Internet Denial of Service Attacks and Defense Mechanisms

MEHMUD ABLIZ
Department of Computer Science, University of Pittsburgh

Availability is one of the three main components of computer security, along with confidentiality and integrity. *Denial of service (DoS)* is a threat that potentially violates the availability of a resource in a system. Existing survey of DoS attack and defense mechanisms are relatively outdated, and do not reflect the significant developments in this area in recent years. In this article, we present an in-depth study of the denial of service problem in the Internet, and provide a comprehensive survey of attacks and their countermeasures. We investigate various DoS attack mechanisms, derive a more practical taxonomy of attack mechanisms, and summarize the challenges in DoS defense. We critically review the state of the art in DoS defense, analyze the strengths and weaknesses of different proposals, and conclude a comprehensive taxonomy of various defense mechanisms.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.3 [**Computer-Communication Networks**]: Network operation

General Terms: Availability, Botnet, Cyber Attack, Distributed Systems, DoS, DDoS, Firewall, Infrastructure, Internet, Intrusion Detection, IP spoofing, Security, TCP/IP, Taxonomy

1. INTRODUCTION

As Internet is increasingly being used in almost every aspect of our lives, it is becoming a critical resource whose disruption has serious implications. Blocking availability of an Internet service may imply large financial losses, as in the case of an attack that prevented users from having steady connectivity to major e-commerce Web sites such as Yahoo, Amazon, eBay, E*Trade, Buy.com, ZDNet and CNN [Sandoval and Wolverton 2000]. It may also imply threat to public safety, as in the case of taking down of Houston port system in Texas [McCue 2003] in 2003, or national security, as in the case of White House Web site becoming the target of Code Red worm attack [Lemos 2001] in 2001.

Such attacks that aimed at blocking availability of computer systems or services are generally referred to as denial of service (DoS) attacks. As more and more essential services become reliant on the Internet as part of their communication infrastructure, the consequences of denial of service attacks can be very damaging. Therefore, it is crucial to deter, or otherwise minimize, the damage caused by denial of service attacks.

The original aim of the Internet was to provide an open and scalable network among research and educational communities [Lipson 2002]. In this environment, security issues were less of a concern. Unfortunately, with the rapid growth of the Internet over the past decade, the number of attacks on the Internet has also increased rapidly. CERT Coordination Center reported that the number of reported Internet security incidents has jumped from six in 1988 to 137,529 in 2003 [CERT/CC 2009]. The annual Computer Security Institute (CSI) computer crime

and security survey reported that 30–40% of the survey participants were targeted by a DoS attack between 1999 and 2005 [Gordon et al. 2005], and 21–29% of the participants were targeted by a DoS attack during 2006 to 2009 time period [Peters 2009]. The 2010 Worldwide Infrastructure Security Report [Dobbins and Morales 2010] found that DoS attacks had gone mainstream, and network operators were facing larger, more frequent DDoS attacks. The volume of the largest single attack observed in 2010 period reached a staggering 100 Gbps point, a 1000 percent increase since 2005 [Dobbins and Morales 2010].

Preventing denial of service attacks can be very challenging, as they can take place even in the absence of software vulnerabilities in a system. Meanwhile, it is extremely hard, if not impossible, to precisely differentiate all attacker's requests from other benign requests. Thus, solutions that rely on detecting and filtering attacker's requests have limited effectiveness. There are various other technical and non-technical challenges that need to be well understood in order to design solutions that fundamentally address the problem. A comprehensive study that provides a clear analysis of the problem and solution space regarding the Internet denial of service attacks can be a tremendous help to researchers in providing better DoS solutions.

In this article, we present an in-depth study of the denial of service in the Internet, and aim to provide researchers with a clear view of (1) what is denial of service and distributed denial of service, (2) what are the targets and possible ways of carrying out DoS and DDoS attacks, (3) what are the challenges in defending against these attacks, (4) what has been done in terms of addressing these challenges and countering the various attacks , and (5) what problems still remain to be further investigated.

The main contribution of this article includes the following: first, we provide an extensive study of various attack mechanisms, and conclude a more practical taxonomy of DoS attack mechanisms; second, we give a detailed analysis of various challenges that researchers have to face when addressing the DoS problem; third, we extensively review the state of the art in the DoS research area, and provide an in-depth analysis of the strengths and weaknesses of notable DoS solutions as well as a taxonomy of various defense mechanisms; and last but not least, we provide suggestions regarding how to address the weaknesses of some of the DoS solutions.

The rest of the article is organized as follows. A formal definition of the terms availability and denial of service is described in Section 2, followed by a description of various attacks and a taxonomy of attack mechanisms. Section 3 discusses various challenges in solving DoS. Section 4 gives a taxonomy of DoS defense strategies and the possible deployment locations of DoS solutions in the Internet. Section 5, 6, 7, 8 discuss DoS prevention, detection, response, and tolerance mechanisms respectively. A conclusion is given in Section 9.

2. DOS AND TYPES OF ATTACKS

2.1 Denial of Service

Availability is one of the three main objectives of computer security, along with confidentiality and integrity. Availability can be defined as the ability to use the information or resource desired [Bishop 2002]. However, this definition of availabil-

ity skips an important aspect – timeliness. According to the Code of Laws of the United States regarding the definition of information security (44 U.S.C § 3542 (b) (1)), "availability means ensuring timely and reliable access to and use of information." So, we define availability as *the ability to use the desired information or resource in a reliable and timely manner.*

Denial of Service is a threat that potentially violates the availability of a resource in a system. A *Denial of Service Attack*, on the other hand, is an action (or set of actions) executed by a malicious entity to make a resource unavailable to its intended users. Gligor defines denial of service as follows [Yu and Gligor 1988]: "a group of otherwise-authorized users of a specified service is said to deny service to another group of otherwise-authorized users if the former group makes the specified service unavailable to the latter group for a period of time that exceeds the intended (and advertised) waiting time." This definition of denial of service takes into account the timeliness aspect of availability, and we use it as the standard definition of denial of service.

Denial of service attacks come in a variety of forms and aim at a variety of services. CERT Coordination Center defines three basic types of attacks [CERT/CC 1997] : 1) consumption of scarce, limited, or non-renewable resources, 2) destruction or alteration of configuration information, 3) physical destruction or alteration of network components. In this article, we are mainly concerned with the first type of attacks, i.e. attacks that consume a scarce, limited or non-renewable resource. The targeted resources can be network bandwidth, CPU, memory, I/O bandwidth, disk space, or any combination of them.

2.2 Vulnerability-based and Flooding Attacks

The different types of denial of service attacks can be broadly classified into *vulnerability attacks* (also called semantic attacks) and *flooding attacks* (also called brute-force attacks).

A DoS vulnerability attack exploits one or more flaws in a policy or in the mechanism that enforces the policy, or a bug in the software that implements the target system, and aims to consume excessive amount of resources of the target by sending it a few carefully crafted requests. For example, in the Ping-of-Death (POD) attack, an attacker cause certain operating systems to crash or reboot by sending a fragmented oversized ICMP (Internet Control Message Protocol) datagrams [CERT/CC 1996a].

A DoS brute-force attack, on the other hand, aims to deny service to legitimate users of a service by invoking vast amount of seemingly valid service requests and trying to exhaust a key resource of the target. For example, in a User Datagram Protocol (UDP) flood attack, an attacker sends excessively high number of UDP segments to random ports on a target host to saturate its bandwidth, rendering the target unreachable by other hosts [CERT/CC 1996c].

2.3 Single -source and Distributed Attacks

In a denial of service attack, attackers may launch their attacks from a single host or from multiple hosts that they control. When attacker's attack messages are originated from multiple hosts that are distributed in the network, it is called a *distributed denial of service (DDoS) attack.* In contrast, when attacker's attack

messages are generated from a single host, we call it a *single-source denial of service (SDoS) attack*. The term DoS may be used in some literature to refer only to the single-source denial of service. However, to avoid confusions, we strictly use DoS to refer to both distributed and single source attacks, and explicitly state whether it is distributed or single source when such clarification is necessary.

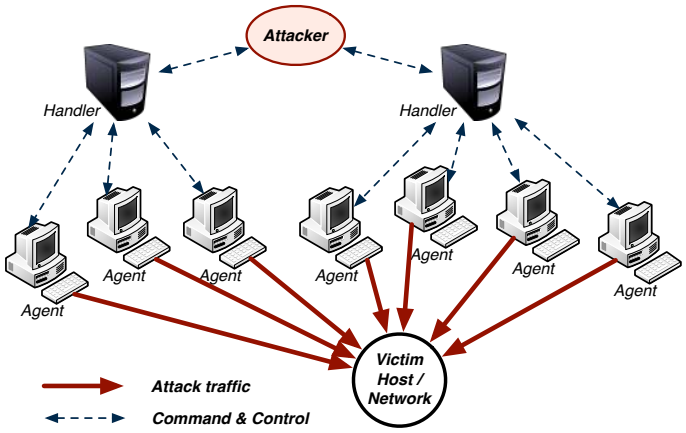

Fig. 1. Distributed Denial of Service Attack

A DDoS attack typically use two types of components: *agents*, which run on compromised hosts and generate the actual attack messages; and a *handler*, which is a program that controls the agents, telling them when to attack, what to attack, and how to attack [Scarfone et al. 2008]. Agents are also referred to as *bots*, and a collection of hosts that are running bots that are controlled by a single attacker is called a *botnet*. The Figure 1 illustrates the steps of a typical DDoS attack. First, an attacker compromises vulnerable hosts in the Internet and deploys attack tools (agents) on them. Next, the attacker disseminates an attack command from the handlers to the agents, instructing the agents on what to attack, when to attack and how to attack. Starting at the instructed attack time, agents generate attack traffic towards to the target to carry out the attack.

There is also a class of DDoS attacks, in which hosts with vulnerabilities are exploited to generate the attack traffic without actually controlling the exploited hosts. The exploited hosts in these type of attacks are called *unwitting agents*. An example of an unwitting agent attack is the exploitation of Microsoft Internet Information Server (IIS) directory traversal vulnerability [US-CERT 2000] to trigger the *ping* utility in the target host to send ICMP Echo Request floods to the attack target.

Generally speaking, DDoS attacks are more powerful than SDoS attacks, since the amount of bandwidth, CPU, memory that can be packed into a single attack computer hardly surpasses the combined resources of hundreds or thousands of compromised machines. In practice, defending against DDoS attacks is proven to be harder than defending against SDoS attacks.

2.4 Targets of DoS

The targeted victim of a DoS attack can be an end system (a computer that implements all layers of OSI reference model [Shirey 2007]), a router, an ongoing communication, a link or an entire network, an infrastructure, or any combination of or variant on these [Handley et al. 2006]. In the case of an end system, the targeted victim can be an application, or an operating system. Note that the term *end system* corresponds to the terms "Internet host", "end host", or simply "host", where an *end host* or *host* is a computer that implements all five layers of TCP/IP protocol stack [Braden 1989].

2.4.1 *DoS on application.* In application DoS attacks, an attacker attempts to prevent the application from performing its intended tasks by causing the application to exhaust the finite supply of a specific resource. For example, in an eXtensible Markup Language (XML) parser DoS attack called Exponential Entity Expansion attack (also known as Billion Laughs attack), an attacker passes to an XML parser a small XML document that is both well-formed and valid, but expands to a very large file [Sullivan 2009]. When the parser attempts to parse the XML, it ends up consuming all memory available to the parser application. Usually, the resources for applications are constrained by configuration, such as the maximum number of processes and the maximum number of simultaneous connections that an application can create, to limit the impact of an application DoS on the entire operating system. However, if such limits are not chosen carefully based on the role of a machine (e.g., a Web server, a database server, or a personal computer etc.), important applications may become the easy targets of DoS.

2.4.2 *DoS on operating system.* Operating system DoS attacks are very similar to application DoS attacks. However, in application DoS attacks, the operating system may be able to protect other applications from being effected; whereas the problem can be more catastrophic in the case of operating system DoS attacks. A very well-known DoS attack on an operating system is the Transmission Control Protocol (TCP) SYN flooding [CERT/CC 1996b], in which an attacker sends a flood of TCP SYN packets to the victim without completing the TCP handshake, and exhausting victim's connection state memory. Such an attack effects all applications in the operating system that relies on TCP for their communication.

2.4.3 *DoS on router.* Many of the DoS attacks against an end system can also be launched against an IP router. Additionally, routing protocols can be used to stage a DoS attack on a router or a network of routers [Handley et al. 2006]. This requires the ability to send traffic from addresses that might plausibly have generated the relevant routing messages. The simplest attack on a router is to overload the routing table with sufficiently large number of routes that the router runs out of memory, or the router has insufficient CPU power to process the routes [Chang et al. 2002]. More serious DoS attacks on routers that use false route updates can cause blackholing of an entire network address block [Nordström and Dovrolis 2004].

2.4.4 *DoS on ongoing communication.* Instead of attacking the end system, an attacker may attempt to disrupt an ongoing communication. If an attacker can

observe a TCP connection, then it is relatively easy to spoof packets to either reset that connection or to de-synchronize it so that no further progress can be made [Joncheray 1995]. Even if an attacker cannot observe a TCP connection, but can infer that such a connection exists, it is still possible to reset or de-synchronize that connection by sending large number of spoofed TCP reset packets that guess the TCP port number and TCP sequence number.

2.4.5 *DoS on links.* The simplest form of DoS attack on links is to send enough non-congestion-controlled traffic (e.g., UDP traffic) such that a link becomes excessively congested, and legitimate traffic suffers unacceptably high packet loss [Handley et al. 2006]. Congesting a link might also cause a routing protocol to drop an adjacency if sufficient routing packets are lost, potentially amplifying the effects of the attack. Moreover, it may be possible for an attacker to deny access to a link by causing the router to generate sufficient monitoring or report traffic such that the link is filled. Simple Network Management Protocol (SNMP) traps [Rose 1991] are one possible vector for such an attack, as they are not normally congestion controlled.

2.4.6 *DoS on infrastructure.* Many communication systems depend on some underlying infrastructure for their normal operations. Such an infrastructure can be as large as a global domain name system or a global public key infrastructure, or can be as small as a local area ethernet infrastructure or a wireless access point. Effects of infrastructure attacks on the users of that infrastructure can be enormous. For example, Domain Name System (DNS) serves as the phone book for the entire Internet by translating human-friendly hostnames into IP addresses. Denying access to a DNS server effectively denies access to all services, such as Web, email, AFS, public keys and certificates etc, that are being served by that DNS server. The larger the zone a DNS server is responsible for, the bigger the impact of a DoS attack. All 13 root DNS servers were subjected to a very large-scale DoS attack in 2002 [Vixie et al. 2002]. As a result, some root name servers were unreachable from many parts of the global Internet.

Some DoS attacks target a common infrastructure that is conjointly used by all hosts in a local network. For example, an attack with access to a subnet may be able to prevent other local hosts from accessing the network by simply exhausting the address pool allocated by a Dynamic Host Configuration Protocol (DHCP) server [Handley et al. 2006]. Although such attacks require the ability to spoof MAC address of of an ethernet or wireless card, it is quite feasible with certain hardware and operating systems.

2.4.7 *DoS on firewalls and IDS.* Firewalls are intended to defend the systems behind them against outside threats by restricting data communication traffic to and from the protected systems [Shirey 2007]. Firewalls may also be used in defending against denial of service attacks. Meanwhile, firewalls themselves may become the targets of DoS attacks. Firewalls can be categorized as stateful and stateless, based on whether the firewall holds state for the active flows traversing it, where a *flow* is a stream of packets sharing IP source and destination addresses, protocol field, and source and destination port numbers.

Stateless firewalls generally can be attacked by attempting to exhaust the process-

ing resources of the firewall. In addition to processing power exhaustion, stateful firewalls can also be attacked by sending traffic that causes the firewall to hold excessive state or state that has pathological structure [Handley et al. 2006]. In the case of excessive state, the firewall may run out of memory, and can no longer instantiate the state required to pass legitimate flows. For most firewalls this will cause denial of service to the systems behind the firewall, since most firewalls are fail-disconnected. In the case of pathological structure, an attacker sends traffic that causes the firewall's data structures to exhibit worst-case behavior. An example of this would be an algorithmic complexity attack on the firewall's forwarding state hash table [Crosby and Wallach 2003].

Intrusion detection systems (IDSs) suffer from similar problems to that of firewalls. Unlike a firewall, an IDS is normally fail-open, which will not deny service to the systems protected by the IDS. However, it may mean that subsequent attacks that the IDS would have detected will be missed.

2.5 DoS at Different Protocol Layers

Attack target based classification of DoS attacks in previous subsection represents a *horizontal classification* based on the horizontal distribution of attack targets in the Internet. Since Internet is vertically structured into multiple protocol layers, it is natural to have a *vertical classification* of Internet DoS attacks. Hence, DoS attacks are categorized into different groups based on the layer of the TCP/IP stack the targeted service or protocol belongs to. To further explain this classification, let us consider the DoS attacks on end systems and routers. End systems implement all five layers of the TCP/IP stack, thus in principle it is possible for an attacker to target services or protocols that belong to any one of these layers. In the case of routers, they implement network layer and below, hence some of the attacks that are possible against end systems may not be possible against routers.

As this article focuses on attacks at the Internet layer and above, here we only give overview of these attacks. We discuss attacks in transport layer together with attacks in the Internet layer. We think this is helpful when reasoning about solutions for bandwidth flooding attacks, since the two major protocols, TCP and IP, of these two layers were designed as a single protocol in the original Internet architecture [Clark 1988]. Note that the Internet layer is commonly referred to as network layer, and they are used interchangeably in this article.

