigate spoofed DDoS attacks," *Security Commun. Netw.*, vol. 7, no. 2, pp. 245–264, 2013.

[87] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, vol. 1, 2006.

[88] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 4, pp. 56–76, 4th Quart. 2008.

[89] T. Shon and J. Moon, "A hybrid machine learning approach to network anomaly detection," *Inf. Sci.*, vol. 177, no. 18, pp. 3799–3821, Sep. 2007.

[90] R. Vijayasarathy, S. V. Raghavan, and B. Ravindran, "A system approach to network modeling for DDoS detection using a naìve bayesian classifier," in *Proc. IEEE 3rd Int. Conf. COMSNETS*, 2011, pp. 1–10.

[91] S. R. Gaddam, V. V. Phoha, and K. S. Balagani, "K-means+ID3: A novel method for supervised anomaly detection by cascading K-means clustering and ID3 decision tree learning methods," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 3, pp. 345–354, Mar. 2007.

[92] M.-Y. Su, "Real-time anomaly detection systems for denial-of-service attacks by weighted k-nearest-neighbor classifiers," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 3492–3498, Apr. 2011.

[93] A. Ramamoorthi, T. Subbulakshmi, and S. M. Shalinie, "Real time detection and classification of DDoS attacks using enhanced SVM with string kernels," in *Proc. IEEE ICRTIT*, 2011, pp. 91–96.

[94] S. N. Shiaeles, V. Katos, A. S. Karakos, and B. K. Papadopoulos, "Real time DDoS detection using fuzzy estimators," *Comput. Security*, vol. 31, no. 6, pp. 782–790, Sep. 2012.

[95] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Offline/realtime traffic classification using semi-supervised learning," *Perform. Eval.*, vol. 64, no. 9, pp. 1194–1213, Oct. 2007.

[96] J. Seo, C. Lee, T. Shon, K.-H. Cho, and J. Moon, "A new DDoS detection model using multiple SVMs and TRA," in *Embedded and Ubiquitous Computing-EUC 2005 Workshops*. Berlin, Germany: Springer-Verlag, 2005, pp. 976–985.

[97] D. Gavrilis and E. Dermatas, "Real-time detection of distributed denial-of-service attacks using RBF networks and statistical features," *Comput. Netw.*, vol. 48, no. 2, pp. 235–245, Jun. 2005.

[98] T. Shon, Y. Kim, C. Lee, and J. Moon, "A machine learning framework for network anomaly detection using SVM and GA," in *Proc. IEEE 6th Annu. SMC IAW*, 2005, pp. 176–183.

[99] M. Ektefa, S. Memar, F. Sidi, and L. S. Affendey, "Intrusion detection using data mining techniques," in *Proc. IEEE Int. Conf. Inf. Retrieval Knowl. Manage CAMP*, 2010, pp. 200–203.

[100] J. Erman, A. Mahanti, and M. Arlitt, "QRP05-4: Internet traffic identification using machine learning," in *Proc. IEEE GLOBECOM*, 2006, pp. 1–6.

[101] L. E. Menten, A. Chen, and D. Stiliadis, "Nobot: Embedded malware detection for endpoint devices," *Bell Labs Tech. J.*, vol. 16, no. 1, pp. 155–170, Jun. 2011.

[102] J. Goebel and T. Holz, "Rishi: Identify bot contaminated hosts by IRC nickname evaluation," in *Proc 1st Conf. 1st Workshop Hot Topics Understand. Botnets*, Cambridge, MA, USA, 2007, pp. 8–8.

[103] W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas, "Botnet detection based on network behavior," in *Botnet Detection*. Berlin, Germany: Springer-Verlag, 2008, pp. 1–24.

[104] S. Yu, S. Guo, and I. Stojmenovic, "Can we beat legitimate cyber behavior mimicking attacks from botnets?" in *Proc. IEEE INFOCOM*, 2012, pp. 2851–2855.

[105] Z. Zhu *et al.*, "Botnet research survey," in *Proc. IEEE 32nd Annu. Int. COMPSAC*, 2008, pp. 967–972.

[106] D. Watson and J. Riden, "The Honeynet Project: Data collection tools, infrastructure, archives and analysis," in *Proc. IEEE WOMBAT Workshop Inf. Security Threats Data Collect. Sharing*, 2008, pp. 24–30.

