A Taxonomy of DDoS Attack and DDoS Defense Mechanisms*

Jelena Mirkovic
449 Smith Hall
Computer and Information Sciences Department
University of Delaware
Newark, DE 19716
sunshine@cis.udel.edu

Peter Reiher
3564 Boelter Hall
Computer Science Department
UCLA
Los Angeles, CA 90095
reiher@cs.ucla.edu

ABSTRACT

Distributed denial-of-service (DDoS) is a rapidly growing problem. The multitude and variety of both the attacks and the defense approaches is overwhelming. This paper presents two taxonomies for classifying attacks and defenses, and thus provides researchers with a better understanding of the problem and the current solution space. The attack classification criteria was selected to highlight commonalities and important features of attack strategies, that define challenges and dictate the design of countermeasures. The defense taxonomy classifies the body of existing DDoS defenses based on their design decisions; it then shows how these decisions dictate the advantages and deficiencies of proposed solutions.

1. INTRODUCTION

Distributed denial-of-service (DDoS) attacks pose an immense threat to the Internet, and many defense mechanisms have been proposed to combat the problem. Attackers constantly modify their tools to bypass these security systems, and researchers in turn modify their approaches to handle new attacks. The DDoS field is quickly becoming more and more complex, and has reached the point where it is difficult to see the forest for the trees. On one hand, this hinders an understanding of the DDoS phenomenon. The variety of known attacks creates the impression that the problem space is vast, and hard to explore and address. On the other hand, existing defense systems deploy various strategies to counter the problem, and it is difficult to understand their similarities and differences, assess their effectiveness and cost, and to compare them to each other.

This paper proposes a taxonomy of DDoS attacks and a taxonomy of DDoS defense systems. Together, they structure the DDoS field and facilitate a global view of the problem and solution space. By setting apart and emphasizing crucial features of attack and defense mechanisms, while abstracting detailed differences, these taxonomies can be used by researchers to answer many important questions:

- What are the different ways of perpetrating a DDoS attack? Why is DDoS a difficult problem to handle?
- What attacks have been handled effectively by existing defense systems? What attacks still remain unaddressed and why?
- Given two defense mechanisms, A and B, how would they perform if attack C occurred? What are their vulnerabilities? Can they complement each other and how? Are there some deployment points that are better suited for A than B and vice versa?
- How can I contribute to the DDoS field?

The proposed taxonomies are complete in the following sense: the attack taxonomy covers known attacks and also those which have not yet appeared but are realistic potential threats that would affect current defense mechanisms; the defense system taxonomy covers not only published approaches but also some commercial approaches that are sufficiently documented to be analyzed. Along with classification, we provide representative examples of existing mechanisms.

We do not claim that these taxonomies are as detailed as possible. Many classes could be divided into several deeper levels. Also, new attack and defense mechanisms are likely to appear, thus adding new classes to the ones we propose. Our goal was to select several important features of attack and defense mechanisms that might help researchers design innovative solutions, and to use these features as classification criteria. It was also important not to confuse the reader with a too elaborate and detailed classification. It is our hope that our work will be further extended by other researchers.

We also do not claim that classes divide attacks and defenses in an exclusive manner, i.e. that an instance of an attack or a particular defense system must be classified into a single class based on a given criterion. It is possible for an attack or defense to be comprised of several mechanisms, each of them belonging to a different class.

The depth and width of the proposed taxonomies are not suitable for a traditional numbering of headings – numbers

*This work is funded by DARPA under contract number N66001-01-1-8937.

would quickly become too elaborate to follow. We therefore introduce a customized marking (numbering) of subsection headings in Sections 3 and 5. Each classification criterion is marked abbreviating its name. Attack classes under this criterion are marked by the criterion abbreviation and an arabic number, connected by a dash. To indicate depth of a specific criterion or a class in the taxonomy, the complete mark of a subsection is generated by traversing the taxonomies depicted in Figure 1 and Figure 2, from root to the object in question, concatenating levels with a colon. For example: if an attack classification criterion is *degree of automation*, it will bear the mark *DA*. The second attack class under this criterion, *semi-automatic attacks*, will bear the mark *DA-2*. A classification criterion for *DA-2* class, the *communication mechanism* will bear the mark *DA-2:CM*, and so on.

This paper does not propose or advocate any specific DDoS defense mechanism. Even though some sections might point out vulnerabilities in certain classes of defense systems, our purpose is not to criticize, but to draw attention to these problems so that they might be solved. Following this introduction, Section 2 investigates the problem of DDoS attacks, and Section 3 proposes their taxonomy. Section 4 discusses the DDoS defense challenge, and Section 5 proposes a taxonomy of DDoS defense systems. Section 6 discusses how to use the taxonomies. Section 7 provides an overview of related work, and Section 8 concludes the paper.

2. DDOS ATTACK OVERVIEW

A denial-of-service attack is characterized by an explicit attempt to prevent the legitimate use of a service [14]. A distributed denial-of-service attack deploys multiple attacking entities to attain this goal. This paper is solely concerned with DDoS attacks in the computer realm, perpetrated by causing the victim to receive malicious traffic and suffer some damage as a consequence.

One frequently exercised manner to perform a DDoS attack is for the attacker to send a stream of packets to a victim; this stream consumes some key resource, thus rendering it unavailable to the victim's legitimate clients. Another common approach is for the attacker to send a few malformed packets that confuse an application or a protocol on the victim machine and force it to freeze or reboot. In September 2002 there was an onset of attacks that overloaded the Internet infrastructure rather than targeting specific victims [55]. Yet another possible way to deny service is to subvert machines in a victim network and consume some key resource so that legitimate clients from the same network cannot obtain some inside or outside service. This list is far from exhaustive. It is certain that there are many other ways to deny service on the Internet, some of which we cannot predict, and these will only be discovered after they have been exploited in a large attack.

What makes DDoS attacks possible? Current Internet design focuses on effectiveness in moving packets from the source to the destination. This design follows the *end-to-end paradigm*: the intermediate network provides the bare-minimum, best-effort packet forwarding service, leaving to the sender and the receiver the deployment of advanced protocols to achieve desired service guarantees such as quality of service, reliable and robust transport or security. The end-to-end paradigm pushes the complexity to end hosts, leaving the intermediate network simple and optimized for packet forwarding. There is one unfortunate implication. If one party in two-way communication (sender or receiver) misbehaves, it can do arbitrary damage to its peer. No one in the intermediate network will step in and stop it, because Internet is not designed to police traffic. One consequence of this policy is the presence of IP spoofing [1]. Another are DDoS attacks. The Internet design raises several security issues concerning opportunities for DDoS attacks.

Internet security is highly interdependent. DDoS attacks are commonly launched from systems that are subverted through security-related compromises. Regardless of how well secured the victim system may be, its susceptibility to DDoS attacks depends on the state of security in the rest of the global Internet [21].

Internet resources are limited. Each Internet entity (host, network, service) has limited resources that can be consumed by too many users.

Intelligence and resources are not collocated. An end-to-end communication paradigm led to storing most of the intelligence needed for service guarantees with end hosts, limiting the amount of processing in the intermediate network so that packets could be forwarded quickly and at minimal cost. At the same time, a desire for large throughput led to the design of high bandwidth pathways in the intermediate network, while the end networks invested in only as much bandwidth as they thought they might need. Thus, malicious clients can misuse the abundant resources of the unwitting intermediate network for delivery of numerous messages to a less provisioned victim.

Accountability is not enforced. IP spoofing gives attackers a powerful mechanism to escape accountability for their actions, and sometimes even the means to perpetrate attacks (reflector attacks[2] [59], such as the Smurf attack [19]).

Control is distributed. Internet management is distributed, and each network is run according to local policies defined by its owners. The implications of this are many. There is no way to enforce global deployment of a particular security mechanism or security policy, and due to privacy concerns, it is often impossible to investigate cross-network traffic behavior.

How are DDoS attacks performed? A DDoS attack is carried out in several phases. The attacker first *recruits* multiple agent machines. This process is usually performed automatically through scanning of remote machines, looking for security holes that will enable subversion. The discovered vulnerability is then *exploited* to break into recruited machines and *infect* them with the attack code. The exploit/infect phase is frequently automated, and the infected machines can be used for further recruitment of new agents. Another recruit/exploit/infect strategy consists of distributing attack software under disguise of a useful application (these software copies are called Trojans). This distribution can be performed, for instance, by sending E-mail messages with infected attachments. Subverted agent machines are *used* to send the attack packets. Attackers often hide the identity of subverted machines during the attack through spoofing of the source address field in attack packets. Note,

[1]Putting a fake source address in a packet's header to hide the sender's identity

[2]During reflector attacks, the attacker fakes the source address of the victim in legitimate service requests directed at several servers, e.g., DNS requests. The servers then reply to the victim, overwhelming it.

however, that spoofing is not always required for a successful DDoS attack. With the exception of reflector attacks [59], all other attack types use spoofing only to hinder attack detection and characterization, and the discovery of agent machines.