2.5.1 *DoS at the network and transport layers.* The Internet layer provides host-to-host connectivity via interconnection of various networks. The major protocol in this layer is the Internet Protocol (IP), that is a connectionless, best-effort packet switching protocol [Malkin 1996]. The datagram or connectionless nature of the IP protocol is a fundamental characteristic of the Internet architecture. Other important protocols in this layer include ICMP [Postel 1981a] (a control protocol that provides error reporting, congestion reporting, and first-hop gateway redirection) and IGMP [Cain et al. 2002] (a protocol for establishing dynamic host groups for IP multicasting). The transport layer provides end-to-end communication services for applications. There are two primary transport layer protocols: Transmission Control Protocol (TCP) [Postel 1981b] and User Datagram Protocol (UDP) [Postel 1980].

We have already described several network layer and transport attacks in previous sections, such as Ping-of-Death [CERT/CC 1996a], UDP flood [CERT/CC 1996c], and TCP SYN flood [CERT/CC 1996b]. Other well-known attacks in these layers include ICMP ping flooding attacks [Burch 2000] and TTL Expiry attacks [Systems 2009]. In a ping flooding attack, an attacker sends ICMP echo requests at a very fast rate to the targeted host or router. In a TTL Expiry attack, an attacker sends a flood of IP packets whose Time to Live (TTL) values are set to expire at the targeted router along the path. Bellovin described a wide range of attacks against TCP/IP protocol suites in [Bellovin 1989], including connection reset attacks using ICMP *Destination Unreachable* and *Time to Live Exceeded* messages and through TCP sequence number prediction. There are also IP Multicast-based DoS attacks, as described in [Handley et al. 2006]. An attacker may cause memory exhaustion on routers by joining a large number of multicast groups and making routers to hold a large amount of multicast routing state. IP multicasting may also be abused by attackers to reflect and amplify attack traffic as we will see later in this section.

Two main strategies of network and transport layer attacks are exhausting bandwidth via a flood of packets and exhausting processing power or memory via exploitation of a particular flaw in the design or a bug in the implementation of a system. In comparison, the bandwidth flooding is a common problem that is relevant to almost all parts of the Internet and needs a common solution.

2.5.2 *DoS at the application layer.* The application layer is the top layer of the Internet protocol suite. There are two main categories of application layer protocols: user protocols that provide service directly to users, and support protocols that provide common system functions [Braden 1989]. The most common user protocols are: Telnet, FTP, HTTP, IMAP, SMTP/POP, SSH, IRC, XMPP etc. Common support protocols include SNMP, DNS, BOOTP/DHCP, NTP, RTP, SIP, TLS/SSL etc. Some routing related application layer protocols, such as BGP and RIP, are only implemented in the routers.

Virtually any one of the protocols shown above may become the object or means of an DoS attack. Since most application layer protocols are organized in terms of client-server model, we use it extensively when describing application layer attacks. A *server* is a process implementing a specific service, for example, a file transfer service or an email service. A *client* is a process that requests a service from a server by sending it a request and subsequently waiting for the server's reply. Sometimes, the terms *client* and *server* are also used refer to the machines that runs the client process and the server process respectively. Clients are further classified as *legitimate clients* that do not contain any malicious logic and *malicious clients* that contain malicious logic. Agents in DDoS attacks are also referred to as malicious client when discussing application layer attacks.

Attackers may aim to exhaust CPU or memory of a target by sending it a large number of service requests, where each request causes the target to perform CPU and/or memory intensive operations. For example, an attacker can instruct malicious clients to send HTTP requests to download a large file from a server. Since the server has to read the large file from the hard disk into memory and send it to the malicious client in many packets, a simple HTTP request may trigger heavy resource consumption at the server in terms of I/O, memory, CPU, and bandwidth.

However, the attack in this example is fairly easy to detect and block. A more sophisticated attacker may mix HTTP requests for various different URLs to closely mimic the normal Web traffic. Other examples of application layer DoS attacks are email bombs [Bass et al. 1998], Session Initiation Protocol (SIP) flooding attacks [Sisalem et al. 2006], DNS flooding attacks [Ballani and Francis 2008] etc.

Attackers may also attempt to exhaust the target's downstream link bandwidth by sending application layer protocol messages at a high rate. However a high enough rate that can exhaust all the downstream bandwidth available to the target most probably trigger the initiation of network layer bandwidth flooding countermeasure, assuming such countermeasures are already deployed. If it is clear that deploying defense mechanisms for bandwidth exhaustion attacks at network layer is more effective and efficient, application layer DoS defense mechanisms should focus on protecting other types of resources such as processing power, memory, I/O, disk space, and upstream link bandwidth.

2.6 Attack Reflection and Amplification

Attackers can render denial of service attacks more difficult to defend against by bouncing their attack traffic off of other hosts in a network, or by amplifying the amount of attack traffic received by the attack target. Such DoS attack techniques are referred to in the literature as *attack reflection* and *attack amplification* respectively. Reflection and amplification techniques are used in combination in many past attacks, thus they are explained together here.

2.6.1 *Reflected attack.* Instead of sending attack requests directly to the targeted victim, a reflected DoS attack sends requests that use victim's address as the source address to *reflectors* that will in turn send their replies to the victim [Paxson 2001]. A reflector can be any host that will return a packet if sent a packet. Web servers, DNS servers, and routers are examples of a reflector. Web servers and DNS servers will return SYN ACKs or RSTs in response to SYN or other TCP packets, and routers return ICMP Time Exceeded or Host Unreachable messages in response to particular IP packets. Eliminating IP address spoofing does not address the reflected attack problem entirely, due to the application-level reflectors such as recursive DNS queries and HTTP Proxy requests [Paxson 2001].

2.6.2 *Attack amplification.* In attack amplification, attackers exploit services that generate one disproportionately large message or multiple messages for each message they receive to amplify the amount of attack traffic directed towards the targeted victim. For example, in a Smurf DoS attack [CERT/CC 1998], attackers exploit the IP network broadcasting service and send ICMP echo request packets to the IP multicast address of an intermediary network to amplify the attack traffic. If the intermediary network does not filter ICMP traffic directed to IP multicast addresses, many of the machines on the network will receive this ICMP echo request packet and send an ICMP echo reply packet back. To direct the ICMP echo reply packets towards the victim, attackers use the victim's IP address as the source address in the ICMP echo request. In this attack, each ICMP request sent by the attacker generates N reply messages from the intermediary network, where N is approximately the number of hosts in the intermediary network. Such an attack achieves a $N : 1$ amplification.

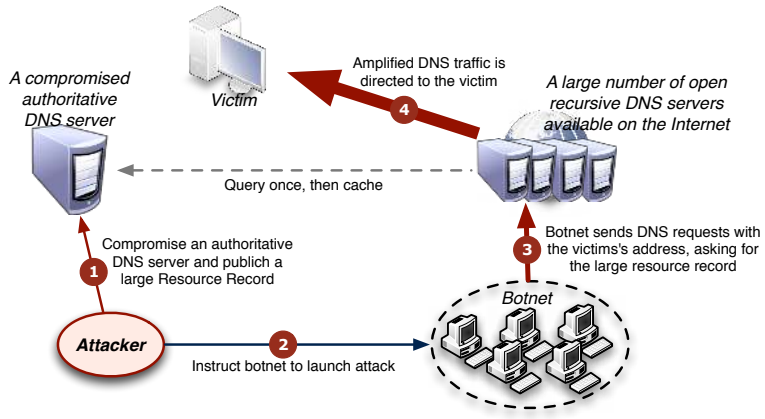

Fig. 2. An example of DoS amplification: a DNS amplification attack

A Smurf DoS attack uses both reflection (spoofing source IP address) and amplification (exploiting IP broadcast), and shows that reflection and amplification techniques are usually used in tandem. Another attack example that includes both reflection and amplification is DNS (Domain Name Service) amplification attack [CERT/CC 2000]. Figure 2 illustrates an example of a DNS amplification attack that was observed in 2006 [VeriSign 2006]. This attack involves more than 32,000 open recursive domain name servers. An attacker first compromises an authoritative DNS server and publishes a large 4K byte resource record (RR). The attacker then instructs the botnet to send DNS requests with the victim's IP address to the large number of open recursive servers, asking for the large resource record. The open recursive servers resolve the query, cache the result, and return the large resource record to the victim. For each 56 byte DNS query initiated by the bots, a 4,028 bytes of response is generated, achieving a 72:1 amplification.

2.7 Attack Taxonomy

Based on the previous discussions of various attack types, we derived a taxonomy of denial of service attacks, as illustrated in Figure 3. Several taxonomies of DoS attacks exist in the literature, such as [Mirkovic and Reiher 2004; Douligeris and Mitrokotsa 2004; Asosheh and Ramezani 2008]. Our classification differs from these taxonomies in several aspects. First, we aim to provide a taxonomy for DoS attacks in general, whereas their taxonomies focuses only on DDoS attacks. Second, we do not attempt to classify the attacks based on the characteristics of the botnet that is used during the attack. Although DDoS is closely related to botnets, so do other attacks such as spam, spyware, click fraud etc. A separate taxonomy for botnets, such as [Dagon et al. 2007], might be more appropriate. Third, we emphasize the practicality of the taxonomy. Hence, we do not use classification criteria that are hard to apply in practice, such as "possibility of characterization" (for a particular DoS attack, we cannot tell if it is characterizable or not until someone actually did characterize it).

We do not claim that the taxonomy we are proposing is complete. For some of

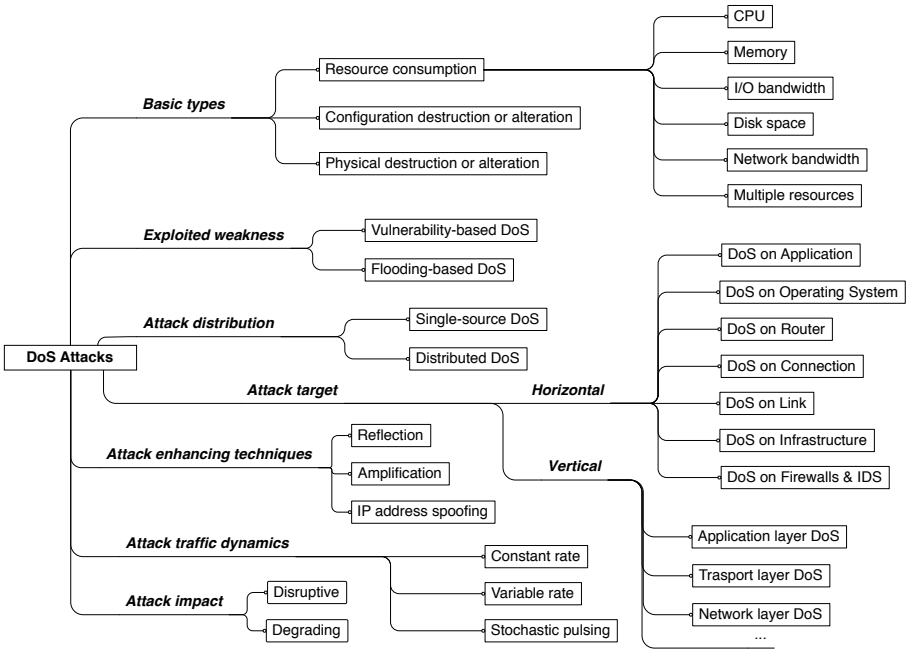

Fig. 3. A practical taxonomy of denial of service attacks

the attacks types in the leaves of the taxonomy tree in Figure 3, it is possible to further divide them into sub-categories.

3. DENIAL OF SERVICE DEFENSE CHALLENGES

Despite the tremendous effort by researchers and experts to address the denial of service problem, Internet denial of service attacks still remain to be an unsolved problem. There are various technical and non-technical challenges that need to be well understood in order to design solutions that fundamentally address the problem, while guaranteeing practicality of deployment. In this section, we first explore the design principles of the Internet and their implication to the denial of service problem. Then, we look at other technical challenges and how they effect the solution to the denial of service problem.

3.1 Internet Architecture Related Challenges

The architecture of the Internet is based on a number of principles, including the packet switching, multipath routing, end-to-end argument, distributed management of resources etc [Clark 1988]. While these design principles led to a robust, scalable, and cost-effectiveness Internet that supports multiple types of networks and protocols, they also created a challenging environment for preventing malicious parties from inflicting damage on others. Although the original design goals of the Internet recognized the need to be robust in the presence of external attack, there was no equivalent concern with regard to the possibility of attacks by the Internets

own users [Lipson 2002]. Thus, the Internet designers' view was that the user community was essentially benign and trustworthy, so simple security measures would be sufficient to ensure the protection of the Internet community. Under this worldview, provisions to track and prevent malicious user behavior were never designed or implemented.

In the following, we look at various Internet design principles and choices, and give an analysis of how each design choice translates into a challenge in addressing the denial of service problem. The order of these principles reflects the importance of different design goals of the Internet presented in [Clark 1988].

3.1.1 *On-demand resource sharing.* The fundamental structure of the Internet is a packet switched communications facility in which networks are inter-connected together using store and forward packet communications processors [Clark 1988]. Packet switching allocates link use on demand, and link capacity will be shared on a packet-by-packet basis among the users who have packets that need to be transmitted. In such environment, a misbehaving user can disrupt service for other users by occupying most of the shared resources. Such resource sharing based on users' demand creates an inter-user dependency. Gligor [Gligor 1984] points out that inter-user dependency is a fundamental factor that enables denial of service to occur.

3.1.2 *Simple core and complex edge.* The end-to-end argument in system design [Saltzer et al. 1984] advocates that "end-to-end protocol design should not rely on the maintenance of state inside the network; such state should be maintained only in the end points, in such a way that the state can only be destroyed when the end point itself breaks." The end-to-end principle leads to the simplicity principle of the Internet architecture, where the complexity of the Internet belongs at the edges, and the network layer of the Internet remains as simple as possible [Bush and Meyer 2002]. Since the interconnection network remains simple, intermediate routers in the network do not have the necessary functionality to detect and limit misbehaving traffic. Furthermore, to meet the requirements of services that demand high-bandwidth and low latency, routers are designed to push packets through as quickly as possible. Therefore, it is not considered practical for a router to do much in the way of processing packets other than routing them. Adding DoS prevention and detection functionalities that require any significant amount of extra processing to the intermediate routers might be undesirable in terms of achieving performance goals.

3.1.3 *Multi-path routing.* One of the most important goals of Internet design is survivability, i.e. the ability to continue communication even though networks and gateways (routers) are failing [Clark 1988]. Thus, the Internet routing infrastructure is designed with the ability to route traffic along alternative paths that bypass failing portions of the network. If packets from the same source address are always routed through the same path, then a router knows the possible set of addresses that an incoming packet at its particular network interface can have as its source address. Consequently, a router can tell a source IP address is spoofed when it receives a packet whose source IP is not in the possible set of addresses. The possibility of multi-path routing diminishes routers' ability to determine spoofed source

addresses, since a router may receive an unexpected packet due to route changes (not just because of spoofed source address). Thus, multi-path routing makes it more challenging to trace the origin of attack packets in the Internet.

3.1.4 *Decentralized management.* Another important design goal of the Internet architecture is that it must permit distributed management of its resources [Clark 1988]. Current Internet can be seen as interconnection of many Autonomous Systems (AS), where each autonomous system is a set of routers and links under a single technical administration. Each autonomous system defines its own set of operating policy and security policy. The enforcement of a global security policy or mechanisms is enormously difficult, which makes solutions that require cross-domain cooperation unattractive. On the other hand, many distributed denial of service attacks may not be mitigated at a single-point, and require the defense mechanisms to be deployed at multiple locations in the Internet. Designing solutions that can satisfy these conflicting requirements is hard.

3.1.5 *Accountability.* Accountability is defined as "the property that ensures that the actions of a system entity may be uniquely traced back to that entity" [Shirey 2007]. Although one of the original goals of the Internet architecture states that "the resources used in the Internet architecture must be accountable", it was put as a last goal [Clark 1988]. The accountability issue received very little attention during the early stages of the design, and is only now being considered. Unlike the telephone system, which has an effective tracking and billing capability based on the need to charge users of its services on a per-call basis, the Internet has no standard provisions for tracking or tracing the activity of its users.

One of the manifestations of lack of accountability in the Internet is the lack of any provision for guaranteeing the authenticity of IP addresses. IP addresses in Internet embody the dual role of locators and end-point identifiers [Lear and Droms 2003]. Each IP address names a topological location in the Internet, thereby acting as a routing direction vector or locator. At the same time, an IP address names the physical network interface currently located at the point-of-attachment, thereby acting as an end-point identifier.

Users with sufficient privileges on a host can generate IP packets with source IP address field set to an address other than the legally-assigned address of that host. This is called *IP address spoofing*. IP address spoofing is frequently used in denial of service attacks. Attackers use IP address spoofing to hide the true origin of attack messages, or they can amplify or reflect attack traffic using address spoofing as we saw in section 2.6. Attackers can use multiple spoofed source addresses for the attack traffic originating from the same attacking machine to achieve diffusion of traffic floods, making threshold based rate-limiting and attack detection mechanisms ineffective.