[107] J. Ritter, "ngrep-network grep." [Online]. Available: http://packetfactory.openwall.net/projects/ngrep/index.html

[108] V. Jacobson, C. Leres, and S. McCanne, "The tcpdump manual page," Lawrence Berkeley Lab., Berkeley, CA, USA, 1989.

[109] E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: Understanding, detecting, and disrupting botnets," in *Proc. USENIX SRUTI Workshop*, 2005, vol. 39, pp. 44–59.

[110] H. Choi, H. Lee, H. Lee, and H. Kim, "Botnet detection by monitoring group activities in DNS traffic," in *Proc. IEEE 7th Int. Conf. CIT*, 2007, pp. 715–720.

[111] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale botnet detection and characterization," in *Proc. 1st Conf. 1st Workshop Hot Topics Understand. Botnets*, Cambridge, MA, USA, 2007, vol. 7, p. 7.

[112] P. Narang, J. M. Reddy, and C. Hota, "Feature selection for detection of peer-to-peer botnet traffic," in *Proc. 6th ACM India Comput. Conv.*, 2013, p. 16.

[113] D. Liu, Y. Li, Y. Hu, and Z. Liang, "A P2P-botnet detection model and algorithms based on network streams analysis," in *Proc. IEEE Int. Conf. FITME*, 2010, vol. 1, pp. 55–58.

[114] M. Jelasity and V. Bilicki, "Towards automated detection of peer-to-peer botnets: On the limits of local approaches," in *Proc. USENIX Workshop LEET*, 2009, p. 3.

[115] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov, "Botgrep: Finding P2P bots with structured graph analysis," in *Proc. USENIX Security Symp.*, 2010, pp. 95–110.

[116] W. Lu and A. A. Ghorbani, "Botnets detection based on IRC-community," in *Proc. IEEE GLOBECOM*, 2008, pp. 1–5.

[117] X. Ma *et al.*, "A novel IRC botnet detection method based on packet size sequence," in *Proc. IEEE ICC*, 2010, pp. 1–5.

[118] C. Mazzariello, "IRC traffic analysis for botnet detection," in *Proc. IEEE 4th Int. Conf. ISIAS*, 2008, pp. 318–323.

[119] J.-S. Lee, H. Jeong, J.-H. Park, M. Kim, and B.-N. Noh, "The activity analysis of malicious HTTP-based botnets using degree of periodic repeatability," in *Proc. IEEE Int. Conf. SECTECH*, 2008, pp. 83–86.

[120] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: Detecting botnet command and control servers through large-scale netflow analysis," in *Proc. 28th ACM Annu. Comput. Security Appl. Conf.*, 2012, pp. 129–138.

[121] G. Gu, J. Zhang, and W. Lee, "Botsniffer: Detecting botnet command and control channels in network traffic," College Comput., Georgia Inst. Technol., Atlanta, GA, USA, 2008.

[122] R. L. Allison and P. J. Marsico, " *Methods and systems for preventing short message service (SMS) message flooding*," U.S. Patent 7 145 875, Dec. 5, 2006.

[123] S. Yadav, A. K. K. Reddy, A. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 48–61.

[124] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: DGA-based botnet tracking and intelligence," in *Detection of Intrusions and Malware, and Vulnerability Assessment*. Berlin, Germany: Springer-Verlag, 2014, pp. 192–211.

[125] A. Castelucio, A. Ziviani, and R. M. Salles, "An as-level overlay network for IP traceback," *IEEE Netw.*, vol. 23, no. 1, pp. 36–41, Jan./Feb. 2009.

[126] A. C. Snoeren *et al.*, "Single-packet IP traceback," *IEEE ToN*, vol. 10, no. 6, pp. 721–734, Dec. 2002.

[127] H. Burch and B. Cheswick, "Tracing anonymous packets to their approximate source," in *LISA*, 2000, pp. 319–327.

[128] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical network support for IP traceback," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, pp. 295–306, Oct. 2000.

[129] M. T. Goodrich, "Efficient packet marking for large-scale IP traceback," in *Proc. 9th ACM Conf. Comput. Commun. Security*, 2002, pp. 117–126.

[130] Y. Xiang, W. Zhou, and M. Guo, "Flexible deterministic packet marking: An IP traceback system to find the real source of attacks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 4, pp. 567–580, Apr. 2009.