Why do people perpetrate DDoS attacks? The main goal is to inflict damage on the victim. Frequently the ulterior motives are personal reasons (a significant number of DDoS attacks are perpetrated against home computers, presumably for purposes of revenge), or prestige (successful attacks on popular Web servers gain the respect of the hacker community). However, some DDoS attacks are performed for material gain (damaging a competitor's resources or blackmailing companies) or for political reasons (a country at war could perpetrate attacks against its enemy's critical resources, potentially enlisting a significant portion of the entire country's computing power for this action). In some cases, the true victim of the attack might not be the actual target of the attack packets, but others who rely on the target's correct operation.

3. TAXONOMY OF DDOS ATTACKS

In order to devise a taxonomy of DDoS attacks, we observe the means used to prepare and perform the attack (recruit, exploit and infect phases), the characteristics of the attack itself (use phase) and the effect it has on the victim. Figure 1 summarizes the taxonomy. In the remainder of this section we discuss each of the proposed criteria and classes.

DA: Degree of Automation

Each of the recruit, exploit, infect and use phases can be performed manually or can be automated. Based on the degree of automation, we differentiate between *manual*, *semi-automatic* and *automatic* DDoS attacks.

DA-1: Manual

The attacker manually scans remote machines for vulnerabilities, breaks into them, installs attack code, and then commands the onset of the attack. Only the early DDoS attacks belonged to the manual category. All of the recruitment actions were soon automated.

DA-2: Semi-Automatic

In semi-automatic attacks, the DDoS network consists of handler (master) and agent (slave, daemon, zombie) machines. The recruit, exploit and infect phases are automated. In the use phase, the attacker specifies the attack type, onset, duration and the victim via the handler to agents, who send packets to the victim.

DA-2:CM: Communication Mechanism

Based on the communication mechanism deployed between agent and handler machines, we divide semi-automatic attacks into *attacks with direct communication* and *attacks with indirect communication.*

DA-2:CM-1: Direct Communication

During attacks with direct communication, the agent and handler machines need to know each other's identity in order to communicate. This is usually achieved by hard-coding the IP address of the handler machines in the attack code that is later installed at the agent machine. Each agent then reports its readiness to the handlers, who store its IP address for later communication. The obvious drawback of this approach for the attacker is that discovery of one compromised machine can expose the whole DDoS network. Also, since agents and handlers listen to network connections, they are identifiable by network scanners.

DA-2:CM-2: Indirect Communication

Attacks with indirect communication use some legitimate communication service to synchronize agent actions. Recent attacks have used IRC (Internet chat program) channels [21]. The use of IRC services replaces the function of a handler, since the IRC channel offers sufficient anonymity to the attacker. The agents do not actively listen to network connections (which means they cannot be discovered by scanners), but instead use the legitimate IRC service and their control packets cannot be easily differentiated from legitimate chat traffic. The discovery of a single agent may lead no further than the identification of one or more IRC servers and channel names used by the DDoS network. From there, identification of the DDoS network depends on the ability to track agents currently connected to the IRC server (which may be the subset of all subverted machines). To further avoid discovery, attackers frequently deploy channel-hopping, using any given IRC channel for short periods of time. Since IRC service is maintained in a distributed manner, and the IRC server hosting a particular IRC channel may be located anywhere in the world, this hinders investigation. Although the IRC service is the only known example of indirect communication so far, there is nothing to prevent attackers from subverting other legitimate services for similar purposes.

DA-3: Automatic

Automatic DDoS attacks automate the use phase in addition to the recruit, exploit and infect phases, and thus avoid the need for any communication between attacker and agent machines. The start time of the attack, attack type, duration and victim are preprogrammed in the attack code. Deployment mechanisms of this attack class offer minimal exposure to the attacker, since he is only involved in issuing a single command — at the start of the recruitment process. The hardcoded attack specification suggests a single-purpose use of the DDoS network, or the inflexible nature of the system. However, the propagation mechanisms usually leave a backdoor to the compromised machine open, enabling easy future access and modification of the attack code. Further, if agents communicate through IRC channels, these channels can be used to modify the existing code [56].

DA-2 and DA-3:HSS: Host Scanning and Vulnerability Scanning Strategy

Both semi-automatic and automatic attacks recruit the agent machines by deploying automatic scanning and propagation techniques, usually through use of worms or Trojans. About 3 million scans per day were reported in [75], which aggregated and analyzed firewall logs from over 1600 networks. 20% to 60% of these scans are Web server vulnerability scans and are linked to worm propagation attempts. The goal of the *host scanning strategy* is to chose addresses of potentially vulnerable machines to scan. The *vulnerability scanning* then goes through chosen list of addresses and probes for vulnerabilities. Based on the host scanning strategy,

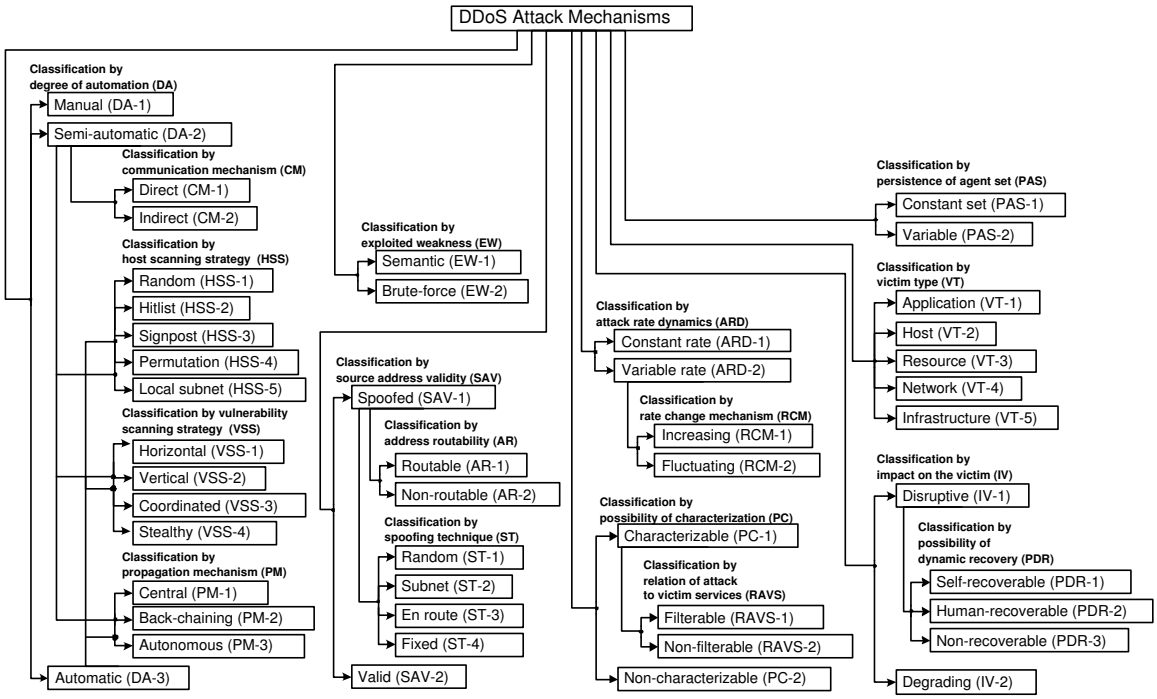

Figure 1: Taxonomy of DDoS Attack Mechanisms

we differentiate between attacks that deploy *random scanning*, *hitlist scanning*, *signpost scanning*, *permutation scanning* and *local subnet scanning*, using material presented in [72, 70]. Based on the vulnerability scanning strategy, we differentiate between attacks that deploy *horizontal scanning*, *vertical scanning*, *coordinated scanning* and *stealthy scanning*, using material presented in [75, 69]. Attackers usually combine the scanning and exploit phases and our description of scanning techniques relates to this model.

DA-2 and DA-3:HSS-1: Random Scanning

During random scanning, each compromised host probes random addresses in the IP address space[3], using a different seed. Code Red (CRv2) performed random scanning [53]. Random scanning potentially creates a high traffic volume. Since many scanned addresses are likely to be in different networks, there is a high amount of internetwork traffic. Also, as infection reaches saturation point (a high percentage of vulnerable machines are infected), duplicate probes to the same addresses escalate, as there is no synchronization of scanning attempts from different infected hosts. The high traffic volume can lead to attack detection.

DA-2 and DA-3:HSS-2: Hitlist Scanning

A machine performing hitlist scanning probes all addresses from an externally supplied list. When it detects a vulnerable machine, it sends a portion of the initial hitlist to the recipient and keeps the rest. Hitlist scanning allows for great propagation speed and no collisions during the scanning phase. If an attacker compiled a list of all vulnerable Internet machines (*flash worm* [70]) he could infect entire population within 30 seconds. The disadvantage to the attacker is that the hitlist needs to be assembled in advance. The information is collected through some inconspicuous means such as using public information on machines running vulnerable software (e.g., found at `netscan.org`) or using slow scans over several months or years. Another disadvantage is that the portion of the hitlist needs to be transmitted to machines that are being infected. If the list is too large, this traffic might be of high volume and lead to attack detection; if it is too small, it will generate a small agent population.