Attackers can forge source IP addresses in several ways. They can randomly pick an address from the entire IP address space, called *fully random spoofing*, or they can choose from IP addresses allocated to the subnetwork that the attacking host belongs, called *subnet spoofing*. Attackers may also avoid using non-routable IP addresses defined in RFC 1918 [Rekhter et al. 1996] or any other non-routable IP addresses, and such address spoofing is called *routable address spoofing*. Last but

not least, attackers can spoof IP addresses that are allocated to the subnetwork of the destination host as the source address, creating extra traffic floods between the target destination and its neighbors.

3.1.6 *Variation in link capacity.* The provisioning of link bandwidth in modern Internet varies significantly from core networks to edge networks. As core networks need to accommodate heavy traffic from many sources to many destinations, their links are highly provisioned. In fact, the dynamics of the Internet backbones favor 1:1 over-provisioning to maintain consistent performance and a reasonably stable network [Bush and Meyer 2002]. In comparison, an edge network only needs to support a smaller size of end users, requiring much less bandwidth. A side effect of this design is that the traffic from the high-bandwidth core link can overwhelm the low-bandwidth edge link, causing a denial of service, if many sources attempt to talk to the same destination simultaneously.

3.2 Other Challenges

In addition to the architectural challenges described above, there are several other challenges that make the Internet DoS hard to defend against. A brief description of these challenges is given next.

3.2.1 *Difficulty of distinguishing malicious requests.* It is difficult to distinguish between malicious requests and legitimate ones. This is true for packets, network flows, transport layer segments, or application service request messages. Even if certain malicious behavior can be reliably detected by signature based attack detection mechanisms, attackers usually modify the characteristics of their attack messages to evade the detection. It is well-known fact that attackers and defenders are locked in an arms race, especially when it comes to signature-based attack detection. Although anomaly based detection mechanisms can detect unknown attacks, they are not very reliable due to the possibility of misidentifying normal behavior as an attack.

Both signature-based and anomaly based detection techniques may work well for certain semantic attacks that rely on exploiting certain vulnerability in a system. However, the detection becomes enormously difficult when it comes to highly distributed flooding attacks, since such attacks do not have to restrict their attack messages to exploit certain vulnerability. Consequently, flooding attack packets need not be malformed (e.g., contain invalid fragmentation field or a malicious packet payload) to be effective. In another word, attackers are free to create attack message that are indiscernible from legitimate request messages. Lastly, in principle it is not possible to distinguish between a sufficiently subtle DoS attack and a flash crowd (a flash crowd occurs when an extremely large number of users all attempt to access the same service) [Handley et al. 2006].

3.2.2 *Asymmetry of request and response overhead.* Asymmetry of request and response overhead refers to the asymmetry in the amount of consumed resources for generating a request at the client and creating its corresponding response at the server. In most cases, a client spends trivial amount of CPU and memory resources to generate a requests, and the operations carried out by the server to produce the corresponding response incurs significantly more resource overhead in comparison.

Making matters worse, attackers can create their malicious requests off-line prior to the attack, further minimizing the overhead of generating a service request.

3.2.3 *DoS research challenges.* Mirkovic et al. argue that the advance of DoS defense research historically has been hindered by the lack of attack information, the absence of standardized evaluation, and the difficulty of large-scale testing [Mirkovic et al. 2004]. They argue that very limited information about DoS incidents are publicly available due to organizations' unwillingness to disclose the occurrence of an attack, for fear of damaging the business reputation of the victim. Without detailed analysis of real-world DoS attacks, it is difficult to design imaginative solutions to the problem.

In terms of standardized evaluation, there is no standard for evaluating the effectiveness of a DoS defense system. And the lack of standard evaluation approaches often leads to a situation where researchers and designers are allowed to present testing and evaluation results that are most advantageous to their system. It also makes it very difficult to compare the performance of various solutions. Moreover, the testing of DoS solutions in a realistic environment is immensely challenging, due to the lack of large-scale test beds or detailed and realistic simulation tools that can support Internet-scale network of nodes.

4. DEFENSE STRATEGIES AND DEPLOYMENT LOCATION

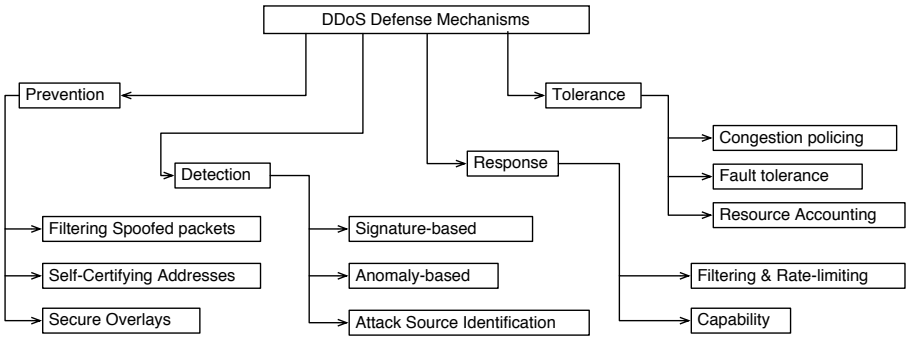

Fig. 4. A practical taxonomy of DoS defense mechanisms

The strategies of various denial of service defense mechanisms can be broadly divided into four categories: *prevention*, *detection*, *response*, and *tolerance*. *Prevention* approaches attempt to eliminate the possibility of DoS attacks or prevent the attack from causing any significant damage. *Detection* can be further classified as *attack detection* and *attack source identification*. *Attack detection* monitors and analyzes events in a system to discover malicious attempts to cause denial of service. It is an important step before directing further actions to counter an attack. *Attack source identification*, on the other hand, aims to locate the attack sources regardless of whether the source address field of malicious requests contain erroneous information. *Response* mechanisms are usually initiated after the detection of an attack

to eliminate or minimize the impact of the attack on the victim. *Tolerance* aims to minimize the damage caused by a DoS attack without being able to differentiate malicious actions from legitimate ones. It might be necessary to merely know the fact that a system is under attack, in order to initiate the tolerance mechanisms.

For each of the four broad defense categories, we can further divide them into different defense mechanism types, based on the similarity of different solutions. Figure 4 illustrates the taxonomy of defense mechanisms that we created to classify the existing DoS solutions.

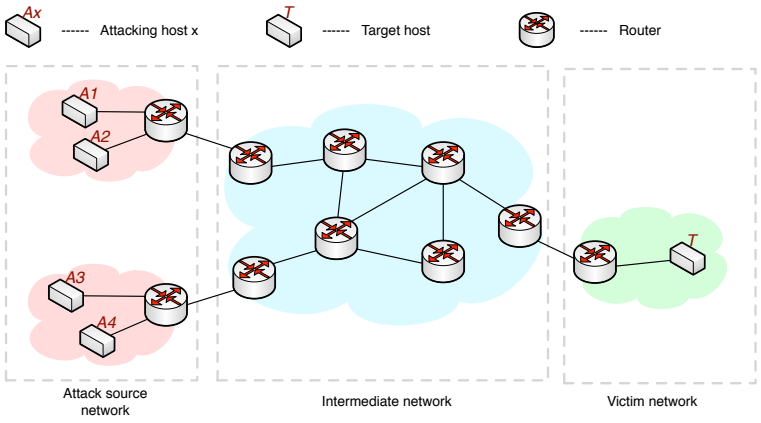

Fig. 5. A simplified network illustrating different locations for deploying DDoS defense

Since DoS attacks can be defended at different locations in the network, defense mechanisms can also be divided into different categories based on the deployment location. Mirkovic et al. provides four possible locations in the Internet for deploying DDoS defense in [Mirkovic et al. 2004]. They are *near the target*, *intermediate network*, *near the attack source*, and *multiple locations*. Figure 5 shows a highly simplified network diagram of the Internet, with the demonstration of different locations for deploying DDoS defense.

5. PREVENTION MECHANISMS

Denial of service prevention mechanisms aim to stop attacks before they actually cause damage. Prevention mechanisms include, but not limited to, spoofed packet filtering, self-certifying addresses, and secure overlays.

5.1 Filtering Spoofed Packets

Many DoS attackers rely on IP address spoofing to hide the origin of an attack. As we seen in Section 2, attack reflection and amplification techniques rely on IP address spoofing. Filtering mechanisms are designed to prohibit DoS attack traffic with spoofed source addresses from reaching the target, by dropping packets with false IP addresses.

5.1.1 *Martian Address Filtering and Source Address Validation.* Martian address filtering and source address validation are defined in the Requirements for IP Version 4 Routers (IETF RFC 1812) [Baker 1995]. Martian address filtering specifies that a router should not forward any Martian packet (or a packet from Mars), where a Martian packet is a packet whose source or destination specifies an IP address designated by the Internet Assigned Numbers Authority (IANA) as reserved or special-use from or to which packets cannot actually originate or be delivered. Latest special-use IPv4 addresses are defined in IETF RFC 5735 [Cotton and Vegoda 2010], and other examples of invalid IP addresses include IP addresses from as-yet-unallocated range (Bogon addresses [Rekhter et al. 1996]) and a destination address of 255.255.255.255/32 etc.

Source address validation specifies that a router should implement the ability to filter traffic based on a comparison of the source address of a packet and the forwarding table for a logical interface on which the packet was received [Baker 1995]. If this filtering is enabled, the router must silently discard a packet if the interface on which the packet was received is not the interface on which a packet would be forwarded to reach the address contained in the source address. In simpler terms, if a router wouldnt route a packet containing this address through a particular interface, it should not believe the address if it appears as a source address in a packet read from this interface.

Martian address filtering eliminates the possibility of spoofing for a small set of addresses. Attacker can simply avoid spoofing any Martian addresses. Source address validation can eliminate majority of source address spoofing. However, with the number of asymmetric routes in the Internet, it is quite possible that the return path to a given packet's source address may not flow out the same interface as that packet arrived upon. Hence, using such technique to filter packets causes collateral damage to legitimate users' traffic.

5.1.2 *Ingress/Egress Filtering.* The purpose of ingress/egress filtering is to allow traffic to enter or leave the network only if its source addresses are within the expected IP address range. *Ingress filtering* refers to filtering the traffic coming into a network, and *egress filtering* refers to filtering the traffic leaving the network. Using network ingress filtering to counter DoS attacks is introduced in RFC 2827 as a Best Current Practice (BCP) [Ferguson and Senie 2000].

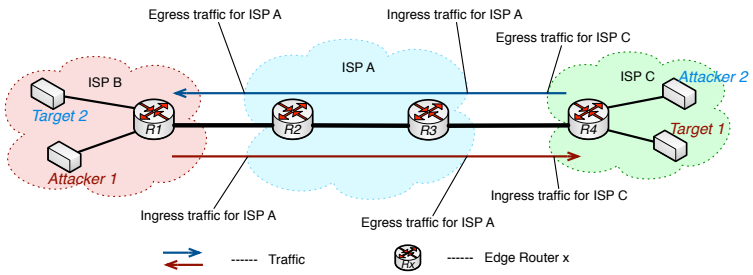

Fig. 6. An example of ingress/egress filtering

The concepts of ingress filtering and egress filtering are illustrated in Figure 6. In the figure, attacker 1 resides within the network 204.69.207.0/24, which is provided Internet connectivity by Internet service provider (ISP) A. An input traffic filter on the ingress (input) link of edge router $R2$, which provides connectivity to the attackers network, restricts traffic to allow only traffic originating from source addresses within the 204.69.207.0/24 prefix, and prohibits an attacker from using "invalid" source addresses which reside outside of this prefix range. This is an ingress filtering. If edge router $R1$, instead of $R2$, provided the same filtering function, then that will be called egress filtering.

The key requirement for in ingress or egress filtering is to know the expected IP addresses at a particular port. For some networks with complicated topologies, it is not easy to obtain this knowledge. Additionally, ingress/egress filtering does not provide a strong deployment incentive to Internet service providers, and it is currently partially deployed. Hence, attackers can carefully choose a network without ingress/egress filtering to launch DoS attacks that use spoofed source addresses. Moreover, ingress/egress filtering does not preclude an attacker using a forged source address of another host within the permitted prefix filter range.

5.1.3 *Route-Based Filtering.* Park and Lee proposed a route-based distributed packet filtering (DPF) approach to filtering out spoofed packet flows [Park and Lee 2001]. DPF uses routing information to determine if a packet arriving at a router — e.g., border router at an AS — is valid with respect to its inscribed source/destination addresses, given the reachability constraints imposed by routing and network topology. DPF uses information about the Border Gateway Protocol (BGP) [Rekhter et al. 2006] routing topology to filter traffic with spoofed source addresses.

There are several limitation of DPF. If multiple paths are permitted when routing packets from source to destination, it becomes significantly easy for attacks that use spoofed source IP addresses to elude route-based filtering. Moreover, DPF may drop legitimate packets if there has recently been a route change. Finally, the filtering rules in DFP have a very coarse AS level granularity, and attackers can still bypass the DFP filters by carefully choosing the range of IP addresses to spoof.

5.1.4 *Source Address Validity Enforcement Protocol.* To overcome the disadvantages of route-based filtering, Li et al. proposed *Source Address Validity Enforcement (SAVE) protocol* [Li et al. 2002]. SAVE constantly propagates messages containing valid source address information from the source location to all destinations. Thus, each router along the way builds an incoming table that associates each link of the router with a set of valid source address blocks. When a packet arrives on an interface, a router consults its incoming table to determine whether this packet comes from the proper direction.

SAVE overcomes the asymmetry of Internet routing by updating the incoming tables on each router periodically. However, it needs to change the routing protocol, which is a daunting task that may take a long time to accomplish. Moreover, as SAVE filters spoofed packets to protect other entities, it does not provide direct deployment incentives. As with ingress/egress and route-based filtering, when partially deployed, attackers can always spoof the IP addresses within networks that

do not implement SAVE.

5.1.5 *Hop-Count Filtering.* Jin et al. [Jin et al. 2003] introduced filtering packets with spoofed IP addresses using a method called *Hop-Counter Filtering (HCF).* They argue that although an attacker can forge any field in the IP header, he or she cannot falsify the number of hops an IP packet takes to reach its destination. They propose a method to infer this hop-count information from Time to Live (TTL) value in the IP header. Using the TTL-based hop-count computation method, a victim builds a hop-count to source IP address mapping table. When a victim receives a packet, it computes a hop-count for its IP address and compares it to the hop-count stored in the mapping table to identify address-spoofed packets. HCF stays in *alert* state by default, in which it monitors the trend of hop-count changes without discarding packets. Upon detection of a flux of spoofed packets, HCF switches to *action* state to examine each packet and discard spoofed IP packets.

The advantage of HCF is that it requires deployment at the victim, which is much easier to deploy compared with network based filtering approaches. Moreover, a potential victim has a much stronger incentive to deploy deploy defense mechanisms that the intermediate network service providers. However, HCF suffers from high false positives and false negatives. HCF hop counting method relies on the initial TTL value for computing the hop-count, and initial TTL values for different operating systems (OSs) are different. HCF hop count method fails to work when the difference between initial TTL values for different OSs are less than the average hop-counts between Internet end hosts, and initial TTL value difference for some OSs are indeed less that the average hop-count (for examples, initial TTL values 30, 32, 60, and 64). Legitimate packets may be identified as spoofed due to inaccurate IP to hop-count mapping or delay in hop-count update. Even assuming the hop-count computation is precise, attackers can still spoof IP addresses with the same hop-count as their machines do. Lastly, as with any other victim-based filtering approaches, HCF cannot prevent DDoS attack traffic that is overwhelming the link coming into the machine that is enforcing the filtering.

5.1.6 *IPv4 Source Guard.* IP Source Guard [Baker 2007] provides source IP version 4 (IPv4) address filtering on a Layer 2 port to prevent a malicious host from spoofing IP addresses. IP Source Guard snoops DHCP address assignments and uses static IPv4 source bindings to automatically configure each Layer 2 port to discard traffic if the source IP address is different from the IP address assigned to that port.

IP Source Guard is designed to be implemented on an IP or Ethernet switch that is used in small office/home office (SOHO), corporate, or access network. The premise of the IP Source Guard is that each host has one network interface and each interface has one address. IP Source Guard can prevent a legitimate host from carrying out an intended function if either of these assumptions is invalid in a network environment.

5.1.7 *Passport.* Passport is a source address validation framework designed by Liu et al [Liu et al. 2008]. It aims to ensure that no host or AS can spoof the address space of an AS that deploys Passport. Figure 7 illustrates how Passport works at a high level.

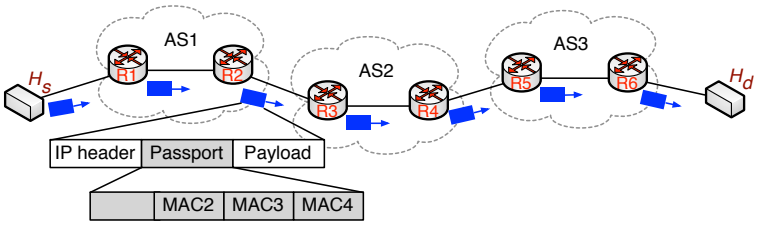

Fig. 7. Border router (R_2 in this case) of source AS stamps source authentication information onto the Passport header of an outbound packet. The border routers (R_3 and R_5 in the example) of intermediate or destination ASes verifies this information.

