D-WARD:
DDoS Network Attack Recognition and Defense

Ph.D. Dissertation Prospectus

Jelena Mirković

Ph.D. Advisor: Peter Reiher

January 23, 2002

Abstract

Distributed denial-of-service (DDoS) attacks present an immense threat to the Internet. They engage the power of a vast number of coordinated Internet hosts to consume some critical resource at the target and deny the service to legitimate clients. As a side effect, they frequently create network congestion on the way from a source to the target, thus disrupting normal Internet operation. The existing security mechanisms do not provide effective defense against these attacks. The large number of attacking machines and the use of source IP address spoofing make the traceback impossible. The use of legitimate packets for the attack and the varying of packet fields disable characterization and filtering of the attack streams. The distributed nature of the attacks calls for a distributed response, but cooperation between administrative domains is hard to achieve, and security and authentication of participants incur high cost.

In this thesis, we investigate effective methods for the detection and response to DDoS attacks. We first present characteristics of these attacks and emerging trends in the DDoS field. We survey possible ways to recognize DDoS attacks at different points in the Internet and to prevent or to stop them effectively. With this overview, we also present related work. We then propose a DDoS defense system deployed at a source-end network that autonomously detects and stops attacks originating from this network. Attacks are detected by monitoring two-way traffic flows between the network and the rest of the Internet. Monitored flows are periodically compared with predefined models of normal traffic, and those flows classified as part of a DDoS attack are rate-limited. We present the preliminary implementation of this system and performance results. We also devote special attention to security issues of the system to avoid misuse or bypass by attackers. We further investigate the possible cooperation within the source-end network to facilitate better detection and response. We also investigate possible interactions between the victim and source-end network to offer reliable feedback to the system. Since the degree of deployment of the proposed system directly affects its effectiveness, we investigate the best deployment strategy in source-end networks and the possible modification of the system for deployment in core networks. We conclude with a summary of the expected contributions of this thesis.

Contents:

1. Introduction

Distributed denial-of-service attacks are comprised of large numbers of packet streams from disparate sources. These streams converge on the victim, consuming some key resource and rendering it unavailable to legitimate clients. This can be done by exploiting vulnerabilities in systems or protocols at the victim network, or by merely requesting normal services at intense rates.

There are no common characteristics of DDoS streams that could be used for their detection and filtering. The attacks achieve their desired effect by sheer volume of attack packets, and can afford to vary all packet fields to avoid characterization. In addition to this, attackers follow advances in the security field and adjust their tools to defeat new security systems.

The cooperation of distributed sources makes DDoS attacks hard to combat or trace back. Any defense against these attacks must address their distributed nature to be successful. Several distributed DDoS defense systems have been proposed that deploy cooperation between intermediate routers to combat an attack. These routers are augmented to monitor traffic and grant requests for rate limiting of the streams that they deliver to their peers. While these approaches seem promising, they require the cooperation of every router from the source to the destination host for response propagation. Since the Internet is not under any single administrative control, global deployment of such mechanisms is hard to enforce. Also, cooperation between routers requires strongly secured and authenticated communication, which incurs high cost.

The attacks ideally should be stopped as close to the sources as possible, saving network resources and reducing congestion. In this thesis we propose a DDoS defense system that is deployed at the source-end and prevents machines at associated networks from participating in DDoS attacks. This system is deployed at the network's "exit" router and monitors two-way traffic between the network and the rest of the Internet. The online traffic statistics are periodically compared to predefined models of normal traffic, and non-complying flows are rate-limited. We investigate different monitoring and rate limiting strategies and possible gains from cooperation with other routers and the victim. We dedicate special attention to securing the system and making it impossible for attackers to misuse it or to bypass it and still perform successful attacks. Since general deployment cannot be expected, we investigate the benefits that can be gained from partial deployment and propose the best deployment points to achieve highest efficiency.

In Section 2 we present the problem statement, and we give a general overview of DDoS attacks and summarize challenges in the DDoS defense field. Section 3 surveys different strategies to combat DDoS attacks and gives an overview of related work. Section 4 defines the requirements for an effective DDoS defense system. Section 5 describes the proposed DDoS defense system, gives details about its architecture and components, and presents preliminary experimental results. Sections 6 and 7 discuss two vital functions of the system, attack detection and response, in more detail. Section 8 investigates security issues, and Section 9 discusses implementation considerations such as partial deployment. Section 10 provides a summary of thesis contributions, and Section 11 concludes the proposal.