[3]Some researchers have pointed out that this scanning is only effective in densely populated IPv4 address space, and would be less successful in a vast, sparsely populated IPv6 address space.

DA-2 and DA-3:HSS-3: Signpost Scanning

Signpost scanning (also called topological scanning in [72, 70]) takes advantage of habitual communication patterns of the compromised host to select new targets. E-mail worms use signpost scanning, exploiting the information from address books of compromised machines for their spread. A Web-server-based worm could spread by infecting each vulnerable Web browser of clients that click on the server's Web page, and then further infecting servers of subsequent Web pages visited by these clients (this worm is called *contagion worm* in [70]). Signpost scanning does not generate a high traffic load and thus reduces chances of attack detection. The drawback is that the spreading speed depends on agent machines and their user behavior, i.e. it is not controllable by the attacker. The recruitment thus may be slower and less complete than with other scanning techniques.

DA-2 and DA-3:HSS-4: Permutation Scanning

During permutation scanning, all compromised machines share a common pseudo-random permutation of the IP address space; each IP address is mapped to an index in this permutation. Permutation scanning is preceded by small

hitlist scanning during which an initial population of agents is formed. A machine infected during this initial phase begins scanning through the permutation by using the index computed from its IP address as a starting point. A machine infected by permutation scanning always starts from a random point in the permutation. Whenever a scanning host sees an already-infected machine, it chooses a new random starting point. After encountering some threshold number of infected machines, a worm copy becomes dormant to minimize collisions. Permutation scanning has the effect of providing a semi-coordinated, comprehensive scan and minimizing duplication of effort. This technique is described in [72, 70] as not yet seen in the wild. The analysis provided there shows that the spreading speed could be on the order of 15 minutes if 10,000 entry hitlist is used and a worm generates 100 scans per second. As infection progresses, a large percentage of infected machines become dormant. This results in a very low number of duplicated scans and hinders detection.

DA-2 and DA-3:HSS-5: Local Subnet Scanning

Local subnet scanning can be added to any of the previously described techniques to preferentially scan for targets that reside on the same subnet as the compromised host. Using this technique, a single copy of the scanning program can compromise many vulnerable machines behind a firewall. The Code Red II [13] and the Nimda Worm [17] used local subnet scanning.

DA-2 and DA-3:VSS-1: Horizontal Scanning

This is the common type of the scan for worms. Scanning machines are looking for a specific vulnerability, scanning the same destination port on all machines from the list, assembled through host scanning techniques.

DA-2 and DA-3:VSS-2: Vertical Scanning

This is the common type of the scan for intrusions and multiple vector worms. Scanning machines probe multiple ports at a single destination, looking for any way to break in.

DA-2 and DA-3:VSS-3: Coordinated Scanning

This is the common type of the scan for intrusions and worms that favor local subnet scanning. Scanning machines are probing the same destination port(s) at multiple machines within a same local subnet (usually a /24 subnet). This yields an effective goal of looking for a specific vulnerability within a given network.

DA-2 and DA-3:VSS-4: Stealthy Scanning

Any of the above scans (horizontal, vertical or coordinated) could be performed slowly, over a long time period, to avoid detection by intrusion detection systems.

DA-2 and DA-3:PM: Propagation Mechanism

After the recruit and exploit phases, the agent machine is infected with the attack code. Based on the attack code propagation mechanism during the infect phase, we differentiate between attacks that deploy *central source propagation*, *back-chaining propagation* and *autonomous propagation*, using the material presented in [21].

DA-2 and DA-3:PM-1: Central Source Propagation

During central source propagation, the attack code resides on a central server or set of servers. After compromise of the agent machine, the code is downloaded from the central source. The 1i0n [16] worm operated in this manner. Central source propagation imposes a large burden on a central server, generating high traffic and possibly leading to attack discovery. The central server is also a single point of failure; its removal prohibits further agent infection.

DA-2 and DA-3:PM-2: Back-Chaining Propagation

During back-chaining propagation, the attack code is downloaded from the machine that was used to exploit the system. The infected machine then becomes the source for the next propagation step. The Ramen worm [18] and Morris Worm [33] used back-chaining propagation. Back-chaining propagation is more survivable than central-source propagation since it avoids a single point of failure (central server).

DA-2 and DA-3:PM-3: Autonomous Propagation

Autonomous propagation avoids the file retrieval step by injecting attack instructions directly into the target host during the exploit phase. Code Red [12], the Warhol Worm [72] and numerous E-mail worms use autonomous propagation. Autonomous propagation reduces the frequency of network traffic needed for agent mobilization, and thus further reduces chances of attack discovery.

Note that one could easily imagine an attack that would not fall into any of the proposed manual, semi-automatic and automatic classes. For instance, just the recruit and use phases of the attack could be automated, and the exploit and infect phases could be performed manually. Generating classes to accommodate all combinations of automated and non-automated phases would introduce unnecessary complexity since most of these attacks are not likely to occur. We therefore limited our attention to known and probable combinations.

EW: Exploited Weakness to Deny Service

DDoS attacks exploit different weaknesses to deny the service of the victim to its clients. We differentiate between *semantic* and *brute-force* attacks (also called *vulnerability* and *flooding* attacks in the literature).

EW-1: Semantic

Semantic attacks exploit a specific feature or implementation bug of some protocol or application installed at the victim in order to consume excess amounts of its resources. For example, in the TCP SYN attack, the exploited feature is the allocation of substantial space in a connection queue immediately upon receipt of a TCP SYN request. The attacker initiates multiple connections that are never completed, thus filling up the connection queue. In the CGI request attack, the attacker consumes the CPU time of the victim by issuing multiple CGI requests. The NAPTHA [61] attack is an especially powerful attack on the TCP protocol. It initiates and establishes numerous TCP connections that consume the connection queue at the victim. NAPTHA bypasses the TCP protocol stack on the agent machine, not keeping the state for connections it originates. Instead, it participates in the connection, inferring reply attributes from received packets. Thus even a poorly provisioned agent machine can

easily deplete the resources of better provisioned victim.

EW-2: Brute-Force

Brute-force attacks are performed by initiating a vast amount of seemingly legitimate transactions. Since an intermediate network can usually deliver higher traffic volume than the victim network can handle, a high number of attack packets exhausts the victim's resources.

There is a thin line between semantic and brute-force attacks. Semantic attacks also overwhelm a victim's resources with excess traffic, and badly designed protocol features at remote hosts are frequently used to perform reflector brute-force attacks, such as the DNS request attack [15] or the Smurf attack [19]. The difference is that the victim can usually substantially mitigate the effect of semantic attacks by modifying the misused protocols or deploying filters. However, it is helpless against brute-force attacks due to their misuse of legitimate services (filtering attempts also harm legitimate traffic) or due to its own limited resources (a victim cannot handle an attack that swamps its network bandwidth). Countering semantic attacks by modifying the deployed protocol or application pushes the corresponding attack mechanism into the brute-force category. For example, if the victim deploys TCP SYN cookies [20] to combat TCP SYN attacks, it will still be vulnerable to TCP SYN attacks that generate more requests than its network can accommodate. Classification of the specific attack needs to take into account both the attack mechanisms used, and the victim's configuration and deployed protocols.

It should be noted that brute-force attacks need to generate a much higher volume of attack packets than semantic attacks to inflict damage to the victim. So by modifying the deployed protocols, the victim pushes its vulnerability limit higher. It is also interesting to note that the variability of attack packet features (header and payload) is determined by the exploited weakness. Packets comprising semantic and some brute-force attacks must specify some valid header fields and possibly some valid contents. For example TCP SYN attack packets cannot vary the protocol or flag field, and HTTP flood packets must belong to an established TCP connection and therefore cannot spoof source addresses. On the other hand, attacks aiming simply to consume the victim's network resources could vary packet features at will, which greatly hinders the defense.

SAV: Source Address Validity

IP spoofing plays an important role in DDoS attacks. If it were eliminated, reflector attacks [59] would be impossible and many other kinds of DDoS attacks could be solved through resource management techniques – giving the fair share of host or network resources to each source IP address. Based on the source address validity, we distinguish between *spoofed source address* and *valid source address* attacks.

SAV-1: Spoofed Source Address

This is the prevalent type of attack since it is always to attacker's advantage to spoof the source address, avoid accountability, and possibly create more noise for detection.

SAV-1:AR: Address Routability

Based on the address routability we differentiate between *routable source address* and *non-routable source address* attacks.

SAV-1:AR-1: Routable Source Address

Attacks that spoof routable addresses take over the IP address of another machine. This is sometimes done not to avoid accountability, but to perform a reflector attack on the machine whose address was hijacked. One example of a reflector attack is a Smurf attack [19].

SAV-1:AR-2: Non-Routable Source Address

Attackers can spoof non-routable source addresses, some of which can belong to a reserved set of addresses (such as 192.168.0.0/16) or be part of an assigned but not used address space of some network. Attack packets carrying reserved addresses can be easily detected and discarded, while those packets carrying non-used addresses would be significantly harder to detect. Spoofing of non-used addresses, however, enables researchers to gain valuable insight into DDoS attacks, using backscatter technique [54].