When a packet leaves its source AS, the border router stamps one Message Authentication Code (MAC) for each AS on the path to the destination. Each MAC is computed using a secret key shared between the source AS and the AS on the path. When the packet enters an AS on the path, the border router verifies the corresponding MAC using the secret key shared with the source AS. The verifying router uses the source address of the packet to look up the source AS, obtains the shared key, and recomputes the MAC. An AS can obtain the mapping between a source address and the corresponding source AS from BGP using the AS-PATH [Rekhter et al. 2006] path attribute. If a router stamps MACs for a source address outside its address space, the MACs will not verify at downstream ASes. A router erases the MAC value in a packet after verification to prevent offline cryptanalysis. A packet with an invalid MAC is demoted at an intermediateAS and is discarded at a destinationAS. Two ASes obtain the pair-wise secret key that is used in computing MACs by piggybacking a standard Diffie-Hellman key exchange [Diffie and Hellman 1976] in their BGP announcements.

Passport scheme only prevents hosts in one AS from spoofing the addresses of other ASes. As a result, attackers can spoof the IP address of any host within the same AS. Moreover, each border router that implements Passport scheme needs extra memory to store AS paths for all destination prefixes, shared secret keys with all ASes, this is a significantly large amount of memory considering the total number of prefixes and ASes in the Internet.

5.2 Self-certifying Addresses

The accountability problem received very little attention during the early stages of the design, and is only now being considered. One of the main problems that needs to be addresses is the accountability of IP addresses. The solutions that we discuss next attempt to solve this problem.

5.2.1 *Host Identity Protocol.* The *Host Identity Protocol (HIP) architecture* [Moskowitz and Nikander 2006] proposes a new namespace called *Host Identity namespace* and a new protocol layer called *Host Identity Protocol* [Moskowitz et al. 2008] between the internetworking and transport layers. The Host Identity namespace consists of *Host Identifiers* (HIs), where a Host Identifier is an public key of an symmetric key-pair. Each host can have more than one Host Identifiers, but no two hosts have the same Host Identifier. There is an important difference between Host Identity

and Host Identifier. An *Host Identity* refers to the abstract entity that is identified, while an Host Identifier refers to the concrete bit pattern that is used in the identification process.

The Host Identifier, can be either published, in which it is considered public, or unpublished. The public Host Identifiers should be stored in DNS or Lightweight Directory Access Protocol (LDAP) directories, or in any existing Public Key Infrastructure (PKI), whereas the unpublished Host Identifiers should only be stored on the host that owns them. Host Identifiers are only used in the built-in key agreement protocol called the *HIP base exchange*, and other protocols use *Host Identity Tag* (HIT), 128-bit hash of Host Identifier. The HITs identify the sender and recipient of a packet in *HIP packets*, that are IP packets that carry HIP protocol messages.

IP addresses in Internet embody the dual role of locators and end-point (a communicating entity) identifiers. In the HIP architecture the end-point names and locators are separated from each other. IP addresses continue to act as locators. The Host Identifiers take the role of end-point identifiers. Here, the end-point names based on Host Identities are slightly different from interface names; a Host Identity can be simultaneously reachable through several interfaces. In HIP architecture, TCP connections and UDP associations are no longer bound to IP addresses but to Host Identities through HIT or LSI, Figure 8 illustrates the difference between bindings of logical entities in current Internet architecture and HIP architecture. Thus, HIP decouples the transport layer from the internetworking layer, allowing each to evolve separately. The decoupling makes end system mobility and multi-homing easier and less costly across IPv4 and IPv6 networks.

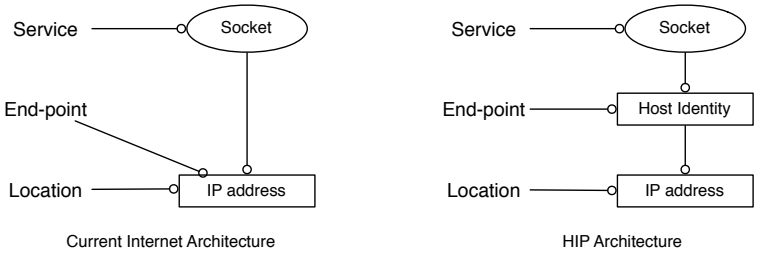

Fig. 8. The difference between the bindings of the logical entities

The Host Identity Tags in HIP has an important security property in that it is *self-certifying*. *Self-certifying* means that the owner of an identifier, whether it is used as an address tag or end-point name, can prove its ownership of the identifier to others without relying on a trust third-party. Since HITs are hash of the public key of a host and only that host has the corresponding private key, the host can prove its ownership of the HIT to another host using a simple challenge-response protocol when the other host requests such a proof.

HIP architecture is effective in preventing IP address spoofing and holding hosts accountable for their actions when used in a closed environment where everyone

knows everyone else's public key. There needs to be some sort of public key infrastructure or a public key to address mapping service. For a very large-scale distributed system that spans a great number of administrative domains, such as Internet, it is very hard to deploy a global PKI. Without being able to securely bind a Host Identifier to the host address, a malicious can essentially mint unlimited number of Host Identifier and use them to hide its true identity.

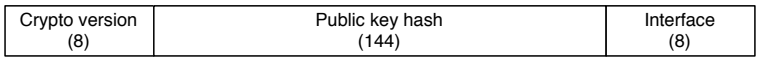

Fig. 9. The structure of an AIP address.

5.2.2 *Accountable Internet Protocol.* Accountable Internet Protocol (AIP) is proposed by Andersen et al. to provide Internet layer accountability using self-certifying addresses [Andersen et al. 2008]. AIP is designed to address the lack of secure binding of a host to its IP addresses, and lack of secure binding of an AS number to the IP prefixes owned by that AS. The AIP design assumes that each autonomous system decomposes its network into one or more *accountability domains (ADs)*, each with a globally unique identifier. Each host is also assigned a unique identifier, and the AIP address of a host currently homed in some AD would have an address of the form AID:EID. AID and EID are both 160-bit long, and the structure of an AID or EID is shown in Figure 9. The 144-bit *public key hash* part of an AID is the hash of the public key of the accountability domain, whereas for an EID, it is the hash of the public key of the corresponding host. To handle the case of a host that attaches multiple times to the same AD, the final 8 bits of the EID are *interface* bits that give each interface a unique identifier. For AID, the interface bits are set to zero. Because AIP uses cryptographic primitives whose strength may degrade over time, each AIP address contains an 8-bit *version number* that indicates what signature scheme incarnation was used to generate the address.

AIP utilizes self-certifying addresses to detect source address spoofing, and uses a network interface implemented *shut-off protocol* to enable receiver to send a signed shut-off messages to a sender from whom it does not want to receive traffic. Assuming the hosts that generate DoS traffic in distributed flooding attacks are compromised hosts, the shut-off scheme that is implemented in the firmware of a host's network interface card (NIC) is immutable by the compromised operating system of that host.

The simplicity and DoS attack prevention effectiveness of AIP make it a very attractive candidate for future generation Internet protocols. However, routing scalability and traffic engineering scalability of AIP's flat addressing scheme for use in the Internet is a very big concern. The scalability demands for Internet addresses resulted in the switch from the classful network addresses, which are similar to flat AIP addresses in hierarchy, to Classless Inter-Domain Routing (CIDR) addresses, and switching back to a more flat addressing scheme surely raises a big scalability question. Morever, since hosts make up their own EID, malicious hosts can create an unlimited number of EIDs and effectively generating unlimited number of identities

that can be used towards flooding the target. Last but not least, recovering from the compromise of a private key that corresponds to the AIP address of a host or an AD is a big hassle. With a compromised key, an attacker could silently impersonate his victim for fairly long time before a victim notices it.

5.3 Secure Overlays

Secure overlay approaches aim to prevent DoS attacks on the limited set of networks they protect, by routing traffic destined to a protected network through an overlay network that is built atop of IP. Since the overlay network only admits authorized users and are carefully designed to provide redundancy and DoS-resistance, it is difficult for attackers to cause DoS on the protected servers or networks. Secure overlay approaches assume that the overlay network is the only way for hosts outside the trusted domain of the protected network to communicate with the protected network. Such isolation of a protected network from rest of the Internet is assumed to be achievable either by hiding the IP addresses of the protected network or by using distributed firewalls to filter all incoming traffic to the protected network except for the traffic only from the trusted nodes in the overlay network.

5.3.1 *SOS.* Secure Overlay Service (SOS) is an overlay network architecture proposed by Keromytis et al. to proactively prevent DoS attacks [Keromytis et al. 2002]. A *secure overlay* is formed by selecting a set of node distributed throughout the wide area network, and are logically linked through secure tunneling. The goal of SOS architecture is to allows communication only between a protected site and a user that is given prior permission to visit that site. To achieve this goal, SOS first assumes a *filtered region* around the protected site that is created by instructing the routers at the perimeter of the site to only allow traffic from a few overlay nodes. These overlay nodes are called *secret servlets*, and are selected by the protected site. The secret servlets compute a set of overlay nodes that will act as *beacons*, and notify the beacons of the fact that they are the secret servlets for the the protected site. A source that wants to communicate with the protected site first contacts an *secure overlay access point* (SOAP), and the SOAP routes the source's traffic to one of the beacons only after authenticating and authorizing the source's communication request. The beacon routes source's packets to a secret servlet that then routes the packets through the filtering router to the target. The routing between SOS overlay nodes are accomplished using Chord service [Stoica et al. 2001]. Figure 10 illustrates the communication between an authorized source and a protected site in the SOS architecture.

SOS is robust against DoS attacks because of the following: 1) If an access point is attacked, the source can simply choose an alternate access point; 2) If a node within the overlay is attacked, the node simply exits the overlay and the Chord service self-heals; 3) If a secret servlet's identity is discovered and the servlet is targeted as an attack point, then the protected site can choose an alternate set of secret servlets.

On the other hand, SOS does not address the general DoS problem in the Internet. SOS is designed to protect few private services only.Moreover, the use of an overlay network by SOS creates a longer and slower route to the destination. The simulation results have shown the latency to be in the order of 10 times larger than in the

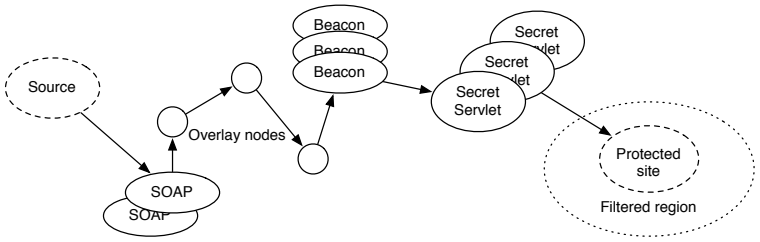

Fig. 10. The communication between an authorized source and a protected site in SOS.

direct communication case [Keromytis et al. 2002]. Lastly, building the secure overlay network requires installing and maintaining additional dedicated nodes in the network, which implies a huge extra cost.

5.3.2 *Secure-i3.* Adkins et al. [Adkins et al. 2003] proposes an overlay network solution called *Secure-i3*, based on the Internet Indirection Infrastructure (*i3*) [Stoica et al. 2002]. *i3* assumes deployment set of special nodes called *i3 nodes* to form an overlay network on top of IP. End to end communication between two hosts is routed within the overlay based on identifiers (rather than IP addresses), and act of sending a packet is decoupled from the act of receiving it. Secure–*i3* utilizes the *i3* overlay network as its means to hide IP addresses of end hosts, and proposes several extensions to give a receiver host the ability to stop receiving traffic from a specific sender.

There are multiple issues with Secure-*i3*. First of all, its entire defense is designed upon the assumption that the IP address of end hosts and portions of *i3* nodes are unknown to the attacker. Although this solution may not exactly qualify as *security through obscurity*, such a heavy reliance secrecy of IP addresses is surely problematic. One should not underestimate the attacker's ability to obtain such information through various footprinting and scanning techniques. Moreover, Secure-*i3* proposes to drop a fraction f of the total traffic destined for a receiver when the receiver is under attacks, hoping that a server under attack will drop a fraction f of the offending traffic by doing so. Such approach surely creates denial of service to the legitimate clients, yet the authors argue it as graceful service degradation. Secure-*i3* requires large number of powerful *i3* nodes with ample bandwidth to be deployed in the Internet, adding a very high extra cost. Since routing in Secure-*i3* overlay network creates extra level of indirection, it increases end-to-end delay and decreases network goodput.

6. DETECTION MECHANISMS

As discussed earlier, detection is an important step before directing further actions to counter a DoS attack. DoS response mechanisms depend on the attack related information discovered by detection mechanisms for countering the attack. Some response mechanisms rely on identification of the malicious actions, while others require identifying the entity that is performing the malicious actions. There are also few mitigation mechanisms that depend on discovering the fact that an attack is ongoing, in order to initiate the mitigation process.

Simply discovering the fact that an attack is taking place is usually easy, since the performance degradation of the attack target is easily noticeable. Except, it is rather difficult to differentiate between flash crowds and subtle DoS attacks. On the other hand, identifying the malicious actions is a very difficult task. Generally, there are two groups of detection techniques for identifying malicious actions: *signature based detection* and *anomaly based detection*. Detection of malicious actions need to be completed quickly, so that response mechanisms have more time to react before serious damages are made. Furthermore, to block malicious actions near their sources or to take legal actions against the attackers, the source of an attack needs to be identified using *attack source identification*. In this section, we look at various signature based detection, anomaly based detection and attack source identification mechanisms.

6.1 Signature Based Detection

Signature based DoS attack detection mechanisms study known DoS attacks to identify unique patterns that differentiates these attacks from normal user activity, and builds a database of known attack patterns. Then they monitor the activity in the network for the presence of these patterns. The drawback of such approach is that only known attacks can be detected, while new attacks or some variations of old attacks go unnoticed.

6.1.1 *Bro.* Bro is a network intrusion detection system (NIDS) for detecting network intruders in real-time by passively monitoring a network link over which the intruder's traffic transits [Paxson 1999]. The main design goals of Bro is high-speed, large volume monitoring, real-time attack notification, separation of mechanisms from policy, and extensibility with regard to new attacks. Bro is conceptually divided into an event engine that reduces a stream of filtered packets to a stream of higher-level network events, and an interpreter for a specialized language that is used to express a site's security policy. The event engine maintains state for each connection based on source IP, source port, destination IP and destination port. Network administrators use the Bro language to write policy scripts.

Bro has a powerful signature language that allows the use of regular expressions, association of traffic going in both directions, and encoding of attacks that comprise multiple stages. The event driven attack detection approach of Bro greatly reduces the number of packets that need to be processed, and helps Bro towards achieving its high-speed traffic monitoring goal. Separation of event generation and event handling helps Bro achieve separation of security mechanisms from policy. Such separation also make it easy to add new protocol analyzers and new event handlers, hence provide good extensibility.

However, similar to many other network intrusion detection systems, Bro is susceptible to evasion attacks and DoS attacks on the monitor. A detailed discussion of various evasion, insertion, and denial of service attack techniques against NIDS is presented in [Ptacek and Newsham 1998]. Bro uses simple four byte XOR to hash values for hash table, and this can be easily exploited by attackers to launch algorithmic complexity DoS attacks on Bro. Moreover, Bro requires manual creation of attack signatures and event handling scripts, which requires meticulous effort and strong intrusion detection background from network administrators.

6.1.2 *MIB Traffic Variable Correlation.* Cabrera et al. investigates how to correlate Management Information Base (MIB) traffic variables that are exchanged through Simple Network Management Protocol (SNMP) to generate statistical signatures of attacking behavior, and how to use such signatures to detect DDoS attacks before they cause damage [Cabrera et al. 2001]. In typical SNMP use, one or more administrative computers called managers have the task of monitoring or managing a group of hosts or devices on a computer network. A SNMP agent that runs on a managed device periodically reports local MIB information to the network management system (NMS), a software that runs on the manager.

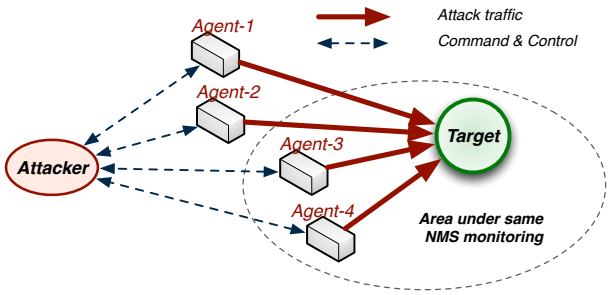

Fig. 11. MIB variable correlation approach assumes that the attack target and some of the attacking agents are under the supervision of the same NMS.

Cabrera DDoS detection approach assumes that the attack target and some of the attack agents are under the supervision of the same NMS [Cabrera et al. 2001], as shown in Figure 11. To detect DDoS attacks, a set of rules or signatures are extracted off-line, and the extracted signatures are matched against monitored traffic online to determine the attack traffic. The signature extraction process includes three steps. First, a set of key MIB variables that characterize the occurrence of an attack are identified from attack target. Next, MIB variables at the agents that are causally related to the key MIB variables from step one are determined through statistical analysis. The assumption here is that any causal relationship between variables at potential attack agents and the key variables at the target is to be inferred as a link between the agent and the target machine. Following the determination of causal correlations between the key MIB variables at the target and key MIB variables at the attack agents, the last step is to determine particular features of the attack agent's key MIB variables that precede the attack. The extracted features that are indicative of an incoming attack is used as signatures.

Cabrera's detection approach may be applicable to mapping TCP, UDP, and ICMP packet statistical abnormalities to specific DDoS attacks. However, the evaluation of the approach in a tightly controlled traffic environment did not show a high accuracy when detecting attacks, and it is unclear how it performs in realistic network environments. Furthermore, this approach assumes that the target and some attack agents are managed by the same NMS, which tremendously limits the applicability of this approach to all parts of the Internet.