[131] H. A. Alwis, R. C. Doss, P. S. Hewage, and M. U. Chowdhury, "Topology based packet marking for IP traceback," in *Proc. ATNAC*, Citeseer, 2006, pp. 224–228.

[132] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[133] A. Mishra, B. Gupta, and R. C. Joshi, "A comparative study of distributed denial of service attacks, intrusion tolerance and mitigation techniques," in *Proc. IEEE EISIC*, 2011, pp. 286–289.

[134] L. Siqueira and Z. Abdelouahab, "A fault tolerance mechanism for network intrusion detection system based on intelligent agents (NIDIA)," in *Proc. IEEE 4th Workshop SEUS 2nd Int. WCCIA*, 2006, pp. 1–6.

[135] F. Wang *et al.*, "SITAR: A scalable intrusion-tolerant architecture for distributed services," in *Proc. Workshop Inf. Assur. Security*, 2003, vol. 1, p. 1100.

[136] D. Arsenault, A. Sood, and Y. Huang, "Secure, resilient computing clusters: Self-cleansing intrusion tolerance with hardware enforced security (SCIT/HES)," in *Proc. IEEE 2nd Int. Conf. ARES*, 2007, pp. 343–350.

[137] P. E. Veríssimo, N. F. Neves, and M. P. Correia, "Intrusion-tolerant architectures: Concepts and design," in *Architecting Dependable Systems*. Berlin, Germany: Springer-Verlag, 2003, pp. 3–36.

[138] H. Johansen, A. Allavena, and R. Van Renesse, "Fireflies: Scalable support for intrusion-tolerant network overlays," *ACM SIGOPS Oper. Syst. Rev.*, vol. 40, no. 4, pp. 3–13, Oct. 2006.

[139] A. Siddiqui, D. Romascanu, E. Golovinsky, and R. Smith, "Real-time application quality of service monitoring (RAQMON) MIB," Internet Draft, Lincroft, NJ, USA, Tech. Rep., Oct. 2002.

[140] B. Joshi, A. S. Vijayan, and B. K. Joshi, "Securing cloud computing environment against DDoS attacks," in *Proc. IEEE ICCCI*, 2012, pp. 1–5.

[141] S. Khattak, N. Ramay, K. Khan, A. Syed, and S. Khayam, "A taxonomy of botnet behavior, detection, and defense," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 898–924, 2nd Quart. 2013.

[142] S. Zhao *et al.*, "Cloud-based push-styled mobile botnets: A case study of exploiting the cloud to device messaging service," in *Proc. 28th ACM Annu. Comput. Security Appl. Conf.*, 2012, pp. 119–128.

[143] K. Singh, S. C. Guntuku, A. Thakur, and C. Hota, "Big data analytics framework for peer-to-peer botnet detection using random forests," *Inf. Sci.*, vol. 278, pp. 488–497, Sep. 2014.

[144] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 27–51, 1st Quart. 2014.

[145] S. Lim, J. Ha, H. Kim, Y. Kim, and S. Yang, "A SDN-oriented DDoS blocking scheme for botnet-based attacks," in *Proc. IEEE 6th ICUFN*, 2014, pp. 63–68.

Nazrul Hoque received the M.Tech. degree in information technology from Tezpur University, Tezpur, India, in 2012. Currently, he is pursuing the Ph.D. degree in the Department of Computer Science and Engineering at Tezpur University. His research interests are machine learning and network security.

Dhruba K. Bhattacharyya received the Ph.D. degree in computer science from Tezpur University, Tezpur, India, in 1999, where he is currently a Professor in the Computer Science & Engineering Department. His research areas include data mining, bioinformatics, network security, and big data analytics. He has published over 220 research papers in leading international journals and conference proceedings. In addition, he has written/ edited 8 books. He is a Programme Committee/ Advisory Body member of several international conferences/workshops.

Jugal K. Kalita received the Ph.D. degree from the University of Pennsylvania, Philadelphia, PA, USA. He is a Professor of Computer Science at the University of Colorado, Colorado Springs, CO, USA. His research interests are in natural language processing, machine learning, artificial intelligence and bioinformatics. He has published over 150 papers in international journals and referred conference proceedings and has written two books.