SAV-1:ST: Spoofing Technique

Spoofing technique defines how the attacker chooses the spoofed source address in its attack packets. Based on the spoofing technique, we divide spoofing attacks into *random*, *subnet*, *en route* and *fixed* spoofed source address attacks.

SAV-1:ST-1: Random Spoofed Source Address

Many attacks spoof random source addresses in the attack packets, since this can simply be achieved by generating random 32-bit numbers and stamping packets with them. Attempts to prevent spoofing using ingress filtering [30] and route-based filtering [45, 58] force attackers to devise more sophisticated techniques, such as subnet and en route spoofing that can avoid current defense approaches.

SAV-1:ST-2: Subnet Spoofed Source Address

In subnet spoofing, the attacker spoofs a random address from the address space assigned to the agent machine's subnet. For example, a machine which is part of 131.179.192.0/24 network could spoof any address in the range 131.179.192.0 - 131.179.192.255. Since machines at a subnet share the medium (Ethernet) to reach the exit router (first hop en route to the outside world), spoofing can be detected by this router using fairly complicated techniques. It is impossible to detect it anywhere between the exit router and the victim. Subnet spoofing is useful for the attackers that compromise machines on networks running ingress filtering [30]. Such networks would easily detect and filter randomly spoofed packets. Subnet-spoofed packets bypass this filtering while still effectively hiding the address of the compromised machine.

SAV-1:ST-3: En Route Spoofed Source Address

An en route spoofed source address attack would spoof the address of a machine or subnet that lies along the path from the agent machine to the victim. There have not been any known instances of attacks that use en route spoofing, but this potential spoofing technique could affect route-based filtering [45, 58] and is thus discussed here.

SAV-1:ST-4: Fixed Spoofed Source Address

Attacker performing a reflector attack or wishing to place a blame for the attack on several specific machines would use

fixed spoofing. Packets carry a source address chosen from a given list.

SAV-2: Valid Source Address

Attackers benefit from source address spoofing and are likely to deploy it whenever possible. Valid source address attacks frequently originate from agent machines running Windows, since all Windows versions prior to XP do not export user-level functions for packet header modification. Those attacks that target specific applications or protocol features must use valid source addresses if the attack mechanism requires several request/reply exchanges between an agent and the victim machine. One example of such an attack is NAPTHA [61].

ARD: Attack Rate Dynamics

During the attack, each participating agent machine sends a stream of packets to the victim. Depending on the attack rate dynamics of an agent machine, we differentiate between *constant rate* and *variable rate* attacks.

ARD-1: Constant Rate

The majority of known attacks deploy a constant rate mechanism. After the onset is commanded, agent machines generate attack packets at a steady rate, usually as many as their resources permit. The sudden packet flood disrupts the victim's services quickly. This approach provides the best cost-effectiveness to the attacker since he can deploy a minimal number of agents to inflict the damage. On the other hand, the large, continuous traffic stream can be detected as anomalous and arouse suspicion in the network hosting an agent machine, thus facilitating attack discovery.

ARD-2: Variable Rate

Variable rate attacks vary the attack rate of an agent machine to delay or avoid detection and response.

ARD-2:RCM: Rate Change Mechanism

Based on the rate change mechanism, we differentiate between *increasing rate* and *fluctuating rate* attacks.

ARD-2:RCM-1: Increasing Rate

Attacks that have a gradually increasing rate lead to a slow exhaustion of the victim's resources. A victim's services could degrade slowly over a long time period, thus substantially delaying detection of the attack. Plus, these attacks could manipulate defenses that train their baseline models (see section AL-2:ADS-2:NBS-2)

ARD-2:RCM-2: Fluctuating Rate

Attacks that have a fluctuating rate adjust the attack rate based on the victim's behavior or preprogrammed timing, occasionally relieving the effect to avoid detection. For example, during a *pulsing attack*, agents periodically abort the attack and resume it at a later time. If this behavior is simultaneous for all agents, the victim experiences periodic service disruptions. If, however, agents are divided into groups that coordinate so that one group is always active, then the victim experiences continuous denial-of-service while the network hosting agent machine may not notice any prolonged anomaly.

PC: Possibility of Characterization

Looking at the content and header fields of attack packets, it is sometimes possible to characterize the attack. Characterization then may lead to devising of filtering rules. Based on the possibility of characterization, we differentiate between *characterizable* and *non-characterizable* attacks.

PC-1: Characterizable

Characterizable attacks are those that target specific protocols or applications at the victim and can be identified by a combination of IP header and transport protocol header values, or maybe even packet contents. Examples include the TCP SYN attack (only packets with SYN bit set in the TCP header can potentially be part of the attack), ICMP ECHO attack, DNS request attack, etc.

PC-1:RAVS: Relation of Attack to Victim Services

Characterizable attacks are further divided, based on the relation of attack to victim services, into *filterable* and *non-filterable* attacks.

PC-1:RAVS-1: Filterable

Filterable attacks are those that use malformed packets or packets for non-critical services of the victim's operation. These can thus be filtered by a firewall. Examples of such attacks are a UDP flood attack or an ICMP ECHO flood attack on a Web server. Since a Web server only needs TCP traffic and some DNS traffic (which can be characterized as permitting only those inbound UDP packets that are DNS replies to previous outbound DNS requests), it can easily block all other inbound UDP traffic and all ICMP traffic, and still operate correctly.

PC-1:RAVS-2: Non-Filterable

Non-filterable attacks use well-formed packets that request legitimate (and critical) services from the victim. Thus, filtering all packets that match the attack characterization would lead to an immediate denial of the specified service to both attackers and legitimate clients. Examples are HTTP requests flooding a Web server or a DNS request flood targeting a name server. In the case of non-filterable attacks, the contents of an attack packet are indistinguishable from the contents of packets originating from a legitimate client.

PC-2: Non-Characterizable

Non-characterizable attacks attempt to consume network bandwidth using a variety of packets that engage different applications and protocols. Sometimes packets will even be randomly generated using reserved protocol numbers.

Note that classification of attack as characterizable or not depends strongly on the resources that can be dedicated to characterization and the level of characterization. For instance, an attack using a mixture of TCP SYN, TCP ACK, ICMP ECHO, ICMP ECHO REPLY and UDP packets would probably be characterizable, but only after considerable effort and time, and only if one had access to a sophisticated characterization tool. Also, an attack using a mixture of TCP packets with various combinations of TCP header fields can be characterized as a TCP attack, but finer characterization would probably be costly. So, when performing classification of attacks into characterizable or non-characterizable, a lot is left to interpretation, and ease

of characterization should be taken into account.

PAS: Persistence of Agent Set

Recently there were occurrences of attacks that varied the set of agent machines active at any one time, avoiding detection and hindering traceback. We regard this technique as important since it invalidates assumptions underlying many defense mechanisms – that agents are active throughout the attack and can thus be traced back following the path of the attack traffic. We divide attacks, based on the persistence of the agent set, into attacks with *constant agent set* and attacks with *variable agent set*.

PAS-1: Constant Agent Set

During attacks with a constant agent set, all agent machines act in a similar manner, taking into account resource constraints. They receive the same set of commands and are engaged simultaneously during the attack. Examples are an attack in which all agents start sending attack traffic simultaneously,[4] or they engage in a pulsing attack but the "on" and "off" periods for pulses match over all agent machines.

PAS-2: Variable Agent Set

During attacks with a variable agent set, the attacker divides all available agents into several groups and engages only one group of agents at any one time — like the army general who deploys his battalions at different times and places. A machine could belong to more than one group, and groups could be engaged again after a period of inactivity. One example attack of the variable agent set type is an attack in which several agent groups take turns pulsing, thus flooding the victim with a constant flow of packets.

VT: Victim Type

As discussed briefly in Section 2, attacks need not be perpetrated against a single host machine. Depending on the type of victim, we differentiate between *application*, *host*, *resource*, *network* and *infrastructure* attacks.

VT-1: Application

Application attacks target a given application on the victim host, thus disabling legitimate client use of that application and possibly tying up resources of the host machine. If the shared resources of the host machine are not completely consumed, other applications and services should still be accessible to the users. For example, a bogus signature attack on an authentication server ties up resources of the signature verification application, but the target machine will still reply to ICMP ECHO requests, and other applications that do not require authenticated access should still work.[5]

Detection of application attacks is challenging because other applications on the attacked host continue their operations undisturbed, and the attack volume is usually small enough not to appear anomalous. The attack packets are virtually indistinguishable from legitimate packets at the transport level (and frequently at the application level), and the semantics of the targeted application must be heavily used for detection. Since there are typically many applications on a host machine, each application would have to be modelled in the defense system and then its operation monitored to account for possible attacks. Once detection is performed, the host machine has sufficient resources to defend against these small volume attacks, provided that it can separate packets that are legitimate from those that are part of the attack.

[4]The definition of a "simultaneous start" is somewhat relaxed in this context since the attacker's command travels to the agents with a variable delay. Further, because agent machines are under different loads they do not start sending at the exact same moment. This lack of perfect synchronization was used in [37] to gain valuable insight into number of agents used in some observed attacks.