6.1.3 *Spectral analysis.* DoS attack flows usually do not follow the regulations of TCP flow control protocols as normal flows do. Consequently, DoS attack flows form different statistical features compared with normal flows. Based on this assumption, Cheng et al. [Cheng et al. 2002] proposed to use spectral analysis to identify DoS attack flows. In this approach, the number of packet arrivals in a fixed interval is used as the signal. In the power spectral density of the signal, a normal TCP flow will exhibit strong periodicity around its round-trip time in both flow directions, whereas an attack flow usually does not.

Hussain, Heidemann, and Papadopoulos [Hussain et al. 2003] developed a framework to identify DoS attacks combining packet header analysis, ramp-up behavior analysis, and spectral analysis. They estimate the number of attackers by counting the number of distinct fragmentation identification field (ID) sequences present in the attack. They argue that distributed activation of attack agents before the attack results in a *ramp-up* of the attack traffic intensity due to the variation in path latency between the attacker and agents and weak synchronization of local clocks between agents. They use this ramp-up behavior to differentiate distributed attacks from single source attacks. Lastly, they claim that attack streams have markedly different spectral content that varies depending on the number of attack agents. Thus, they treat packet traces as a time series, and perform spectral analysis of attack traffic to identify spectral characteristics of single source and distributed attacks. They argue that it is difficult for attackers to conceal their spectrum without reducing attack effectiveness, because the traffic spectrum is influenced by OS and network behavior. Although the proposed framework seem to differentiate between single source and distributed attacks, it is unclear from the paper how accurate it is.

Hussain et al. [Hussain et al. 2006] also presented a method to automatically fingerprint and identify repeated attacks that use a particular combination of attack agent pool and attack tool. Generation of attack fingerprints are based on spectral characteristics of the attack stream. Although the described fingerprinting method seem to generate consistent fingerprints even in the presence of noise introduced by changes in environmental conditions, attackers can still evade the detection by changing the number of attack agent pool or modifying the agent software.

6.1.4 *Others.* Snort [Roesch 1999] is an open-source light-weight network intrusion detection and prevention tool originally developed by Martin Roesch. It is one of the most widely deployed network intrusion detection and prevention systems worldwide. Snort combines the signature based detection with protocol and anomaly based inspection for better coverage of various attack detection.

To automate the attack signature generation process, Kreibich and Crowcroft [Kreibich and Crowcroft 2004] describes a system called Honeycomb for automated generation of attack signatures for network attacks. Honeycomb applies pattern-matching techniques and protocol conformance checks on multiple levels in the protocol hierarchy to network traffic captured by a honeypot system.

In [Kandula et al. 2005], Kandula et al. presents design and implementation of a kernel extension tool called Kill-Bots to protect Web servers against DDoS attacks that masquerade as flash crowds. Kill-Bots attempts to distinguish attack agents from legitimate clients via observing their reaction to CAPTCHA [Ahn et al. 2003]

challenges, assuming attack agents repeatedly send wrong puzzle solutions at a high rate.

Kulkarni and Bush [Kulkarni and Bush 2006] describes an approach to detecting DDoS attacks using Kolmogorov Complexity. A theorem derived using principles of Kolmogorov Complexity states that the joint complexity measure of random strings is lower than the sum of the complexities of the individual strings when the strings exhibit some correlation [Kulkarni and Bush 2006]. The premise is that when the network is in a healthy state, its behavior is highly complex due to many simultaneous independent users, whereas specific attacks will have low complexity. Assuming an attacker performs an attack using large numbers of similar packets sourced from different locations but intended for the same destination, there will be a high degree of similarity in the traffic pattern. A Kolmogorov Complexity based detection algorithm can quickly identify such patterns.

6.2 Anomaly Based Detection

Signature based DoS mechanisms can detect known attacks with relatively high accuracy. However, they fail to achieve such accuracy as malicious behaviors evolve and new attacks appear. Instead of profiling malicious behavior, anomaly based DoS attack detection mechanisms analyze the normal behavior in a system and aims to detect attacks via identifying significant deviation from the normal behavior. Compared to signature based detection approaches, they can discovery previously unseen attacks. Anomaly based approaches faces a challenge, however, when determining the threshold for anomalous behavior. A model that uses a tight threshold for legitimate behavior in the system may wrongly label normal behavior as malicious (false positive), whereas a loose threshold may lead to many attacks go undetected (false negative).

6.2.1 *MULTOPS.* Gil and Poletto proposed MUltiLevel Tree for Online Packet Statistics (MULTOPS) [Gil and Poletto 2001] – a tree data structure that network devices, such as routers, can use to detect bandwidth flooding attacks. MULTOPS tree contains packet rate statistics for subnet prefixes at different aggregation levels, and it expands and contracts within a fixed memory budget. A network device using MULTOPS detects ongoing bandwidth attacks by the significant, disproportional difference between packet rates going to and coming from the attack agent or the attack target.

When the packet rate to or from a subnet reaches a certain threshold, a new sub-node is created in a MULTOPS tree to keep track of more fine-grained packet rates. This process can go till per IP address packet rates are being maintained. Therefore, starting from a coarse granularity one can detect with increasingly finer accuracy, the exact attack source or destination addresses.

However, MULTOPS detection assumes that incoming and outgoing packet rates for a host is proportional, which may not be true for certain traffic types. Furthermore, MULTOPS fails to detect attacks that use randomly spoofed IP addresses to proportionally distribute attack traffic across multiple IP addresses. Lastly, the MULTOPS tree itself may be the target of a memory exhaustion DoS attack. Abdelsayed et al. provides a more memory efficient data structure called Tabulated Online Packet Statistics (TOPS) for detecting packet flow unbalances [Abdelsayed

et al. 2003].

6.2.2 *SYN flood detection.* Wang et al. [Wang et al. 2002] proposed flooding detection system (FDS), a SYN flood detection method based on the protocol behavior of TCP SYN/FIN, SYN/RST pairs. It is assumed that FDS is deployed at the first-mile or last-mile edge routers. SYN flood detection in FDS is based on the fact that a normal TCP connection starts with a SYN and ends with a FIN or RST packet, whereas in SYN floods there will be more SYN packets than FIN and RST packets. FDS models the difference between the number of SYN packets and FIN/RST packets for normal traffic as a stationary ergodic random process, and uses a non-parametric Cumulative Sum (CUSUM) sequential change point detection algorithm described in [Brodsky and Darkhovsky 1993] to detect attack traffic. The drawback of FDS is that an attacker can avoid detection by sending low Time to Live value FIN or RST packets that will be dropped after passing the detection point.

Siris and Papagalou presented and evaluated two anomaly detection algorithms for detecting SYN floods in [Siris and Papagalou 2004]: an adaptive threshold algorithms and a particular application of the CUSUM algorithm for change point detection. They claim that their algorithm achieves a better performance than FDS in terms of false positive.

Kompella, Singh, and Varghese [Kompella et al. 2004] argued that existing attack detection schemes are not scalable for multi-gigabit speeds, and proposed scalably detecting SYN floods using Partial Completion Filter (PCF). A PCF consists of parallel stages each containing hash buckets that are incremented for a SYN and decremented for a FIN. Multiple independent hash functions are employed to compute the hash values, using destination IP address and destination port number as their input. At any point, if all of the buckets that correspond to a particular destination IP and port pair are greater than some threshold, the PCF reports an attack. All of the buckets are set to zero after a fixed measurement interval. One notable property of PCF is that it does not account for connections closed by TCP RSTs. While the majority of TCP connections may close neatly with FINs, there are instances where a host close many connections via RSTs. This leads to PCF overestimating the number of open connections and raising false alarms.

6.2.3 *Others.* Talpade et al. [Talpade et al. 1999] introduced a scalable network monitoring framework called NOMAD, that detects network anomalies through characterization of the dynamic statistical properties of network traffic.NOMAD incorporates a suite of anomaly identification algorithms based on path changes, flow shift, and packet delay variance, and relies extensively on IP packet header information, such as Time to Live, source and destination addresses, packet length, and router timestamps.

Lee and Stolfo [Lee and Stolfo 1998] use data mining techniques to discover patterns of system features that describe program and user behavior and compute a classifier that can recognize anomalies and intrusions. This approach focuses on the host-based intrusion detection. An improvement of this approach is a meta-detection model [Lee et al. 1999], which uses results from multiple models to provide more accurate detection.

A mechanism called congestion triggered packet sampling and filtering has been proposed by Huang and Pullen [Huang and Pullen 2001]. According to this approach, a subset of dropped packets due to congestion is selected for statistical analysis. If an anomaly is indicated by the statistical results, a signal is sent to the router to filter the malicious packets.

Barford et al. [Barford et al. 2002] performed signal analysis of IP flow-level information and SNMP MIB information in order to identify frequency characteristics of DoS attacks and other anomalous network traffic. They developed the concept of deviation score which considers signal variation in both high and low frequency bands, and used it as a means of identifying anomalies. All of their analysis is performed offline, and it is unclear how well their technique works for real-time traffic.

D-WARD [Mirkovic et al. 2002; Mirkovic 2003] is a DoS attack detection and rate-limiting framework. It aims to stop DoS attacks near their sources, thus need to be widely deployed at edge routers of a network. It passively monitors the network traffic and gathers two-way traffic statistics from the edge router at the source network and compares them to network traffic models built upon application and transport protocol specifications. Based on the results of comparison, it classifies traffic into three category: legitimate, suspicious, and attack traffic. Based on this three-tiered model, D-WARD applies rate limits at the edge router on all the outgoing traffic, preferring legitimate traffic, slightly slowing down suspicious traffic, and severely slowing down attack traffic. Rate limits are dynamic and change over time according to the observation of the attack signal and policed traffic aggresiveness. Drawback of D-WARD is that it might classifies legitimate bursts of traffic and legitimate flows that do exhibit asymetry as suspicious or attack traffic, and adversely penalize legitimate senders. Moreover, since deploying D-WARD at a network does not protect that network itself but other networks from DoS, it does not give any deployment incentive to network operators.

In [Feinstein and Schnackenberg 2003], Feinstein et al. propose a statistical mechanism to defend against DDoS attack by analyzing the entropy and calculating the chi-square statistic of IP attributes. The mechanism divides source addresses into a few bins based on their frequency. During detection, the chi-square statistic detection component finds out source addresses which belong to bins in which distributions of frequencies are anomalous. Then, a number of static filtering rules will be set up to filter out packets from these bins. An obvious drawback of the mechanism is that it does not provide good performance on attacks with no spoofed packets. For this kind of attacks, the frequency of source address variation is small and not easily detectable.

Peng et al. [Peng et al. 2004] proposed a source IP address monitoring based detection scheme, called SIM. SIM is deployed at the edge router that provides Internet access to subnetwork in which the target resides. SIM assumes that the set of source IP addresses that is seen during normal operation is somewhat constant, and most of the previously unseen addresses that showed up during an attack belong to attackers. By using a prebuilt database of IP addresses, SIM sequentially monitors the proportion of previously unseen source IP addresses, and detect any abrupt changes using the cumulative sum (CUSUM) sequential change point de-

tection algorithm described in [Brodsky and Darkhovsky 1993]. An abrupt change of the proportion of new source IP addresses is labeled as a strong indication of a DoS attack. They also try to improve the detection accuracy by simultaneously monitoring the traffic rate per IP address.

6.3 Attack Source Identification

Most of the detection techniques presented in previous sections focuses on identifying the malicious traffic in the network, and they are mostly deployed close to the attack target. However, merely differentiating the attack traffic at the target may not be enough for effectively filtering them out. This is because malicious traffic consumes bandwidth of all routers along the attack traffic till near the target where they are filtered. Furthermore, if attack volume is high enough, filtering near the target does not help alleviate the congestion of the links near the target. Setting up filters close to the source addresses found in the attack traffic may not help eliminate the attack or even adversely deny service to legitimate senders, since the source address can be spoofed. Due to the stateless nature of IP routers, it is not possible to acquire information regarding the source of a traffic by querying the target's upstream routers. Attack source identification techniques are proposed to address this problem. These techniques also provide certain level of deterrence, since attackers might be discouraged from launching attack for fear of exposing their location.

6.3.1 *Probabilistic IP traceback.* Savage et al. [Savage et al. 2000] proposed a *probabilistic packet marking (PPM)* scheme for tracing anonymous packet flooding attacks in the Internet back to their source. The main idea of probabilistic packet marking is that each router probabilistically encodes distance-to-receiver and router address information into a marking field in the packet header, and the receiver reconstructs the path that a packet traveled from the encoded information. This technique assumes that attacks generally comprise large numbers of packets. While each marked packet represents only a sample of the path it has traversed, by combining a large number of such packets a victim can reconstruct the entire path. The authors suggest to encode the router address and distance information into the identification field of IP header. The advantage of this traceback scheme is that no extra traffic is generated, since the packet itself carries the extra information. However, using the identification field of IP header for such path encoding purposes conflicts with the semantic interpretation of this field by other Internet protocols, hence raises a backward incompatibility issue. Moreover, reconstruction of a path at the receiver requires a large computation overhead and gives high false positives, as discussed in [Song and Perrig 2001].

To improve probabilistic packet marking, Song and Perrig [Song and Perrig 2001] proposed two new marking schemes: an *Advanced Marking Scheme* that provides a more efficient and accurate attack path construction than PPM, and an *Authenticated Marking Scheme* that ensures the integrity of the marking information. The advanced marking scheme can accurately reconstruct the attack path when there is up 1000 simultaneous attack agents. However it requires the target to have a map of its upstream routers. Although the Authenticated Marking Scheme provides an elegant way of authenticating marking information in a packet using timed-release

key chains, it still requires each router to have a certified public key for signing the commitment of a hash chain. Both marking schemes encode information in the identification field of the IP header, hence, face similar compatibility issue as in the case of probabilistic packet marking by Savage et al.

6.3.2 *ICMP traceback message.* Bellovin [Bellovin and Leech 2000] proposed a new type of ICMP message called *iTrace packet*, to help receiver reconstruct the path that packets take through the Internet. Each router generates a traceback message with a very low probability for each packet it forwards, and sends the message to the receiver. To avoid burdening the receiver with too much extra traffic, the probability of generating the traceback message should be adjustable, and should not be greater than 1/1000. If enough traceback messages are gathered at the receiver, the source of traffic can be found by constructing a chain of traceback messages. The advantage of this scheme is that it is simple and easy to implement. However, it needs digital signatures to protect the integrity of the information contained in a traceback message. Such digital signatures are expensive to generate and verify, and requires a key distribution infrastructure. Moreover, compared with other traceback schemes, this scheme creates extra traffic in the network.

6.3.3 *Single-Packet IP Traceback.* Snoeren et al. [Snoeren et al. 2001] developed Source Path Isolation Engine (SPIE) to identify the source of a particular IP packet, given a copy of the packet to be traced, its destination, and an approximate time of receipt. SPIE requires all routers to keep a hash digest of recently forwarded packets. In order to minimize the required memory space, SPIE uses Bloom filters [Bloom 1970] to store the packet digests. When a traceback is needed, a query is dispatched to SPIE which in turn queries routers for packet digests of the relevant time periods. The results of this query are used in a simulated reverse-path flooding (RPF) algorithm to build an attack graph that indicates the packet's source.

The main advantage of SPIE over PPM and ICMP traceback is that the receiver does not need to receive large number of attack packets to traceback the attack source. However, despite using Bloom filters, SPIE still requires large amount of memory for storing packet digests at routers. For a core router with 32 OC-192 links, 23.4 gigabytes of memory is needed to store one minute's worth of packet digests. Since huge amount of memory is required for storing digests for fairly short period of time, the traceback needs to be performed in a very time-efficient manner.

7. RESPONSE MECHANISMS

Attack detection and attack source identification techniques discussed above aim to isolate the attack traffic and identify the source of the attack in a timely manner, so that further actions can be initiated to counter the attack. Response mechanisms are usually initiated after the detection of an attack to eliminate or minimize the impact of the attack.

7.1 Filtering and Rate-limiting

Filtering and rate-limiting mechanisms use the characterization of malicious traffic that is provided by detection mechanisms to filter out or rate-limit attack flows. Rate-limiting is usually used in cases where detection has many false positives or

cannot precisely characterize the attack traffic. If part or all of the attack flow can be precisely distinguished by detection mechanisms, then they can be filtered out completely. Rate-limiting and filtering may seem very simple to implement at first, but there are challenges in designing effective filtering and rate-limiting mechanisms. For rate-limiting mechanisms, they need to find a good balance between letting some attack traffic through and harming some legitimate traffic. For both mechanisms, if the location where attack traffic is detected is not the optimal location for filtering or rate-limiting them, then there needs to be mechanisms to coordinate the detection and the response at different locations.

7.1.1 *Pushback high-bandwidth aggregates.* Mahajan et al. [Mahajan et al. 2002] proposed aggregate-based congestion control (ACC) for controlling high bandwidth aggregates in the network, where an *aggregate* is a collection of packets from one or more flows that have some property in common. ACC includes a local mechanisms for identifying and controlling an aggregate at a single router, and a cooperative pushback mechanism in which a router can ask upstream routers to control an aggregate. ACC is triggered only when a link experiences sustained severe congestion, which can be determined by looking for an extended high packet loss rate period. After a severe congestion is detected, ACC identifies the aggregates that are responsible for the congestion. However, there are links in the network that are dominated by particular aggregates even in the normal case, and that might remain dominated by them even in the presence of diffuse congestion [Mahajan et al. 2002]. If ISP wants prevent such aggregates from being rate-limited, they can configure rate-limiting policy for ACC to do so. Next, the rate-limit for the identified aggregates are chosen so that a minimum level of service can be guaranteed for the remaining traffic.