2. Problem Statement

DDoS attacks appeared as a serious threat to the Internet in 1999 and have since experienced a rapid development of techniques to prepare and perform the attack and to avoid detection. Distributed intruder technology is not new; however, it is maturing to the point where even unsophisticated intruders could do serious damage. In the following sections we describe the methods to perform DDoS attacks and analyze several infamous DDoS tools.

2.1. DDoS Attack Scenario

There are several steps in preparing and conducting a DDoS attack:

1. **Recruitment** - The attacker chooses one or several machines in the Internet that will perform the attack. Those machines, usually called *agents*, are commonly: (a) external to the victim's own network, to avoid efficient response from the victim, and (b) external to the network of the attacker, to avoid liability if the attack is traced back. The agent machines need to have some vulnerability that the attacker will use to gain access to them. It is also preferable that they have abundant resources that will enable them to generate powerful attack streams. Early on, this selection process was performed manually, but it was soon automated. Early scanning tools produced lists of hosts that were potentially vulnerable.

2. **Compromise -** The attacker gains access (usually as a root) to agent machines by exploiting security holes and plants the attack code. He further takes steps to protect the code from discovery (by renaming the files, making them hidden or placing them in system directories) and deactivation (by instructing a system scheduler, such as Linux `cron`, to restart the code periodically). This phase was also soon automated. The scanning tools exploited the discovered vulnerability to gain access to the machine and deploy the attack code; then they reported the list of compromised hosts to the attacker. Recently, several automated, self-propagating tools appeared (Ramen worm [32], Code Red [13]), that integrate scanning, exploitation, deployment and propagation phases, thus offering fast deployment and high survivability of the attack code.

3. **Communication** - Agents report their readiness to the attacker via *handlers* - compromised machines that will be used to control the attack. In the early DDoS days, the IP addresses of handlers were hardcoded in the attack code, and handlers stored the encrypted information about available agents in the file. Thus the discovery of a single machine in a DDoS network revealed all other participants. Recently the Internet Relay Chat (IRC) channels started being used for communication. The IRC server tracks the addresses of connected agents and handlers and facilitates communication between them. The discovery of the single participant leads to discovery of the communication channel, but other participants' identities are protected.

4. **Attack** - The attackers usually command the onset of attacks via handlers and communication channels to the agents. The target, duration and features of the attack packets such as type, length, TTL, port numbers, etc., can be customized. Early on, DDoS tools discovered the benefits of varying attack packet properties to avoid detection: IP spoofing is used to mask the source address, and packet type, header fields (all except the destination IP address) and communication channel can all be changed during the attack. Some attack tools, such as Code Red [13], hardcode all these parameters to avoid communication with the handlers.

2.2. Overview of Several DDoS Tools

Distributed denial-of-service attacks exploit different strategies to exhaust the resources of the victim. Some protocol implementations have bugs that allow a few malformed packets to severely degrade server or network performance. The victim can frequently prevent these attacks by implementing various patches to fix the vulnerability, or by filtering malformed packets. On the other hand, DDoS attacks can leverage the power of many distributed machines to make a massive number of legitimate requests for a service from a single host. Since these requests are legitimate, the victim cannot refuse to service them, nor can it recognize them as part of the attack until its resources are exhausted.

Attackers follow trends in the network security field and adjust their attacks to defeat current defense mechanisms. Spoofing of source addresses is used to avoid traceback, and decoy packets and encryption

are used to combat signature-based detection. We now provide a quick overview of the several well-known DDoS attack tools in order to illustrate the variety of mechanisms deployed.

Trinoo [25] is a simple tool used to launch coordinated UDP flood attacks against one or many IP addresses. The attack uses constant-size UDP packets to target random ports on the victim machine. The handler uses UDP or TCP to communicate with the agents. This channel can be encrypted and password protected as well. Trinoo does not spoof source addresses although it can easily be extended to include this capability.

Tribe Flood Network (TFN) [26] can generate UDP and ICMP echo request floods, TCP SYN floods and ICMP directed broadcast (e.g., Smurf). It can spoof source IP addresses and also randomize the target ports. Communication between handlers and agents occurs exclusively through `ICMP_ECHO_REPLY` packets.