[5]This example assumes that CPU time is shared in a fair manner between all active applications.

VT-2: Host

Host attacks disable access to the target machine completely by overloading or disabling its communication mechanism or making a host crash, freeze or reboot. An example of this attack is a TCP SYN attack [20]. All attack packets carry the destination address of the target host. If protocols running on the host are properly patched, the host attacks likely to be perpetrated against it are reduced to attacks that consume network resources. The high packet volume of such attacks facilitates detection, but the host cannot defend against these attacks alone. It must usually request help from some upstream machine (e.g., an upstream firewall).

VT-3: Resource Attacks

Resource attacks target a critical resource in the victim's network such as a specific DNS server, a router or a bottleneck link. The paths of the attack packets merge before or at the target resource and may diverge afterwards. These attacks can usually be prevented by replicating critical services and designing a robust network topology.

VT-4: Network Attacks

Network attacks consume the incoming bandwidth of a target network with attack packets whose destination address can be chosen from the target network's address space. These attacks can deploy various packets (since it is volume and not content that matters) and are easily detected due to their high volume. The victim network must request help from upstream networks for defense since it cannot handle the attack volume itself.

VT-5: Infrastructure

Infrastructure attacks target some distributed service that is crucial for global Internet operation. Examples include the attacks on domain name servers [55], large core routers, routing protocols, certificate servers, etc. The key feature of these attacks is not the mechanism they deploy to disable the target (e.g., from the point of view of a single attacked core router, the attack can still be regarded as a host attack), but the simultaneity of the attack on multiple instances of a critical service in the Internet infrastructure. Infrastructure attacks can only be countered through the coordinated action of multiple Internet participants.

IV: Impact on the Victim

Depending on the impact of a DDoS attack on the victim, we differentiate between *disruptive* and *degrading* attacks.

IV-1: Disruptive

The goal of disruptive attacks is to completely deny the victim's service to its clients. All currently reported attacks

belong to this category.

IV-1:PDR: Possibility of Dynamic Recovery

Depending on the possibility of dynamic recovery during or after the attack, we differentiate between *self-recoverable*, *human-recoverable* and *non-recoverable* attacks.

IV-1:PDR-1: Self-Recoverable

In the case of self-recoverable attacks, the victim recovers without any human intervention, as soon as the influx of attack packets has stopped. For example, if the attack was a UDP flooding attack, tying up the victim's network resources, the victim will be able to use these resources when the attack stops. A prompt defense could make these attacks transparent to the legitimate clients.

IV-1:PDR-2: Human-Recoverable

A victim of a human-recoverable attack requires human intervention (e.g., rebooting the victim machine or reconfiguring it) for recovery, after the attack is stopped. For example, an attack that causes the victim machine to crash, freeze or reboot would be classified as a human-recoverable attack.

IV-1:PDR-3: Non-Recoverable

Non-recoverable attacks inflict permanent damage to victim's hardware. A new piece of hardware must be purchased for recovery. While non-recoverable attacks are conceivable, the authors have no reliable information that they have occurred in practice.

IV-2: Degrading

The goal of degrading attacks is to consume some (presumably constant) portion of a victim's resources, seriously degrading service to legitimate customers. Since these attacks do not lead to total service disruption, they could remain undetected for a long time. On the other hand, damage inflicted on the victim's business could be immense. For example, an attack that effectively ties up 30% of the victim's resources would lead to a denial-of-service to some percentage of customers during high load periods, and possibly slower average service. Some customers, dissatisfied with the quality, would consequently change their service provider, and the attack victim would lose income. Alternately, the false load could result in the victim spending money to upgrade its servers and networks. The addition of new resources could easily be countered by the attacker through more powerful attacks. Almost all existing proposals to counter DDoS attacks would fail to address degrading attacks.

4. DDOS DEFENSE CHALLENGE

The seriousness of the DDoS problem and the increased frequency, sophistication and strength of attacks have led to the proposal of numerous defense mechanisms. Yet, although many solutions have been developed, the problem is hardly tackled, let alone solved. Why is this so? There are several serious factors that hinder the advance of DDoS defense research.

Need for a distributed response at many points on the Internet. The previous sections have elaborated on the fact that there are many possible DDoS attacks, very few of which can be handled only by the victim. It is frequently necessary to have a distributed, possibly coordinated response system. It is also crucial that the response be deployed at many points on the Internet to cover diverse choices of agents and victims. Since the Internet is administered in a distributed manner, wide deployment of any defense system or even cooperation between networks cannot be enforced or guaranteed. This discourages many researchers from even designing distributed solutions.

Economic and social factors. A distributed response system must be deployed by parties that do not suffer direct damage from the DDoS attack (source or intermediate networks). This implies an unusual economic model since parties that will sustain the deployment cost are not the parties that directly benefit from the system. Similar problems, such as the Tragedy of the Commons [34], have been handled historically through legislative measures, and it is possible that the DDoS problem will eventually attract sufficient attention of lawmakers to invoke a legislative response. Until then, it is possible that many good distributed solutions will achieve only sparse deployment and will thus have a very limited effect.

Lack of detailed attack information. It is necessary to thoroughly understand DDoS attacks in order to design imaginative solutions for them. While there exist publicly available analyses of popular DDoS attack tools [27, 28, 29, 64], what is lacking is the information on frequency of various attack types (e.g., UDP floods, TCP SYN floods), and the distribution of the attack parameters such as rate, duration of the attack, packet size, number of agent machines, attempted response and its effectiveness, damages suffered, etc. It is generally believed that publicly reporting attacks damages the business reputation of the victim network. Attacks are therefore reported only to government organizations under obligation to keep the details secret.

Some notable efforts were made by researchers to infer necessary information from packet traces gathered at their organizations. CAIDA's backscatter packet analysis technique [54] provides a valuable insight into frequency, duration and rate distribution of various attack types. This information was gathered by observing reply packets send to an unused IP address range at CAIDA. ISI/USC attack traffic analysis [37] provided information about the likely number of agents involved in several attacks observed in Los Nettos packet traces. Both of these efforts are a step in the right direction, but they still reveal only a small part of the total picture.

Lack of defense system benchmarks. Many vendors and researchers make bold claims that their solution completely handles the DDoS problem. There is currently no benchmark suite of attack scenarios or established evaluation methodology that would enable comparison between defense systems. Such a situation is likely to discourage networks from investing in DDoS protection, since they cannot be assured of the quality of the product being purchased.

Difficulty of large-scale testing. DDoS defenses need to be tested in a realistic environment. This is currently impossible due to the lack of large-scale testbeds, safe ways to perform live distributed experiments across the Internet, or detailed and realistic simulation tools that can support several thousands of nodes. Claims about defense system performance are thus made based on small-scale experiments and simulations, and are not credible. This situation will likely change in the near future. The US National Science

Foundation is currently funding development of a large-scale cybersecurity test bed and has sponsored research efforts to design benchmarking suites and measurement methodology for security systems evaluation. We expect that this will greatly improve DDoS defense research.

5. TAXONOMY OF DDOS DEFENSES

The proposed taxonomy of DDoS defenses is shown in Figure 2. The remainder of this section will discuss each of the proposed classes of defense mechanisms.

AL: Activity Level

Based on the activity level of DDoS defense mechanisms, we differentiate between *preventive* and *reactive* mechanisms.

AL-1: Preventive

Preventive mechanisms attempt either to eliminate the possibility of DDoS attacks altogether or to enable potential victims to endure the attack without denying services to legitimate clients.

AL-1:PG: Prevention Goal

According to the prevention goal, we further divide preventive mechanisms into *attack prevention* and *denial-of-service prevention* mechanisms.

AL-1:PG-1: Attack Prevention

Attack prevention mechanisms modify systems and protocols on the Internet to eliminate the possibility of subversion or of performing a DDoS attack. The history of computer security suggests that a prevention approach can never be 100% effective, since global deployment cannot be guaranteed. However, doing a good job here will certainly decrease the frequency and strength of DDoS attacks. Deploying comprehensive prevention mechanisms can make a host resilient to known protocol attacks. Also, these approaches are inherently compatible with and complementary to all other defense approaches.

AL-1:PG-1:ST: Secured Target

Based on the secured target, we further divide attack prevention mechanisms into *system security* and *protocol security* mechanisms.

AL-1:PG-1:ST-1: System Security

System security mechanisms increase the overall Internet security, guarding against illegitimate accesses to a machine, removing application bugs and updating protocol installations to prevent intrusions and misuse. DDoS attacks are successful due to large numbers of subverted machines that cooperatively generate attack streams. If these machines were secured, the attackers would lose their army, and the DDoS threat would lessen. Prevention also benefits the deploying machines, guarding them from the intrusions and potential damage. Examples of system security mechanisms include monitored access to the machine [71], applications that download and install security patches, firewall systems [49], intrusion detection systems [6], and worm defense systems [73].