To save upstream bandwidth through early dropping of packets that would have been dropped downstream at the congested router, rate-limiting of the aggregate traffic is pushed upstream towards the source of aggregates. This cooperative high-bandwidth aggregate control mechanism is called *pushback*. In pushback, the congested router sends pushback messages to the neighbor routers that send a significant fraction of the aggregate traffic, and asks them to rate-limit the aggregate. The recipient routers can recursively propagate pushback further upstream. Figure 12 shows the propagation of pushback messages towards upstream.

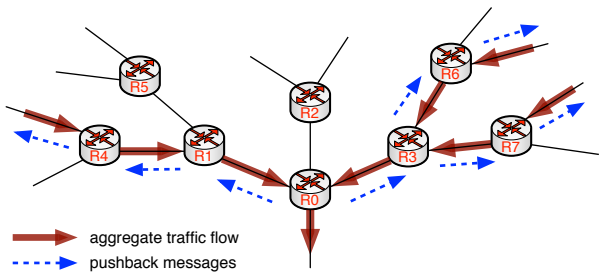

Fig. 12. Rate-limiting the aggregate traffic is propagated upstream towards the source.

Lastly, the rate-limiting decisions in ACC are revisited periodically to revise the limit on the rate-limited aggregates based on the current conditions, and to release some aggregates altogether if they have started to behave. These decisions are easy when the rate-limiting is local. However, they are more difficult when pushback is involved, as the routers must distinguish between not seeing much traffic from the aggregate due to upstream rate-limiting, and the aggregate ceasing to be high-bandwidth. Disambiguating these two requires feedback from upstream routers, and it is implemented by upstream routers sending status messages that contains arrival rate of rate-limited aggregates to the requesting router.

ACC provides pushing attack flows back towards their sources when an end-system or network link discovers it is under attack. Unfortunately to pushback each source in a large distributed DoS attack is likely to be relatively expensive in terms of network state, and requires each source to be identified and pushed back individually. Such identification may be challenging, since it may be hard to tell good traffic from bad if an attack is not too blatant. Moreover, pushback mechanism requires routers to cooperate by sending each other pushback and status messages. These messages need to be authenticated to prevent attackers from sending forged status or pushback messages to routers. Therefore, some sort of key distribution infrastructure needs to be implemented to guarantee authenticity of pushback messages, and such key distribution infrastructure is hard to implement across security domains.

7.1.2 *StopIt.* StopIt is a filter-based DoS defense framework proposed by Liu, Yang, and Lu [Liu et al. 2008]. StopIt aims stop the unwanted traffic destined to a receiver without inflicting damage on legitimate hosts sending traffic to that receiver. Figure 13 depicts the overall architecture of StopIt, and shows how a receiver can send StopIt requests to block a sender from sending it traffic.

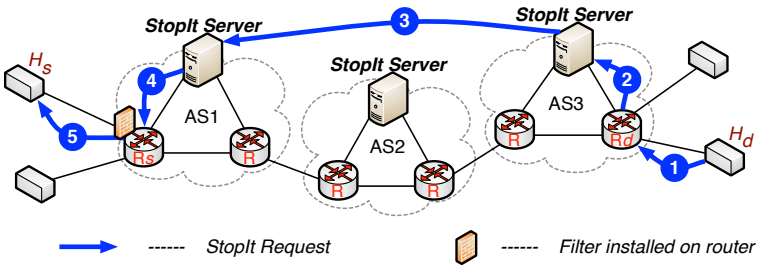

Fig. 13. Destination H_d prevents unwanted traffic from H_s using StopIt request.

As Figure 13 shows, each AS has a *StopIt server* that handles filter requests. A StopIt server learns the addresses of other StopIt servers by listening to Border Gateway Protocol (BGP) [Rekhter et al. 2006] updates. It is assumed that StopIt servers can authenticate each other, and each StopIt server can also authenticate routers within its AS. As illustrated in Figure 13, if a destination host H_d wishes to block an attack flow from the sender host H_s, it sends a host-to-router StopIt

request to its access router R_d, the access router then sends a router-to-server StopIt request to the StopIt server for its AS. The StopIt server then forwards an inter-AS StopIt request to the StopIt server of the AS from which the attack flow is originated. And so, this source StopIt server locates the access router R_s of the attacking host H_s, and sends it a server-to-router StopIt request. The access router R_s sends a router-to-host StopIt request to H_s, and installs a filter that filters out flow from H_s to H_d in case H_s does not comply with R_s's request. All StopIt requests contain a blocking time length, address of the receiver host (H_d), and address of the misbehaving sender (H_s).

In StopIt, a host cannot directly send a StopIt request to the StopIt server, it must send the request to its access router first. Cross AS StopIt requests can only be sent between StopIt servers. The StopIt framework uses a secure source authentication scheme called Passport [Liu et al. 2008] to prevent source IP address spoofing.

To prevent attackers from flooding the routers and StopIt servers with filter requests and packet floods, StopIt framework ensures that a router or a StopIt server only receives StopIt requests from local nodes in the same AS, or another StopIt server. However, doing so requires network administrators to manually configure the routers and StopIt requests with the list of hosts, routers, and other StopIt servers. However, for an AS with tens or hundreds of thousands of nodes, such manual configuration can be onerous. Moreover, StopIt design requires hosts and routers to implement complex StopIt request verification/authentication techniques and misbehaving StopIt server detection mechanisms, which overly complicates the StopIt framework, and makes it challenging to deploy and manage in practice.

7.2 Capability

Some researchers argue that the fundamental problem of the Internet with regard to DoS attacks is that the receiver has not control over who can send how much traffic to it. Even though flow control and congestion control mechanisms are already in place in the current Internet, there is no guarantee that the sender will follow the congestion control and flow control signals by the receiver. A misbehaving sender can simply ignore any congestion control signals and send traffic at the maximum possible rate. Capability based DoS response solutions aim to enable receiver to stop misbehaving senders.

7.2.1 *SIFF.* A Stateless Internet Flow Filter (SIFF) is designed by Yaar et al. to selectively stop unwanted flows from reaching the recipient's network [Yaar et al. 2004]. SIFF assumes two Internet packet classes: *privileged* packets that are subject to receiver's control and *unprivileged* packets that are used in legacy traffic and in SIFF handshake. The SIFF handshake protocol is used by senders to obtain capabilities to send privileged traffic, and it is illustrated in Figure 14. Sender starts the handshake process by sending a capability request packet with its *capability field* initialized to zero. Each router along the path from the sender to the receiver adds its *marking* into the capability field, and the value of the capability field of the arriving packet at the receiver becomes the *capability*. If the receiver wants to allow the sender to send privileged traffic, then it sends this capability back to the sender. Sender includes the capability in subsequent packets that will

be treated as privileged packets by the intermediate routers. Routers use keyed hash of the packet source and destination addresses etc. as the marking, and the keys are rotated frequently to limit the validity time of a capability. When a router receives a privileged packet, it recomputes the marking and compare it with the marking contained in the capability to decide whether to forward the packet.

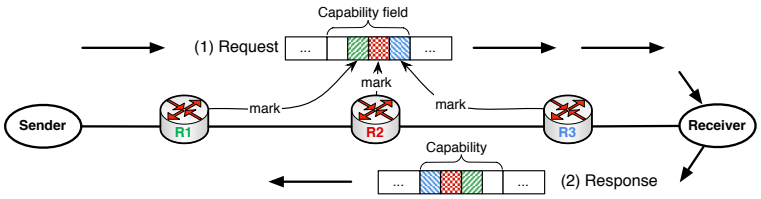

Fig. 14. The process for a sender to acquire a capability to send privileged packets.

Privileged packets are given priority at routers, and majority of the bandwidth is allocated to privileged traffic so that privileged packets are never dropped by unprivileged packet flooding. Flooding with privileged packets is highly unlikely, since the receiver can stop any unwanted flow simply by not sending the capability back to the sender.

SIFF has several advantages, including that it does not require end-host/ISP or inter-ISP cooperation, and requires a very small constant state and constant per-packet processing at routers etc. However, SIFF assumes that the receiver can differentiate the attack traffic from the legitimate traffic. Thus, it needs to be combined with some sort of attack traffic identification method, and ultimately the effectiveness of SIFF DoS defense depends on the accuracy of the attack detection method. Moreover, an attacker can still launch DoS attack by saturating the bandwidth allocated to unprivileged traffic. Since capability request packets are sent as unprivileged packets, saturating the unprivileged traffic bandwidth prevents senders from obtaining capability. In addition, if route between the sender and receiver changes, then a privileged packet will be dropped due to failing to pass the validation at the new router. SIFF also requires all routers to implement the SIFF mechanisms to be effective against DoS attacks, because an attacker can cause denial of service by flooding the bandwidth of the router that does not implement SIFF. Last but not least, SIFF cannot prevent colluding attackers from sending capabilities to each other and flooding a link that is shared by the targeted victim.

7.2.2 *TVA.* Yang et al. proposed Traffic Validation Architecture (TVA), a network architecture that limits the impact of DoS floods [Yang et al. 2005]. The TVA architecture builds on the previously proposed work [Anderson et al. 2003] on capabilities, and tries to address the limitations of previous capability mechanisms such as SIFF. In particular, TVA aims to address the flooding attack against capability setup channel and attacks that flood the receiver using already acquired capabilities. The process for acquiring a capability is similar to that of SIFF (Figure 14). However, the format of capabilities used in the TVA architecture are somewhat different from that of SIFF capabilities.

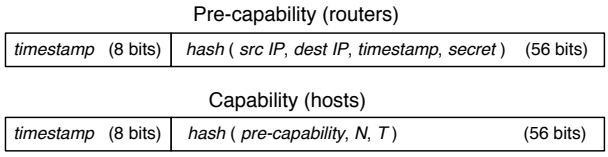

Fig. 15. Format of capabilities used in TVA.

Figure 15 shows the format of capabilities used in TVA. Pre-capability is composed of an 8-bit coarse timestamp that changes every T seconds and a 56-bit hash of source IP, destination IP, timestamp, and a router secret. Each router generates its own pre-capability and attaches it to the capability request packet. When the request packet reaches the receiver, if the destination wishes to authorize the request, it generates one *capability* that corresponds to each pre-capability. This capability is computed by hashing pre-capability plus N and T, where N and T mean that the sender can only send N bytes in total within the next T seconds using the capability. Each router changes the secret used for computing pre-capability at twice the rate of the timestamp rollover, and only uses current or previous secret to validate capability.

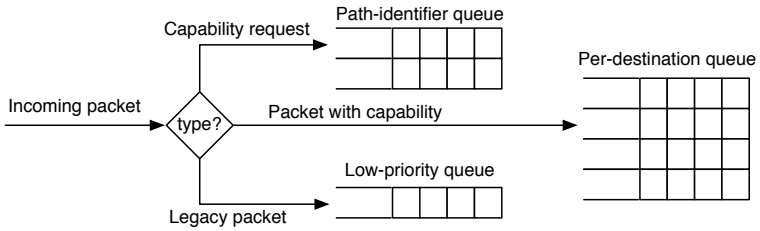

Fig. 16. Queue management at a capability router that implements TVA.

In TVA, long capabilities (64-bit per router) are used to ensure security, and capabilities are cached at routers so that they can subsequently be omitted for bandwidth efficiency. TVA divides traffic into three types: 1) capability requests that are rate-limited, 2) regular packets with associated capabilities that receive preferential forwarding, and 3) legacy traffic that competes for any remaining bandwidth. TVA's queueing mechanisms for these three types of traffic are illustrated in Figure 16. TVA architecture rate-limits capability request traffic to 5% of the capacity of each link. Furthermore, to prevent attacker from flooding this 5% bandwidth, capability requests are fair queued based on most recent path identifier. Path identifier is a 16-bit value derived from the hash of the IP address of the router interface from which the packet is received. Only the edge routers at the ingress of a trust boundary attaches a path identifier to a capability request packet. Downside of fair queueing capability requests based on path identifier is that senders that share the same path identifier share fate.

To prevent colluding attackers flooding a bottleneck link by authorizing high-rate transfers traffic between themselves, packets with capabilities are fair-queued based

on their destination IP addresses. Each router not only check validity of capability, but also verify that capability has not been used for longer than T, and no more than N bytes sent using this capability. Router can minimize the state used for counting amount of traffic, by keeping state only for flows sending at a rate larger than N/T. In the case of route changes, router might not have associated state for a packet. In this case, the router demotes the packet to legacy packet by setting the demotion bit in the header. When destination sees a demoted packet, it informs the sender of the demotion, and the sender knows that it must re-acquire capabilities.

There are several weaknesses of TVA scheme. First of all, as with other capability-based solutions, TVA assumes that the receiver can distinguish the malicious sender from the legitimate ones. Hence, TVA relies on an attack identification method to filter malicious traffic, and its DoS prevention effectiveness depends on the accuracy of the identification method. Furthermore, TVA requires too much per flow state information (flow nonce, N and T values, byte counter) to be maintained at each router. Certain DoS attacks are still possible even if after all routers implement TVA. An attacker sends requests to its colluder spoofing the victim's address, and the colluder returns the list of capabilities to the attacker's real address. The attacker can then flood authorized traffic to the colluder using victim's address. Last but not least, it is possible to forge capabilities in TVA. Recall that a capability in TVA is the hash of N, T, and the pre-capability, which is 56-bit long. An attacker can build a dictionary that contains mappings of all possible 56-bit long data and their hashes, which requires 16 petabytes of storage space. Since, value of T is known and N is fairly easy to guess, an attacker can easily compute pre-capability from a given capability. Using the pre-capability, an attacker can then forge a capability with N and T values of its choice, hence being able to send packets any rate possible. This vulnerability of TVA, however, can be easily fixed by adding a random salt when computing the capability, and returning the capability to the sender together with the salt.

8. TOLERANCE MECHANISMS

Tolerance mechanisms aim to minimize the damage caused by a DoS attack without being able to differentiate malicious behavior from legitimate ones. The obvious advantage of tolerance mechanisms is that they do not rely on detection mechanisms to identify attack, and in some cases they do not even need to know that an attack is happening. This is very helpful where detection of an attack and separating attack traffic or malicious service requests is especially hard, or when the accuracy of detection is low. Tolerance mechanisms recognize the unattainability of complete DoS prevention or detection, and focuses on minimizing the attack impact and maximizing the quality of service provided during the attack.

Existing approaches to DoS attack tolerance can be grouped into several categories. They are congestion policing, fault tolerance, and resource accounting.

8.1 Congestion Policing

As bandwidth flooding DoS attacks are inherently about congestion of a resource, a good congestion policing and control mechanism can alleviate or even eliminate the effects of the bandwidth flooding attacks. We discuss Re-feedback and NetFence to see how congestion policing mechanisms may be able to provide tolerance against

DoS. Aggregate-based Congestion Control by Mahajan et al. [Mahajan et al. 2002], which we discussed earlier, can also be classified as a tolerance mechanism if it is used without combining it with an attack detection method.

8.1.1 *Re-feedback.* Brisco et al. [Briscoe et al. 2005] introduced a receiver-aligned feedback mechanism called *re-feedback*. They argued that existing congestion feedback mechanisms such as Explicit Congestion Notification (ECN) standard [Ramakrishnan et al. 2001] is hard to enforce, in that path congestion is only known at the last egress of an internetwork, but policing is most useful at the first ingress [Briscoe et al. 2005]. They proposed collecting congestion and delay information in packet header fields in a receiver-aligned manner as data traverses a path. Here, *receiver-aligned* means that the value of an IP header field starts with various different values at the sender, but stops at a previously agreed-upon fixed value when the packet reaches the receiver. For example, Time to Live (TTL) is *sender-aligned* in that it always starts with a fixed value such as 255 at the sender. For it to be receiver-aligned, the TTL should arrive at the receiver set to a fixed value, say 16. To achieve receiver-aligned TTL, each receiver needs to occasionally feed back the TTL values it sees, so that the sender can hit the fixed value 16 by adjusting the initial TTL value.

Authors of re-feedback proposed to use TTL for implementing delay re-feedback and use ECN with nonce [Spring et al. 2003] for implementing congestion re-feedback. Once re-feedback is in place, each packet arrives at each network element carrying a congestion view of its own downstream path, although this view is a round trip ago. As a result, full path congestion becomes visible at the first ingress, where a rate policer might be most useful.

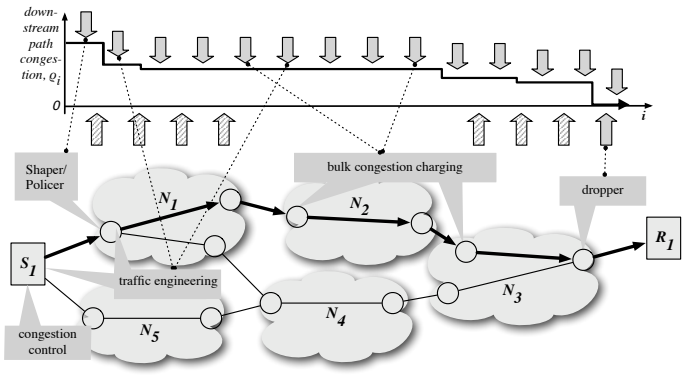

Fig. 17. Overview of the incentive framework envisioned in re-feedback.

However, there is no guarantee that a sender always sets the congestion information truthfully. To solve this problem, re-feedback aim to create an incentive environment in which anyone's selfish behavior leads to maximization of social welfare. Figure 17 illustrates the incentive framework envisioned in re-feedback.