Stacheldraht [27] combines features of Trinoo (handler/agent architecture) with those of the original TFN (ICMP/TCP/UDP flood and Smurf style attacks). It adds encryption to the communication channels between the attacker and Stacheldraht handlers. Communication is performed through TCP and ICMP packets. It allows automated update of the agents using `rcp` and a stolen account at some site as a cache. New program versions will have more features and different signatures to avoid detection.

TFN2K [15] is the variant of TFN that includes features designed specifically to make TFN2K traffic difficult to recognize and filter. Targets are attacked via UDP, TCP SYN, `ICMP_ECHO` flood or Smurf attack, and the attack type can be varied during the attack. Commands are sent from the handler to the agent via TCP, UDP, ICMP, or all three at random. The command packets may be interspersed with any number of decoy packets sent to random IP addresses to avoid detection. In networks that employ ingress filtering as described in [24], TFN2K can forge packets that appear to come from neighboring machines. All communication between handlers and agents is encrypted and base-64 encoded.

The **mstream** [28] tool uses spoofed TCP packets with the ACK flag set to attack the target. Communication is not encrypted and is performed through TCP and UDP packets. Access to the handler is password protected. This program has a feature not found in other DDoS tools. It informs all connected users of access, successful or not, to the handler(s) by competing parties.

Shaft [9] uses TCP, ICMP or UDP flood to perform the attack, and it can deploy all three styles simultaneously. UDP is used for communication between handlers and agents, and messages are not encrypted. Shaft randomizes the source IP address and the source port in packets. The size of packets remains fixed during the attack. A new feature is the ability to switch the handler's IP address and port during the attack.

The **Code Red** [13] worm is self-propagating malicious code that exploits a known vulnerability in Microsoft IIS servers for propagation. It achieves a synchronized attack by preprogramming the onset and abort time of the attack, attack method and target addresses (i.e., no handler/agent architecture is involved).

2.3. DDoS Defense Problems

While previous sections gave the overview of DDoS attack characteristics, we summarize here the important features that make defense against DDoS attacks such a hard problem to solve:

1. **Lack of common characteristics of DDoS streams** - There are no common characteristics of DDoS streams that can be used for their detection and filtering. The attacks achieve the desired effect by

sheer volume of the attack packets, and can afford to vary all packet fields to avoid characterization. In addition to this, attackers follow advances in the security field and adjust their tools to bypass new security systems.

2. **Lack of cooperation across administrative domains** - The cooperation of distributed sources makes DDoS attacks hard to combat or trace back. At the same time, there is no cooperation between participating administrative domains (source, target and intermediate domains that carry DDoS traffic) that would enable fast, efficient and distributed response to the attack.

3. **Automated tools** - DDoS attack code and automated tools for propagation and deployment can be easily downloaded from the Internet. Thus, even novice intruders can perform powerful attacks.

4. **Hidden identity of participants** - Attackers use IP source address spoofing to hide the identity of the attacking machines and use handler machines to hide their own identity. In addition to this, recent use of IRC channels for communication makes it unnecessary to store peer information on the agent or handler machine for future communication. Thus, no single participant has the information about the identity of other machines in DDoS network.

5. **Persistent security holes in Internet hosts** - There are easily downloadable patches for the majority of security holes that are used by attackers to compromise agent machines. Still, there is a vast community of Internet users that are not sufficiently technically sophisticated or security-conscious to apply these patches. Their machines resemble loaded weapons that can be used to target any machine on the Internet. There is no reason to believe that we will ever be able to enforce a sufficient security level on all machines in the Internet.

3. Related Work

From their early days, DDoS attacks have attracted a lot of attention in research and commercial communities. Many security efforts aim at exploiting a certain feature of current attacks to prevent them or to constrain their effect. Unfortunately, the attackers closely follow developments in the security field and are often able to modify the given feature and thus bypass the security system. We present here related work on both prevention and defense areas.

3.1. DDoS Attack Prevention

An increased security of Internet hosts would prevent their exploitation by attackers and would thus offer improved security against DDoS attacks. Application vendors regularly publish patches for security holes in their applications. Many virus detection programs include in their database signatures of known attack tools, and provide the user with the deactivation code. Intrusion detection systems detect attack signatures or anomalous behavior of the compromised host and alert the user. Due to the constant discovery of new vulnerabilities and appearance of new exploits, both patches and signature databases need to be regularly updated from vendor sites. Even though there are tools that automate this update process and make it transparent to the user, there is still a large number of Internet hosts that run unpatched code and have no virus detection software. These machines represent a constant threat to the rest of the Internet.