AL-1:PG-1:ST-2: Protocol Security

Protocol security mechanisms address the problem of a bad protocol design. For example, many protocols contain operations that are cheap for the client but expensive for the server. Such protocols can be misused to exhaust the resources of a server by initiating large numbers of simultaneous transactions. Classic misuse examples are the TCP SYN attack, the authentication server attack, and the fragmented packet attack (in which the attacker bombards the victim with malformed packet fragments, forcing it to waste its resources on reassembly attempts), etc. IP source address spoofing is another example: the design of current routing protocols offers no ability for enforcement of valid source addresses. Examples of protocol security mechanisms include guidelines for a safe protocol design in which resources are committed to the client only after sufficient authentication is done [44, 50], or the client has paid a sufficient price [5], deployment of a powerful proxy server that completes TCP connections [63], TCP SYN cookies [11], protocol scrubbing that removes ambiguities from protocols that can be misused for attacks [47], approaches that eliminate spoofing [58, 45, 30], etc.

AL-1:PG-2: DoS Prevention

Denial-of-service prevention mechanisms enable the victim to endure attack attempts without denying service to legitimate clients. This is done either by enforcing policies for resource consumption or by ensuring that abundant resources exist so that legitimate clients will not be affected by the attack.

AL-1:PG-2:PM: Prevention Method

Based on the prevention method, we divide DoS prevention mechanisms into *resource accounting* and *resource multiplication* mechanisms.

AL-1:PG-2:PM-1: Resource Accounting

Resource accounting mechanisms police the access of each user to resources based on the privileges of the user and his behavior. The user in this case might be a process, a person, an IP address, or a set of IP addresses having something in common. Resource accounting mechanisms guarantee fair service to legitimate well-behaved users. In order to avoid user identity theft, they are usually coupled with legitimacy-based access mechanisms that verify the user's identity. Approaches proposed in [40, 76, 68, 31, 43] illustrate resource accounting mechanisms. On the extreme end, systems that test client legitimacy and allow access to the system only to legitimate clients belong in this category [22, 42, 1, 57, 3].

AL-1:PG-2:PM-2: Resource Multiplication

Resource multiplication mechanisms provide an abundance of resources to counter DDoS threats. The straightforward example is a system that deploys a pool of servers with a load balancer and installs high bandwidth links between itself and upstream routers. This approach essentially raises the bar on how many machines must participate in an attack to be effective. While not providing perfect protection, for those who can afford the costs this approach has often proved sufficient.

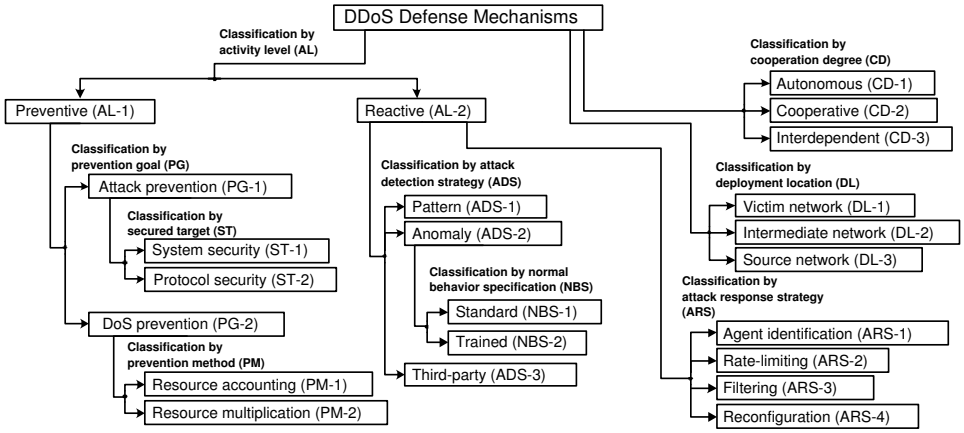

Figure 2: Taxonomy of DDoS Defense Mechanisms

AL-2: Reactive

Reactive mechanisms strive to alleviate the impact of an attack on the victim. To attain this goal they need to detect the attack and respond to it. The goal is to detect every attempted DDoS attack as early as possible and to have a low degree of false positives. Steps then can be taken to characterize the attack packets and provide this characterization to the response mechanism.

AL-2:ADS: Attack Detection Strategy

We classify reactive mechanisms based on the attack detection strategy into mechanisms that deploy *pattern detection*, *anomaly detection*, and *third-party detection*.

AL-2:ADS-1: Pattern Detection

Mechanisms that deploy pattern detection store the signatures of known attacks in a database and monitor each communication for the presence of these patterns. The obvious drawback is that only known attacks can be detected, whereas new attacks or even slight variations of old attacks go unnoticed. On the other hand, known attacks are easily and reliably detected, and no false positives are encountered. Snort [67] provides one example of a DDoS defense system that uses pattern attack detection. A similar approach has been helpful in controlling computer viruses. Like in virus detection programs, signature databases must be regularly updated to account for new attacks.

AL-2:ADS-2: Anomaly Detection

Mechanisms that deploy anomaly detection have a model of normal system behavior, such as normal traffic dynamics or expected system performance. The current state of the system is periodically compared with the models to detect anomalies. Approaches presented in [74, 46, 38, 32, 23, 48, 4, 8, 9, 52, 51, 7] provide examples of anomaly detection approaches. The advantage of anomaly detection over pattern detection is that previously unknown attacks can be discovered. The caveat is that anomaly detectors must trade off their ability to detect all attacks against their tendency to misidentify normal behavior as an attack.

AL-2:ADS-2:NBS: Normal Behavior Specification

Based on the specification of a normal behavior, we divide anomaly detection mechanisms into *standard* and *trained* mechanisms.

AL-2:ADS-2:NBS-1: Standard

Mechanisms that use standard specifications of normal behavior rely on some protocol standard or a set of rules. For example, the TCP protocol specification describes a three-way handshake that has to be performed for TCP connection setup. An attack detection mechanism can make use of this specification to detect half-open TCP connections and delete them from the queue. The advantage of a standard-based specification is that it generates no false positives; all legitimate traffic must comply to the specified behavior. The disadvantage is that attackers can still perform sophisticated attacks which, on the surface, seem compliant to the standard and thus pass undetected.

AL-2:ADS-2:NBS-2: Trained

Mechanisms that use trained specifications of normal behavior monitor network traffic and system behavior and generate threshold values for different parameters. All communications exceeding one or more (depending on the approach) of these values are regarded as anomalous. Trained models catch a broad range of attacks, but have two disadvantages. (1) *Threshold setting.* Anomalies are detected when the current system state differs from the model by a certain threshold. The setting of a low threshold leads to many false positives, while a high threshold reduces the sensitivity of the detection mechanism. (2) *Model update.* Systems and communication patterns evolve with time, and models need to be updated to reflect this change. Trained specification systems usually perform automatic model update using statistics gathered at a time when no attack was detected. This approach makes the detection mechanism vulnerable to slowly increasing rate attacks that can, over a long period of time, mistrain models and delay or even avoid attack detection.

AL-2:ADS-3: Third-Party Detection

Mechanisms that deploy third-party detection do not handle the detection process themselves, but rely on an external

message that signals the occurrence of the attack and provides attack characterization. Examples of third-party detection are easily found among traceback mechanisms [10, 62, 25, 66, 65].

AL-2:ARS: Attack Response Strategy

The goal of the attack response is to relieve the impact of the attack on the victim while imposing minimal collateral damage to legitimate clients. We classify reactive mechanisms, based on the response strategy, into mechanisms that deploy *agent identification*, *rate-limiting*, *filtering* and *reconfiguration*.

AL-2:ARS-1: Agent Identification

Agent identification mechanisms provide the victim with (somewhat accurate) information about the identity of the machines that are performing the attack. This information can be used by other approaches to alleviate the impact of the attack. Agent identification examples include numerous traceback techniques [10, 62, 25, 66, 65]. One frequently mentioned motivation for deployment of DDoS defenses far from the victim network is a possible enforcement of liability for attack traffic. A mechanism for reliable and accurate agent identification would be necessary for liability enforcement.

AL-2:ARS-2: Rate-Limiting

Rate-limiting mechanisms impose a rate limit on a set of packets that have been characterized as malicious by the detection mechanism. It is a lenient response technique that is usually deployed when the detection mechanism has many false positives or cannot precisely characterize the attack stream. The disadvantage is that rate limiting will allow some attack traffic through, so extremely high-scale attacks might still be effective even if all traffic streams are rate-limited. Examples of rate-limiting mechanisms are found in [46, 32, 23, 52, 51, 4].

AL-2:ARS-3: Filtering

Filtering mechanisms use the characterization provided by detection mechanisms to filter out the attack streams completely. Examples include dynamically deployed firewalls [24], and some commercial systems [48, 4]. Unless the characterization is very accurate, filtering mechanisms run the risk of accidentally denying service to legitimate traffic. Worse, clever attackers might leverage them as denial-of-service tools.

AL-2:ARS-4: Reconfiguration

Reconfiguration mechanisms change the topology of the victim or the intermediate network to either add more resources to the victim or to isolate the attack machines. Examples include reconfigurable overlay networks [2, 38], resource replication services [74], attack isolation strategies [8], etc.