To ensure senders set the congestion information truthfully, a dropper is used at the last network egress. If a sender understates the congestion, the congestion

value becomes negative when the packet reaches the dropper, and the dropper drops packets in flows that persistently understates downstream congestion. The incentive framework also envisions that it is in network operator's best interest to police and shape congestion response of senders. So, it a sender overstates the downstream congestion, then its own network's ingress policer will overly throttle its traffic. Next, the inter-domain congestion charging mechanisms is used to ensure that any network that harbors compromised 'zombie' hosts will have to pay for the congestion that their attacks cause in downstream networks. With inter-domain congestion charging, a network's profit depends on the difference between congestion at its ingress (its revenue) and at its egress (its cost). So overstating internal congestion seems to increase profit. However, smart border routing bias its multipath routing towards the least cost routes, so the cheating network risks losing its revenue to competitive routes if it overstates congestion.

If Re-feedback mechanism and all of its incentive framework is in place, it does help relief network congestion and DDoS attacks to certain degree. However, it is not clear that Re-feedback can help alleviate large-scale DDoS attacks. Other disadvantages of Re-feedback are as follows. First, the ingress policer need to keep per-flow state in order to police sender's congestion response, which lays the policer open to resource depletion DoS attacks. Second, it is challenging for the sender to correctly set the true congestion value, since it may require many trial-and-errors. Making the matter worse, sending understated congestion value increases the risk of discard at the egress dropper, while sending overstated congestion value increases the risk of sanction at the ingress policer. Third, if a DDoS attack that understates the congestion is carried out, then attack traffic can only be dropped at the last egress point which is close to the receiver. Attack traffic still consumes large amount of bandwidth on the path toward the target, and may significantly effect other senders that share the same path. Last, but not least, Re-feedback works well for TCP-like protocols with an acknowledgement (ACK) mechanism. But, it does not work for protocols, such as UDP, that do not have such acknowledgement mechanism.

8.1.2 *NetFence.* NetFence [Liu et al. 2010] is a DoS-resistant network architecture proposed by Liu et al. It uses a secure congestion policing feedback mechanism inside the network to enable robust congestion control. Packets carry unforgeable congestion policing feedback stamped by routers that suffer congestion, and access routers at the trust boundaries between the network and end systems examine the congestion feedback and police the sender's traffic. The unforgeable congestion feedback can be used as capability, similar to the capability in TVA [Yang et al. 2005], SIFF [Yaar et al. 2004], and Portcullis [Parno et al. 2007], by the receiver to control unwanted traffic (by not returning the capability to the sender). NetFence uses each Autonomous System as a fate sharing unit, and proposes to use AS-level queuing and rate-limiting at the AS boundaries to limit DoS damage to the ASes that harbor compromised routers.

Similar to TVA and SIFF, NetFence requires each router to keep three different queues for request packets, regular packets, and legacy packets. A packet without a congestion feedback is treated as a request packet, and the bandwidth allocated to request traffic is limited to 5% of the output link capacity. Regular packets are

given higher priority than the legacy packets. Unlike TVA and SIFF, NetFence requires only access routers, that are located at the trust boundaries between the access network and end systems, and bottleneck routers to perform the congestion feedback generation and congestion policing functions.

NetFence defines three types of congestion feedback: no policing, link overloaded, and link underloaded. Access router of a sending end host inserts 'no policing' or 'link underloaded' feedback into each packet's NetFence header, and this feedback is updated along the path to the receiver by the first router that is experiencing a congestion or an attack to 'link overloaded'. The congestion feedback is then piggybacked to data or ACK packets if a two-way protocol such as TCP is used, or sent back in a separate packet to the sender if one-way transport protocols such as UDP is used. The sender needs to attach this feedback to its packets to avoid them being treated as low priority legacy packets. An access router maintains one rate limiter for each sender-bottleneck pair to limit a sender's regular packet traffic that traverses a bottleneck link. The access router uses Additive Increase and Multiplicative Decrease algorithm to control the rate limit. It either increases the rate limit additively or decreases it multiplicatively, depending on whether the congestion feedback is 'link underloaded' or 'link overloaded'.

NetFence uses cryptographic hash functions to guarantee the integrity of congestion feedback. In order to enable access routers validate the congestion feedback generated by bottleneck routers, NetFence requires symmetric key sharing between ASes by exchanging Diffie-Hellman public keys embedded in BGP update messages. It also uses Passport [Liu et al. 2008] to prevent source IP address spoofing.

The advantage of NetFence over previous congestion policing mechanisms is that it provides unforgeability of congestion feedback. And its advantage over previous capability approaches is that it reduces the need to verify and generate capability (here, the congestion feedback is the capability) only to access routers and bottleneck routers, hence saving significant processing overhead at the routers. Furthermore, compared to Portcullis and TVA's per-host fair-queuing requirement at each router, NetFence significantly reduces the amount of state kept by a bottleneck router by implementing the fair-queuing at access routers.

However, guaranteeing per-sender fairness at the bottleneck router may not be enough to prevent denial of service to legitimate senders, if the number of malicious senders is very large. NetFence need to be combined with detection mechanisms to identify the large portion of the malicious senders and prevent their traffic from ever entering the network. Another concern with the NetFence is that NetFence adds a 28-byte feedback header plus 24-byte Passport [Liu et al. 2008] header for each packets, which is a considerably large overhead. Moreover, the symmetric key sharing between each pair of ASes and distribution of this AS level key to all access routers within an AS is somewhat problematic considering the extra configuration and management overhead that it will incur. Overall, NetFence achieves certain level of improvement over previous congestion policing and capability based DDoS defense approaches. Meanwhile, it does it integrating several complex mechanisms, and significantly increases the complexity of the systems that deploy NetFence, losing the simplicity and elegance of capability based approaches.

8.2 Fault Tolerance

Building fault tolerance into the system is one of the main approaches to achieving high availability. Fault tolerance mechanisms can be implemented at hardware, software, and system levels. In DoS attack mitigation settings, fault tolerance is often achieved by replication of the service or multiplication of the resources used by that service.

Sivasubramanian et al. [Sivasubramanian et al. 2004] studied various replication techniques for Web hosting systems, and identified issues in building a Web replica hosting system. Yan et al. proposed XenoService [Yan et al. 2000], an infrastructure of a distributed network of web hosts that respond to an attack on any website by replicating the website rapidly and widely among XenoService servers, thereby allowing the attacked site to acquire more network connectivity to absorb a packet flood.

The resource multiplication technique, also know as capacity over-provisioning, aim to tolerate the DoS attack by over-provisioning or dynamically increasing the targeted resource of DoS attack. Menth et al. [Menth et al. 2006] argue that admission control methods unnecessarily block excess traffic caused by fluctuations of the traffic rate on a link due to its normal stochastic behavior and redirected traffic due to network failures. The propose a capacity dimensioning method for networks with resilience requirements and changing traffic matrices.

Fault tolerance mechanisms can be very effective against DoS attacks, however they are usually very costly to implement, and resources may be wasted as the over-provisioned resources are not utilized during non-attack period.

8.3 Resource Accounting: Puzzles

Cryptographic puzzle approaches add resiliency to the protected system, as they try to minimize the effects of an attack on legitimate users of a system without being able to identify malicious clients from legitimate ones.

8.3.1 *Client Puzzles.* Dwork and Noar [Dwork and Naor 1992] were the first to introduce the concept of requiring a client to compute a moderately hard but not intractable function, in order to gain access to a shared resource. However this scheme is not suitable for defending against the common form of DoS attack due to its vulnerability to puzzle solution pre-computations.

Juels and Brainard [Juels and Brainard 1999] introduced a hash function based puzzle scheme, called *client puzzles*, to defend against connection depletion attack. Client puzzles addresses the problem of puzzle pre-computation. Aura et al. [Aura et al. 2000] extended the client puzzles to defend DoS attacks against authentication protocols, and Dean and Stubblefield [Dean and Stubblefield 2001] implemented a DoS resistant TLS protocol with the client puzzle extension. Wang and Reiter [Wang and Reiter 2003] further extended the client puzzles to prevention of TCP SYN flooding, by introducing the concept of *puzzle auction*. Price [Price 2003] explored a weakness of the client puzzles and its above mentioned extensions, and provided a fix for the problem by including contribution from the client during puzzle generation.

Waters et al. [Waters et al. 2004] proposed outsourcing of puzzle distribution to an external service called *bastion*, in order to secure puzzle distribution from DoS

attacks. However, the central puzzle distribution can be the single point of failure, and the outsourcing scheme is also vulnerable to the attack introduced by Price [Price 2003].

Wang and Reiter [Wang and Reiter 2004] used a hash-based puzzle scheme to prevent bandwidth-exhaustion attacks at the network layer. Feng et al. [Feng 2003] argued that a puzzle scheme should be placed at the network layer in order to prevent attacks against a wide range of applications and protocols. And Feng and Kaiser et al. [Feng et al. 2005] implemented a hint-based hash reversal puzzle at the IP layer to prevent attackers from thwarting application or transport layer puzzle defense mechanisms.

Portcullis [Parno et al. 2007] by Parno et al. used a puzzle scheme similar to the puzzle auction by Wang [Wang and Reiter 2003] to prevent denial-of-capability attacks that prevent clients from setting up capabilities to send prioritized packets in the network. In Portcullis, clients that are willing to solve harder puzzles that require more computation are given higher priority, thus potentially giving unfair advantage to powerful attackers.

In all of proposals above, finding the puzzle solution is parallelizable. Thus an attacker can obtain the puzzle solution faster by computing it in parallel using multiple machines. Morever, they all suffer from the resource disparity problem, and interferes with the concurrently running user applications. In comparison, guided tour puzzles are non-parallelizable, and addresses the problems of resource disparity and interference with user applications.

8.3.2 *Non-Parallelizable Puzzles.* Non-parallelizable puzzles prevents a DDoS attacker that uses parallel computing with large number of compromised clients to solve puzzles significantly faster than average clients. Rivest at al. [Rivest et al. 1996] designed a *time-lock puzzle* which achieved non-parallelizability due to the lack of known method of parallelizing repeated modular squaring to a large degree [Rivest et al. 1996]. However, time-lock puzzles are not very suitable for DoS defense because of the high cost of puzzle generation and verification at the server.

Ma [Ma 2005] proposed using *hash-chain-reversal puzzles* in the network layer to prevent against DDoS attacks. Hash-chain-reversal puzzles have the property of non-parallelizability, because inverting the digest i in the chain cannot be started until the inversion of the digest $i + 1$ is completed. However, construction and verification of puzzle solution at the server is expensive. Furthermore, using a hash function with shorter digest length does not guarantee the intended computational effort at the client, whereas using a longer hash length makes the puzzle impossible to be solved within a reasonable time.

Another hash chain puzzle is proposed by Groza and Petrica [Groza and Petrica 2006]. Although this hash-chain puzzle provides non-parallelizability, it has several drawbacks. The puzzle construction and verification at the server is relatively expensive, and the transmission of a puzzle to client requires high-bandwidth consumption.

More recently Tritilanunt et al. [Tritilanunt et al. 2007] proposed a puzzle construction based on the subset sum problem, and suggested using an improved version [Coster et al. 1992] of *LLL lattice reduction* algorithm by Lenstra et al. [Lenstra et al. 1982] to compute the solution. However, the subset sum puzzles has prob-

lems such as high memory requirements and the failure of LLL in dealing with large instance and high density problems.

Although the non-parallelizable puzzles addresses one of the weaknesses of client puzzles discussed in Section 8.3.1, they still suffer from the resource disparity problem and interferes with the concurrently running user applications on client machines. Guided tour puzzles, on the other hand, address these two weaknesses of non-parallelizable puzzles.

8.3.3 *Memory-Bound Puzzles.* Abadi et al. [Abadi et al. 2003] argued that memory access speed is more uniform than the CPU speed across different computer systems, and suggested using memory-bound function in puzzles to improve the uniformity of puzzle cost across different systems. Dwork et al. [Dwork et al. 2003] further investigated Abadi's scheme and provided an abstract memory-bound function with an amortized lower bound on the number of memory accesses required for the puzzle solution. Although these results are promising, there are several issues need to be solved regarding memory-bound puzzles.

First, memory-bound puzzles assume a upper-bound on the attacker machine's cache size, which might not hold as technology improves. Increasing this upper-bound based on the maximum cache size available makes the memory-bound puzzles too expensive to compute by average clients. Secondly, deployment of proposed memory-bound puzzle schemes require fine-tuning of various parameters based on a system's cache and memory configurations. Furthermore, puzzle construction in both schemes is expensive, and bandwidth consumption per puzzle transmission is high. Last, but not least, clients without enough memory resources, such as PDAs and cell phones, cannot utilize both puzzle schemes, hence require another service that performs the puzzle computation on their behalf.

9. CONCLUSION

In this article, we studied one of the major security threats in the Internet – denial of service, and provided a comprehensive survey of DoS attacks and their countermeasures. We analyzed various different attacks, developed a more practical classification method for DoS attacks, and provided a taxonomy of DoS attack mechanisms. Our taxonomy distinguishes itself from the existing taxonomy by considering DoS in general and emphasizing practicality.

Furthermore, we analyzed the original design goals of the Internet and how they may have contributed to the challenges of the DoS problem. We then discussed other technical and research challenges in addressing the DoS, and emphasized the importance of understanding these challenges for the design of better DoS solutions.

We also critically reviewed a large number of prominent research proposals in the DoS defense area, analyzed their strengths and weaknesses. We suggested possible improvements to some of the defense solutions, and concluded a comprehensive taxonomy of various defense mechanisms. Compared to existing taxonomies, our taxonomy took into account the latest developments in the DoS field, and is easily applicable to existing DoS solutions.

REFERENCES

Abadi, M., Burrows, M., Manasse, M., and Wobber, T. 2003. Moderately hard, memory-bound functions. In *10th Network and Distributed System Security Symposium*. 25–39.

Abdelsayed, S., Glimsholt, D., Leckie, C., Ryan, S., and Shami, S. 2003. An efficient filter for denial-of-service bandwidth attacks. In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '03)*. Vol. 3. 1353–1357.

Adkins, D., Lakshminarayanan, K., Perrig, A., and Stoica, I. 2003. Towards a more functional and secure network infrastructure. Tech. Rep. UCB/CSD-03-1242, EECS Department, University of California, Berkeley.

Ahn, L. V., Blum, M., Hopper, N. J., and Langford, J. 2003. CAPTCHA: using hard ai problems for security. In *Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques*. EUROCRYPT'03. Springer-Verlag, Berlin, Heidelberg, 294–311.

Andersen, D. G., Balakrishnan, H., Feamster, N., Koponen, T., Moon, D., and Shenker, S. 2008. Accountable internet protocol (AIP). In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*. SIGCOMM '08. 339–350.

Anderson, T., Roscoe, T., and Wetherall, D. 2003. Preventing Internet denial-of-service with capabilities. In *Proceedings of HotNets-II*.

Asosheh, Dr., A. and Ramezani, N. 2008. A comprehensive taxonomy of ddos attacks and defense mechanism applying in a smart classification. *WSEAS Transactions on Computers 7*, 281–290.

Aura, T., Nikander, P., and Leiwo, J. 2000. DoS-resistant authentication with client puzzles. In *8th International Workshop on Security Protocols*. Vol. 2133. 170–181.

Baker, F. 1995. Requirements for IP Version 4 Routers. RFC 1812 (Proposed Standard). Updated by RFC 2644.

Baker, F. 2007. Cisco ip version 4 source guard. Internet-Draft (Informational).

Ballani, H. and Francis, P. 2008. Mitigating dns dos attacks. In *Proceedings of the 15th ACM conference on Computer and communications security*. CCS '08. 189–198.

Barford, P., Kline, J., Plonka, D., and Ron, A. 2002. A signal analysis of network traffic anomalies. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. IMW '02. ACM, New York, NY, USA, 71–82.

Bass, T., Freyre, A., Gruber, D., and Watt, G. 1998. E-mail bombs and countermeasures: Cyber e-mail bombs and countermeasures: Cyber attacks on availability and brand integrity. *IEEE Network 12,* 2 (March), 10–17.

Bellovin, S. M. 1989. Security problems in the TCP/IP protocol suite. *COMPUTER COMMUNICATIONS REVIEW 19,* 2 (mar), 32–48.

Bellovin, S. M. and Leech, M. 2000. ICMP traceback messages. Obsolete Internet draft.

Bishop, M. 2002. *Computer Security: Art and Science*. Addison Wesley, Chapter 1, 3–6.

Bloom, B. H. 1970. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM 13(7)*, 422–426.

Braden, R. 1989. Requirements for Internet Hosts - Communication Layers. RFC 1122 (Standard). Updated by RFCs 1349, 4379, 5884.

Briscoe, B., Jacquet, A., Di Cairano-Gilfedder, C., Salvatori, A., Soppera, A., and Koyabe, M. 2005. Policing congestion response in an internetwork using re-feedback. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*. SIGCOMM '05. ACM, New York, NY, USA, 277–288.

Brodsky, B. and Darkhovsky, B. 1993. *Nonparametric methods in change-point problems*. Mathematics and Its Applications. Kluwer Academic Publishers, Dordrecht, Netherlands.

Burch, H. 2000. Tracing anonymous packets to their approximate source. In *Proceedings of the 14th USENIX conference on System administration*. USENIX Association, 319–328.

Bush, R. and Meyer, D. 2002. Some Internet Architectural Guidelines and Philosophy. RFC 3439 (Informational).