CD: Cooperation Degree

DDoS defense mechanisms can perform defensive measures either alone or in cooperation with other entities in the Internet. Based on the cooperation degree, we differentiate between *autonomous*, *cooperative* and *interdependent* mechanisms.

CD-1: Autonomous

Autonomous mechanisms perform independent defense at the point where they are deployed (a host or a network). Firewalls and intrusion detection systems provide easy examples of autonomous mechanisms. Even if a defense system performs its function in a distributed manner it would still be considered autonomous if it can be completely deployed within the network it protects (e.g., a network intrusion detection system).

CD-2: Cooperative

Cooperative mechanisms are capable of autonomous detection and response, but can cooperate with other entities and frequently have significantly better performance in joint operation. The aggregate congestion control (ACC) system [46] deploying a pushback mechanism [39] provides an example. ACC can autonomously detect the attack, characterize the offending traffic and act locally to impose a rate limit on that traffic. However, it achieves significantly better performance if the rate-limit requests can be propagated to upstream routers that otherwise may be unaware of the attack.

CD-3: Interdependent

Interdependent mechanisms cannot operate autonomously at a single deployment point. They either require deployment at multiple networks, or rely on other entities for attack prevention, attack detection or for efficient response. For example, a traceback mechanism [10, 62, 25, 66, 65] deployed at a single router would provide no benefit. Secure overlay services [42] are another example of an interdependent mechanism, which needs deployment at multiple networks and modification of client software.

DL: Deployment Location

With regard to deployment location, we differentiate between mechanisms deployed at the *victim*, *intermediate*, or *source network*.

DL-1: Victim Network

DDoS defense mechanisms deployed at the victim network protect this network from DDoS attacks and respond to detected attacks by alleviating the impact on the victim. Historically, most defense systems are located at the victim since it suffers the greatest impact of the attack and is therefore the most motivated to deploy (and bear the cost of) DDoS defense. Resource accounting [40, 76, 68, 31, 43] and protocol security mechanisms [44, 50, 5, 63] provide examples of victim network defenses.

DL-2: Intermediate Network

DDoS defense mechanisms deployed at the intermediate network provide infrastructural DDoS defense service to a large number of Internet hosts. Victims of DDoS attacks can contact the infrastructure and request the service, possibly providing adequate compensation. Pushback [46] and traceback [10, 62, 25, 66, 65] techniques are examples of intermediate-network mechanisms. Such mechanisms are not yet widely deployed, and many of them can only be effective in wide deployment.

DL-3: Source Network

The goal of DDoS defense mechanisms deployed at the source network is to prevent network customers from generating DDoS attacks. Such mechanisms are desirable, but motivation for their deployment is low since it is unclear who would pay the expenses associated with this service. [32, 23, 52, 51] provide examples of source-network defenses.

6. USING THE TAXONOMIES

In designing the above taxonomies, we selected those features of attack and defense mechanisms that, in our opinion, offer critical information regarding seriousness and type of threats, and effectiveness and cost of defenses. How can these taxonomies be used?

A map of DDoS research field. For novice researchers, these taxonomies offer a comprehensive overview for a quick introduction to the DDoS field. Experienced researchers can use and extend these taxonomies to structure and organize their knowledge in the field. This should lead to identification of new directions for DDoS research and improve understanding of the threat.

Exploring new attack strategies. In addition to known threats, the attack taxonomy explored a few strategies seen rarely in the wild (e.g., variable agent set, variable rate attacks), and some novel attack methods (e.g., increasing rate attacks). As usual suspects get handled by defense systems, these exotic attacks will gain popularity. Understanding these threats, implementing them in a test bed environment, and using them to test defense systems will help researchers keep one step ahead of the attackers.

DDoS benchmark generation. There is an ongoing effort to design attack benchmarks for evaluation of DDoS defenses. The attack taxonomy will help the completeness of benchmark generation. The defense taxonomy will help expose and identify common weaknesses of a class of DDoS solutions, and design tailored experiments to test these weaknesses.

Common vocabulary. With exception of well understood attacks (e.g., UDP flood, ICMP flood, etc.), researchers frequently resort to descriptive explanations of sophisticated attack mechanisms or weaknesses of a specific solution. The taxonomies offer a common vocabulary for these discussions.

Design of attack class-specific solutions. The aim of DDoS defenses is frequently to be a silver bullet for all possible attacks. This is unrealistic. The attack taxonomy facilitates identification of subsets of DDoS threats and design of tailored solutions. If enough such solutions are generated, and can operate jointly with synergistic effect, we could complete the puzzle.

Understanding solution constrains. The defense taxonomy highlights common performance constraints and weaknesses for each class of DDoS defenses. Understanding these problems will focus research efforts on solving them.

Identifying unexplored research areas. Examining the effectiveness of different defense classes against different classes of attacks should highlight unexplored venues for research.

7. RELATED WORK

In [41] authors present a classification of denial-of-service attacks according to the type of the target (e.g., firewall, Web server, router), a resource that the attack consumes (network bandwidth, TCP/IP stack) and the exploited vulnerability (bug or overload). This classification focuses more on the actual attack phase, while we are interested in looking at the complete attack mechanism in order to highlight features that are specific to distributed attacks. Hussain et al. [37] classify flooding (in our terminology *brute-force*) DDoS attacks based on number of agent machines performing the attack and whether the attack was reflected or not. In [35] and [36], Howard proposes a taxonomy of computer and network attacks. This taxonomy focuses on computer attacks in general and does not sufficiently highlight features particular to DDoS attacks. CERT is currently taking the initiative to devise a comprehensive taxonomy of computer incidents as part of the design of common incident data format and exchange procedures, but unfortunately results are not yet available. BBN is also working on generation of a DDoS attack overview, but its results are not yet released. The work in [60] provides a valuable discussion of the DDoS problem and of some defense approaches. A solid body of work on classification exists in the field of intrusion detection systems [36, 26, 6] and offers informative reading for researchers in the DDoS defense field.

8. CONCLUSION

The DDoS field contains a multitude of attack and defense mechanisms, which obscures a global view of the DDoS problem. This paper is a first attempt to cut through the obscurity and structure the knowledge in this field. The proposed taxonomies are intended to help the community think about the threats we face and the possible countermeasures.

One benefit we foresee from these taxonomies is that of fostering easier cooperation among researchers. Attackers cooperate to exchange attack code and information about vulnerable machines, and to organize their agents into coordinated networks of immense power and survivability. The Internet community must be equally cooperative within itself to counter the DDoS threat. Good taxonomies will facilitate communication and offer the a common language for discussing solutions. They will also clarify how different mechanisms are likely to work in concert, and identify areas of remaining weaknesses that require additional work. There is a pressing need for the research community to develop common metrics and benchmarks for DDoS defense evaluation. The taxonomies will be helpful in shaping these tasks, as well.

The proposed taxonomies are by no means complete and all-encompassing. New attacks will appear, some of which we cannot yet imagine. They will highlight new features for classification. Innovative approaches to DDoS defense will be designed. They will also offer new design features carrying their share of benefits and weaknesses. We expect these taxonomies to offer a foundation for classifying threats and defenses in DDoS field. As the field grows, the taxonomies will also grow and be refined.

9. ACKNOWLEDGEMENTS

Authors would like to thank Dr. Sven Dietrich for his helpful comments, members of the UCLA LASR group for valuable discussions, and Janice Wheeler for her infinite patience in editing this and many other papers. Verica Savic-Jovcic generated the customized marking scheme for subsections. This scheme has made the paper infinitely easier to read, and we are deeply grateful for that.

10. REFERENCES

[1] D. G. Andersen. Mayday: Distributed filtering for internet services. In *In Proceedings of 4th Usenix Symposium on Internet Technologies and Systems*, March 2003.

[2] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proceedings of 18th ACM SOSP*, October 2001.

[3] T. Anderson, T. Roscoe, and D. Wetherall. Preventing internet denial-of-service with capabilities. In *In Proceedings of HotNets II*, November 2003.

[4] Arbor Networks. *The Peakflow Platform.* http://www.arbornetworks.com.

[5] T. Aura, P. Nikander, and J. Leiwo. DOS-Resistant Authentication with Client Puzzles. *Lecture Notes in Computer Science*, 2133, 2001.

[6] S. Axelsson. Intrusion detection systems: A survey and taxonomy. Technical Report 99-15, Department of Computer Engineering, Chalmers University, March 2000.

[7] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *In Proceedings of the 2nd ACM SIGCOMM Internet Measurement Workshop*, November 2002.

[8] BBN Technologies. *Applications that participate in their own defense.* http://www.bbn.com/infosec/apod.html.

[9] BBN Technologies. *Intrusion tolerance by unpredictability and adaptation.* http://www.bbn.com/infosec/itua.html.

[10] S. Bellovin, M. Leech, and T. Taylor. ICMP Traceback Messages. *Internet draft, work in progress*, October 2001.

[11] D. J. Bernstein. Syn cookies. http://cr.yp.to/syncookies.html.

[12] CERT CC. *CA-2001-19 "Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL.* http://www.cert.org/advisories/CA-2001-19.html.

[13] CERT CC. *Code Red II.* http://www.cert.org/incident_notes/IN-2001-09.html.

[14] CERT CC. *Denial of Service Attacks.* http://www.cert.org/tech_tips/denial_of_service.html.

[15] CERT CC. *DoS using nameservers.* http://www.cert.org/incident_notes/IN-2000-04.html.

[16] CERT CC. *erkms and li0n worms.* http://www.cert.org/incident_notes/IN-2001-03.html.

[17] CERT CC. *Nimda worm.* http://www.cert.org/advisories/CA-2001-26.html.

[18] CERT CC. *Ramen worm.* http://www.cert.org/incident_notes/IN-2001-01.html.

[19] CERT CC. *Smurf attack.* http://www.cert.org/advisories/CA-1998-01.html.

[20] CERT CC. *TCP SYN flooding and IP spoofing attacks.* http://www.cert.org/advisories/CA-1996-21.html.

[21] CERT CC. *Trends in Denial of Service Attack Technology*, October 2001. http://www.cert.org/archive/pdf/DoS_trends.pdf.

[22] Cisco. Strategies to protect against Distributed Denial of Service Attacks. http://www.cisco.com/warp/public/707/newsflash.html.

[23] Cs3. Inc. *MANAnet DDoS White Papers.* http://www.cs3-inc.com/mananet.html.

[24] T. Darmohray and R. Oliver. *Hot spares for DDoS attacks.* http://www.usenix.org/publications/login/2000-7/apropos.html.

[25] D. Dean, M. Franklin, and A. Stubblefield. An algebraic approach to IP Traceback. In *Proceedings of the 2001 Network and Distributed System Security Symposium*, February 2001.

[26] H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion-detection systems. In *Computer Networks*, volume 31(8), pages 805–822, April 1999.

[27] D. Dittrich. *The DoS Project's trinoo distributed denial of service attack tool.* http://staff.washington.edu/dittrich/misc/trinoo.analysis.

[28] D. Dittrich. *The Tribe Flood Network distributed denial of service attack tool.* http://staff.washington.edu/dittrich/misc/tfn.analysis.txt.

[29] D. Dittrich, G.Weaver, S. Dietrich, and N.Long. *The mstream distributed denial of service attack tool.* http://staff.washington.edu/dittrich/misc/mstream.analysis.txt.

[30] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing. *RFC 2827*, May 2000.

[31] A. Garg and A. L. N. Reddy. Mitigation of DoS attacks through QoS Regulation. In *Proceedings of IWQOS workshop*, May 2002.

[32] T. M. Gil and M. Poletto. MULTOPS: a data-structure for bandwidth attack detection. In *Proceedings of 10th Usenix Security Symposium*, August 2001.

[33] K. Hafner and J. Markoff. *Cyberpunk: Outlaws and hackers on the computer frontier.* Simon & Schuster, 1991.

[34] G. Hardin. The Tragedy of the Commons. *Science*, 162(1968):1243–1248, 1968.

[35] J. D. Howard. *An analysis of security incidents on the Internet.* PhD thesis, Carnegie Mellon University, August 1998.

[36] J. D. Howard and T. A. Longstaff. *A common language for computer security incidents.*

[37] A. Hussain, J. Heidemann, and C. Papadopoulos. A Framework for Classifying Denial of Service Attack. In *Proceedings of SIGCOMM 2003*, 2003.

[38] Information Sciences Institute. *Dynabone.* http://www.isi.edu/dynabone/.

[39] J. Ioannidis and S. M. Bellovin. Pushback: Router-Based Defense Against DDoS Attacks. In *Proceedings of NDSS*, February 2002.

[40] A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proceedings of the 1999 Networks and distributed system security symposium*, March 1999.

[41] F. Kargl, J. Maier, and M. Weber. Protecting web servers from distributed denial of service attacks. In *Proceedings of 10th International World Wide Web Conference*, May 2001.

[42] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *Proceedings of SIGCOMM 2002*, 2002.

[43] F. Lau, S. H. Rubin, M. H. Smith, and L. Trajkovic. Distributed Denial of Service Attacks. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 2275–2280, Nashville, TN, USA, October 2000.

[44] J. Leiwo, P. Nikander, and T. Aura. Towards network denial of service resistant protocols. In *Proceedings of the 15th International Information Security Conference*, August 2000.

[45] J. Li, J. Mirkovic, M. Wang, P. Reiher, and L. Zhang. SAVE: Source Address Validity Enforcement Protocol. In *Proceedings of INFOCOM 2002*, June 2002. to appear.

[46] R. Mahajan, S. Bellovin, S. Floyd, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. *ACM Computer Communications Review*, 32(3), July 2002.

[47] G. R. Malan, D. Watson, F. Jahanian, and P. Howell. Transport and Application Protocol Scrubbing. In *Proceedings of INFOCOM 2000*, pages 1381–1390, 2000.

[48] Mazu Networks. *Mazu Technical White Papers*. http://www.mazunetworks.com/white_papers/.

[49] McAfee. *Personal Firewall*. http://www.mcafee.com/myapps/firewall/ov_firewall.asp.

[50] C. Meadows. A formal framework and evaluation method for network denial of service. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, June 1999.

[51] J. Mirkovic. *D-WARD: Source-End Defense Against Distributed Denial-of-Service Attacks*. PhD thesis, University of California Los Angeles, August 2003.

[52] J. Mirkovic, G. Prier, and P. Reiher. Attacking DDoS at the Source. In *Proceedings of the ICNP 2002*, November 2002.

[53] D. Moore. *The spread of the code red worm (crv2)*. http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml.

[54] D. Moore, G. Voelker, and S. Savage. Inferring Internet Denial-of-Service Activity. In *Proceedings of the 2001 USENIX Security Symposium*, 2001.

[55] R. Naraine. *Massive DDoS Attack Hit DNS Root Servers*, October 2002. http://www.esecurityplanet.com/trends/article/0,,10751_1486981,00.html.

[56] National Infrastructure Protection Center. *Advisory 01-014: New Scanning Activity (with W32-Leave.worm) Exploiting SubSeven Victims*, June 2001. http://www.nipc.gov/warnings/advisories/2001/01-014.htm.

[57] E. O'Brien. *NetBouncer : A practical client legitimacy-based DDoS defense via ingress filtering*. http://www.nai.com/research/nailabs/development-solutions/netbouncer.asp.

[58] K. Park and H. Lee. On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets. In *Proceedings of ACM SIGCOMM 2001*, August 2001.

[59] V. Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *ACM Computer Communications Review (CCR)*, 31(3), July 2001.

[60] V. Razmov. *Denial of Service Attacks and How to Defend Against Them*. http://www.cs.washington.edu/homes/valentin/papers/DoSAttacks.pdf.

[61] SANS Institute. *NAPTHA: A new type of Denial of Service Attack*, December 2000. http://rr.sans.org/threats/naptha2.php.

[62] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical Network Support for IP Traceback. In *Proceedings of ACM SIGCOMM 2000*, August 2000.

[63] C. Schuba, I. Krsul, M. Kuhn, G. Spafford, A. Sundaram, and D. Zamboni. Analysis of a denial of service attack on TCP. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, May 1997.

[64] S.Dietrich, N. Long, and D. Dittrich. An Analysis of the "shaft" distributed denial of service tool. In *Proceedings of LISA 2000*, 2000. http://www.adelphi.edu/ spock/shaft-lisa2000.pdf.

[65] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Hash-Based IP Traceback. In *Proceedings of ACM SIGCOMM 2001*, August 2001.

[66] D. X. Song and A. Perrig. Advanced and authenticated marking schemes for IP Traceback. In *Proceedings of IEEE Infocom 2001*, 2001.

[67] Sourcefire. *Snort: The Open Source Network Intrusion Detection System*.

[68] O. Spatscheck and L. L. Petersen. Defending Against Denial of Service Attacks in Scout. In *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation*, February 1999.

[69] S. Staniford, J. Hoagland, and J. McAlerney. Practical automated detection of stealthy portscans. *Journal of Computer Security*, 10(1/2), 2002.

[70] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the internet in your spare time, 2002. In Proceedings of the 11th USENIX Security Symposium.

[71] Tripwire. *Tripwire for servers*. http://www.tripwire.com/products/servers/.

[72] N. Weaver. *Warhol Worm*. http://www.cs.berkeley.edu/ nweaver/worms.pdf.

[73] M. Williamson. Throttling viruses: Restricting propagation to defeat malicious mobile code. In *18th Annual Computer Security Applications Conference*, December 2002.

[74] J. Yan, S. Early, and R. Anderson. The XenoService – A Distributed Defeat for Distributed Denial of Service. In *Proceedings of ISW 2000*, Oct. 2000.

[75] V. Yegneswaran, P. Barford, and J. Ullrich. Internet intrusions: Global characteristics and prevalence. In *In Proceedings of the 2003 ACM SIGMETRICS International conference on Measurement and Modeling of Computer Systems*, pages 138–147, 2003.

[76] Y. L. Zheng and J. Leiwo. A Method to Implement a Denial of Service Protection Base. In *Information Security and Privacy*, volume 1270 of LNCS, pages 90–101, 1997.