Cabrera, J. B. D., Cabrera, J. A. B. D., Lewis, L., Qin, X., Lee, W., Prasanth, R. K., Ravichandran, B., and Mehra, R. K. 2001. Proactive detection of distributed denial of service attacks using MIB traffic variables - a feasibility study. In *2001 IEEE/IFIP International Symposium on Integrated Network Management Proceedings*. 609–622.

Cain, B., Deering, S., Kouvelas, I., Fenner, B., and Thyagarajan, A. 2002. Internet Group Management Protocol, Version 3. RFC 3376 (Proposed Standard). Updated by RFC 4604.

CERT/CC. 1996a. Denial-of-service attack via ping. CERT Advisory CA-1996-26.

CERT/CC. 1996b. Tcp syn flooding and ip spoofing attacks. CERT Advisory CA-1996-21.

CERT/CC. 1996c. UDP port denial-of-service attack. CERT Advisory CA-1996-01.

CERT/CC. 1997. Denial of service attacks.

CERT/CC. 1998. Smurf IP denial-of-service attacks. CERT Advisory CA-1998-01.

CERT/CC. 2000. Denial of service attacks using nameservers. CERT Incident Note.

CERT/CC. 2009. Cert statistics (historical).

Chang, D.-F., Govindan, R., and Heidemann, J. 2002. An empirical study of router response to large BGP routing table load. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. IMW '02. 203–208.

Cheng, C.-M., Kung, H. T., and Tan, K.-S. 2002. Use of spectral analysis in defense against DoS attacks. In *Proceedings of the IEEE GLOBECOM '02*. Vol. 3. 2143–2148.

Clark, D. D. 1988. The design philosophy of the DARPA internet protocols. In *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*. New York, NY, USA, 106–114.

Coster, M. J., Joux, A., Lamacchia, B. A., Odlyzko, A. M., Schnorr, C., and Stern, J. 1992. Improved low-density subset sum algorithms. *Computational Complexity 2(2)*.

Cotton, M. and Vegoda, L. 2010. Special Use IPv4 Addresses. RFC 5735 (Best Current Practice).

Crosby, S. A. and Wallach, D. S. 2003. Denial of service via algorithmic complexity attacks. In *Proceedings of the 12th conference on USENIX Security Symposium - Volume 12*.

Dagon, D., Gu, G., Lee, C., and Lee, W. 2007. A taxonomy of botnet structures. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*. 325 –339.

Dean, D. and Stubblefield, A. 2001. Using client puzzles to protect TLS. In *10th USENIX Security Symposium*. 1–8.

Diffie, W. and Hellman, M. E. 1976. New directions in cryptography. *IEEE Transactions on Information Theory IT-22,* 6 (November), 644–654.

Dobbins, R. and Morales, C. 2010. Worldwide infrastructure security report. Arbor Networks Annual Survey.

Douligeris, C. and Mitrokotsa, A. 2004. Ddos attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks 44*, 643–666.

Dwork, C., Goldberg, A., and Naor, M. 2003. On memory-bound functions for fighting spam. In *CRYPTO '03*.

Dwork, C. and Naor, M. 1992. Pricing via processing or combatting junk mail. In *CRYPTO '92*. 139–147.

Feinstein, L. and Schnackenberg, D. 2003. Statistical approaches to ddos attack detection and response. In *Proceedings of the 2003 DARPA Information Survivability Conference and Exposition (DISCEX '03)*. Vol. 1. 303–314.

Feng, W. 2003. The case for TCP/IP puzzles. In *ACM SIGCOMM Future Directions in Network Architecture*.

Feng, W., Kaiser, E., and Luu, A. 2005. The design and implementation of network puzzles. In *IEEE INFOCOM '05*. Vol. 4. Miami, FL, 2372–2382.

Ferguson, P. and Senie, D. 2000. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827 (Best Current Practice). Updated by RFC 3704.

Gil, T. M. and Poletto, M. 2001. MULTOPS: a data-structure for bandwidth attack detection. In *Proceedings of the 10th conference on USENIX Security Symposium*. 23–38.

Gligor, V. D. 1984. A note on denial-of-service in operating systems. *Software Engineering, IEEE Transactions on 10,* 3 (may), 320–324.

Gordon, L. A., Loeb, M. P., , Lucyshyn, W., and Richardson, R. 2005. CSI/FBI computer crime and security survey. Annual Report.

Groza, B. and Petrica, D. 2006. On chained cryptographic puzzles. In *3rd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence (SACI '06)*. Timisoara, Romania.

Handley, M., Rescorla, E., and IAB. 2006. Internet Denial-of-Service Considerations. RFC 4732 (Informational).

Huang, Y. and Pullen, J. M. 2001. Countering denial-of-service attacks using congestion triggered packet sampling and filtering. In *Proceedings of tenth International Conference on Computer Communications and Networks*. IEEE, 490–494.

Hussain, A., Heidemann, J., and Papadopoulos, C. 2003. A framework for classifying denial of service attacks. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. SIGCOMM '03. ACM, New York, NY, USA, 99–110.

Hussain, A., Heidemann, J., and Papadopoulos, C. 2006. Identification of repeated denial of service attacks. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM '06)*. IEEE, Barcelona, Spain, 1–15.

Jin, C., Wang, H., and Shin, K. G. 2003. Hop-count filtering: an effective defense against spoofed DDoS traffic. In *Proceedings of the 10th ACM conference on Computer and Communications Security*. CCS '03. 30–41.

Joncheray, L. 1995. A simple active attack against TCP. In *Proceedings of the 5th conference on USENIX UNIX Security Symposium - Volume 5*. 2–2.

Juels, A. and Brainard, J. 1999. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *NDSS '99*. San Diego, CA, 151–165.

Kandula, S., Katabi, D., Jacob, M., and Berger, A. 2005. Botz-4-sale: surviving organized ddos attacks that mimic flash crowds. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*. NSDI '05, vol. 2. USENIX Association, Berkeley, CA, USA, 287–300.

Keromytis, A. D., Misra, V., and Rubenstein, D. 2002. Sos: secure overlay services. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*. SIGCOMM '02. 61–72.

Kompella, R. R., Singh, S., and Varghese, G. 2004. On scalable attack detection in the network. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement (IMC) 2004*. 187–200.

Kreibich, C. and Crowcroft, J. 2004. Honeycomb: creating intrusion detection signatures using honeypots. *SIGCOMM Comput. Commun. Rev. 34*, 51–56.

Kulkarni, A. and Bush, S. 2006. Detecting distributed denial-of-service attacks using kolmogorov complexity metrics. *Journal of Network and Systems Management 14*, 69–80.

Lear, E. and Droms, R. 2003. What's in a name: Thoughts from the NSRG, 2003. draft-irtf-nsrg-report-10. IETF draft (Work in Progress).

Lee, W. and Stolfo, S. J. 1998. Data mining approaches for intrusion detection. In *Proceedings of the 7th conference on USENIX Security Symposium - Volume 7*. USENIX Association, Berkeley, CA, USA, 6–6.

Lee, W., Stolfo, S. J., and Mok, K. W. 1999. A data mining framework for building intrusion detection models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*. 120–132.

Lemos, R. 2001. Web worm targets white house. CNET News. `http://news.cnet.com/2100-1001-270272.html`.

Lenstra, A. K., Lenstra, H. W., and Lovász, L. 1982. Factoring polynomials with rational coefficients. *Mathematische Annalen 261(4)*, 515–534.

Li, J., Mirkovic, J., Wang, M., Reiher, P., and Zhang, L. 2002. SAVE: Source address validity enforcement protocol. In *In Proceedings of IEEE INFOCOM Äô2002*. 1557–1566.

Lipson, H. F. 2002. Tracking and tracing cyber-attacks: Technical challenges and global policy issues. Special report CMU/SEI-2002-SR-009, Cert Coordination Center. November.

Liu, X., Li, A., Yang, X., and Wetherall, D. 2008. Passport: secure and adoptable source authentication. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*. NSDI'08. USENIX Association, Berkeley, CA, USA, 365–378.

Liu, X., Yang, X., and Lu, Y. 2008. To filter or to authorize: network-layer DoS defense against multimillion-node botnets. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*. SIGCOMM '08. ACM, New York, NY, USA, 195–206.

Liu, X., Yang, X., and Xia, Y. 2010. NetFence: preventing internet denial of service from inside out. In *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM*. SIGCOMM '10. ACM, New York, NY, USA, 255–266.

Ma, M. 2005. Mitigating denial of service attacks with password puzzles. In *International Conference on Information Technology: Coding and Computing*. Vol. 2. Las Vegas, 621–626.

Mahajan, R., Bellovin, S. M., Floyd, S., Ioannidis, J., Paxson, V., and Shenker, S. 2002. Controlling high bandwidth aggregates in the network. *SIGCOMM Computer Communication Review 32*, 62–73.

Malkin, G. 1996. Internet Users' Glossary. RFC 1983 (Informational).

McCue, A. 2003. 'Revenge' hack downed US port systems. ZDNet News. `http://www.zdnet.co.uk/news/security-management/2003/10/07/revenge-hack-downed-us-port-systems-39116978/`.

Menth, M., Martin, R., and Charzinski, J. 2006. Capacity overprovisioning for networks with resilience requirements. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*. SIGCOMM '06. ACM, New York, NY, USA, 87–98.

Mirkovic, J. 2003. Phd thesis. Ph.D. thesis, University of California, Los Angeles.

Mirkovic, J., Dietrich, S., Dittrich, D., and Reiher, P. 2004. *Internet Denial of Service: Attack and Defense Mechanisms*. Prentice Hall PTR, Upper Saddle River, NJ, USA, Chapter 1.

Mirkovic, J., Prier, G., and Reiher, P. L. 2002. Attacking DDoS at the source. In *Proceedings of the 10th IEEE International Conference on Network Protocols*. ICNP '02. IEEE Computer Society, Washington, DC, USA, 312–321.

Mirkovic, J. and Reiher, P. 2004. A taxonomy of DDoS attack and DDoS defense mechanisms. *SIGCOMM Comput. Commun. Rev. 34,* 2, 39–53.

Moskowitz, R. and Nikander, P. 2006. Host Identity Protocol (HIP) Architecture. RFC 4423 (Informational).

Moskowitz, R., Nikander, P., Jokela, P., and Henderson, T. 2008. Host Identity Protocol. RFC 5201 (Experimental).

Nordström, O. and Dovrolis, C. 2004. Beware of bgp attacks. *SIGCOMM Computer Communication Review 34*, 1–8.

Park, K. and Lee, H. 2001. On the effectiveness of route-based packet filtering for distributed dos attack prevention in power-law internets. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. SIGCOMM '01. 15–26.

Parno, B., Wendlandt, D., Shi, E., Perrig, A., Maggs, B., and Hu, Y.-C. 2007. Portcullis: protecting connection setup from denial-of-capability attacks. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*. SIGCOMM '07. ACM, New York, NY, USA, 289–300.

Paxson, V. 1999. Bro: a system for detecting network intruders in real-time. *Computer Networks: The International Journal of Computer and Telecommunications Networking 31*, 2435–2463.

Paxson, V. 2001. An analysis of using reflectors for distributed denial-of-service attacks. *SIGCOMM Computer Communication Review 31,* 3, 38–47.

Peng, T., Leckie, C., and Ramamohanarao, K. 2004. Proactively detecting distributed denial of service attacks using source IP address monitoring. In *Proceedings of the Third International IFIP-TC6 Networking Conference (Networking 2004)*. 771–782.

Peters, S. 2009. CSI/FBI computer crime and security survey. Annual Report.

Postel, J. 1980. User Datagram Protocol. RFC 768 (Standard).

Postel, J. 1981a. Internet Control Message Protocol. RFC 792 (Standard). Updated by RFCs 950, 4884.

Postel, J. 1981b. Transmission Control Protocol. RFC 793 (Standard). Updated by RFCs 1122, 3168.

Price, G. 2003. A general attack model on hash-based client puzzles. In *9th IMA Conference on Cryptography and Coding*. Vol. 2898. Cirencester, UK, 319–331.

Ptacek, T. H. and Newsham, T. N. 1998. Insertion, evasion, and denial of service: Eluding network intrusion detection. Tech. rep., Secure Networks Inc. January.

Ramakrishnan, K., Floyd, S., and Black, D. 2001. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard).

Rekhter, Y., Li, T., and Hares, S. 2006. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard).

Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and Lear, E. 1996. Address Allocation for Private Internets. RFC 1918 (Best Current Practice).

Rivest, R. L., Shamir, A., and Wagner, D. A. 1996. Time-lock puzzles and timed-release crypto. Tech. rep., MIT, Cambridge, Massachusetts.

Roesch, M. 1999. Snort - lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX conference on System administration*. LISA '99. USENIX Association, Berkeley, CA, USA, 229–238.

Rose, M. 1991. Convention for defining traps for use with the SNMP. RFC 1215 (Informational).

Saltzer, J. H., Reed, D. P., and Clark, D. D. 1984. End-to-end arguments in system design. *ACM Transactions on Computer Systems 2,* 4, 277–288.

Sandoval, G. and Wolverton, T. 2000. Leading Web sites under attack. CNET News. `http://news.cnet.com/Leading-Web-sites-under-attack/2100-1017_3-236683.html`.

Savage, S., Wetherall, D., Karlin, A., and Anderson, T. 2000. Practical network support for IP traceback. In *Proceedings of ACM SIGCOMM 2000*. ACM, New York, NY, USA, 295–306.

Scarfone, K., Grance, T., and Masone, K. 2008. Computer security incident handling guide. Special Publication 800-61, National Institute of Standards and Technology (NIST). March.

Shirey, R. 2007. Internet Security Glossary, Version 2. RFC 4949 (Informational).

Siris, V. A. and Papagalou, F. 2004. Application of anomaly detection algorithms for detecting SYN flooding attacks. In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '04)*. Vol. 4. 2050–2054.

Sisalem, D., Kuthan, J., and Ehlert, S. 2006. Denial of service attacks targeting a SIP VoIP infrastructure: Attack scenarios and prevention mechanisms. *IEEE IEEE Networks Magazine 20,* 5.

Sivasubramanian, S., Szymaniak, M., Pierre, G., and Steen, M. v. 2004. Replication for web hosting systems. *ACM Computing Surveys 36*, 291–334.

Snoeren, A. C., Partridge, C., Sanchez, L. A., Jones, C. E., Tchakountio, F., Kent, S. T., and Strayer, W. T. 2001. Hash-based IP traceback. In *Proceedings of ACM SIGCOMM 2001*. ACM, New York, NY, USA, 3–14.

Song, D. X. and Perrig, A. 2001. Advanced and authenticated marking schemes for IP traceback. In *Proceedings of IEEE INFOCOM 2001*. Vol. 2. 878–886.

Spring, N., Wetherall, D., and Ely, D. 2003. Robust Explicit Congestion Notification (ECN) Signaling with Nonces. RFC 3540 (Experimental).

Stoica, I., Adkins, D., Zhuang, S., Shenker, S., and Surana, S. 2002. Internet indirection infrastructure. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*. SIGCOMM '02. 73–86.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. 2001. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. SIGCOMM '01. 149–160.

Sullivan, B. 2009. XML denial of service attacks and defenses. *MSDN Magazine*.

Systems, C. 2009. TTL expiry attack identification and mitigation. White paper.

Talpade, R. R., Kim, G., and Khurana, S. 1999. NOMAD: Traffic-based network monitoring framework for anomaly detection. In *Proceedings of the The Fourth IEEE Symposium on Computers and Communications*. IEEE Computer Society, Washington, DC, USA, 442–.

Tritilanunt, S., Boyd, C., Foo, E., and González, J. M. 2007. Toward non-parallelizable client puzzles. In *6h International Conference on Cryptology and Network Security*. 247–264.

US-CERT. 2000. Microsoft IIS 4.0 / 5.0 vulnerable to directory traversal via extended unicode in URL. Vulnerability Note.

VeriSign. 2006. Anatomy of recent dns reflector attacks from the victim and anatomy of recent dns reflector attacks from the victim and reflector point of view. White paper.

Vixie, P., Sneeringer, G., and Schleifer, M. 2002. Events of 21-oct-2002. `http://c.root-servers.org/october21.txt`.

Wang, H., Zhang, D., and Shin, K. G. 2002. Detecting SYN flooding attacks. In *Proceedings of the IEEE INFOCOM 2002*. 1530–1539.

Wang, X. and Reiter, M. K. 2003. Defending against denial-of-service attacks with puzzle auctions. In *IEEE Symposium on Security and Privacy*. Washington DC, 78–92.

Wang, X. and Reiter, M. K. 2004. Mitigating bandwidth-exhaustion attacks using congestion puzzles. In *11th ACM Conference on Computer and Communications Security*. 257–267.

Waters, B., Juels, A., Halderman, J. A., and Felten, E. W. 2004. New client puzzle outsourcing techniques for dos resistance. In *11th ACM CCS*. 246–256.

Yaar, A., Perrig, A., and Song, D. 2004. SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks. In *IEEE Symposium on Security and Privacy*. 130–143.

Yan, J., Early, S., and Anderson, R. 2000. The XenoService - a distributed defeat for distributed denial of service. In *Proceedings of the 3rd Information Survivability Workshop*. ISW 2000.

Yang, X., Wetherall, D., and Anderson, T. 2005. A DoS-limiting network architecture. In *ACM SIGCOMM '05*. 241–252.

Yu, C.-F. and Gligor, V. D. 1988. A formal specification and verification method for the prevention of denial of service. *IEEE Symposium on Security and Privacy*, 187–202.