Prevention approaches will always be vulnerable to new attacks for which signatures or patches do not exist in the database. New applications and services constantly appear that introduce new bugs and vulnerabilities and open different ways to exploit the system. Prevention approaches offer increased security but can never completely remove the threat of DDoS attacks.

3.2. DDoS Attack Detection and Defense

A DDoS attack engages the power of dispersed machines and DDoS streams traverse many Internet hosts. The involved machines are in the *victim network* (administrative domain that contains the target of the attack), *intermediate network* (administrative domain that forwards attack packets to the victim) and *source network* (administrative domain that has one or more attacking machines). We investigate potential detection and defense opportunities in each of these domains, and give an overview of related work.

3.2.1. Victim Network

While the intermediate routers that carry DDoS traffic might occasionally experience congestion due to large traffic volume, denial-of-service attacks inflict the greatest harm on the victim. Therefore, the victim has the greatest incentive to deploy a DDoS defense system, and maybe sacrifice some of its performance and resources for increased security. Historically, most of the systems for combating DDoS attacks have been designed to work on the victim side.

Much of the work related to DDoS defense has been carried on in the area of intrusion detection. Intrusion detection systems detect DDoS attacks either by using the database of known signatures or by recognizing anomalies in system behavior. Several popular network monitors perform mostly signature-based detection with some limited statistical processing, such as CISCO's NetRanger [1], NID [11], SecureNet PRO [12], RealSecure [14], and NFR-NID [16]. Most of these systems do not take automated action to stop the attack, but just raise an alert to the system administrator.

In [10] an on-off control approach is proposed to fight DDoS attacks in a victim network. In this approach, a router detects that it is the target of a DDoS attack by monitoring its buffer queue size. Once the queue size grows over a specified threshold, the router reacts by switching to a different mode of operation and throttling incoming traffic. Only packets belonging to a certain type of traffic identified as problematic are dropped. When the queue size is reduced, throttling is stopped. This approach is similar to the RED congestion control mechanism [17]. For a reasonable volume of attack packets, this approach efficiently protects the target of the attack from overflowing its buffers by early detection of DDoS attack, while at the same time does not completely cut off the traffic from the source network.

Porras and Valdes [8] have recognized that the analysis of TCP/IP packet streams through the network could be beneficial to fighting intrusion attacks. This idea is implemented in EMERALD, an environment for anomaly and misuse detection ([5], [6], [7], and [8]) developed at SRI. It combines a signature-based approach to IDS with statistical analysis for anomaly detection.

CISCO routers have built-in features such as debug logging and IP accounting that can be used for characterizing and tracing common attacks [2]. An access list can be configured to log many network events, e.g., to count packets received and categorize them by type, and to log sources of packets that are of special interest. This feature only gathers statistical data and offers no automated analysis or response. Manual configuration is also needed to characterize the attacks and initialize access lists.

All the above systems increase a victim's ability to recognize early that it is the target of an attack, and thus gain more time to respond. This feature would be useful for the attacks that aim to degrade a victim's services instead of denying them completely, and are thus difficult to detect. However, most of the attacks aim at crippling the victim completely by exhausting some of its resources; this effect is so excessive and differs so greatly from normal behavior that the detection is trivial. Most of the proposed systems then attempt to characterize the attack traffic and install filtering rules in the upstream routers in the victim

network. In some cases this can successfully relieve the attack effect (i.e., a Web server that is under a UDP flood attack can request that all UDP traffic be filtered out). However, this technique fails when:

(a) The offending traffic has the same characteristics as the legitimate traffic (i.e., the Web server is under HTTP request flood or TCP SYN flood), or
(b) The amount of offending traffic is so great that the filtering mechanism in the upstream router cannot handle it.

The second problem is a natural consequence of the Internet architecture. The Internet core has much greater resources than its edges, and thus a coordinated attack can fully utilize those resources to deliver overwhelming volume of packets at the victim network and exhaust its limited resources. This leads to conclusion that intermediate networks that forward packets to the victim should participate in the DDoS response.

3.2.2. Intermediate Network

Bradley et al., have proposed WATCHERS [3], an algorithm to detect misbehaving routers that launch denial-of-service attacks by absorbing, discarding or misrouting packets. WATCHERS uses the conservation of flow principle to examine flows between neighbors and endpoints. Hughes et al., have shown [4] that WATCHERS makes several implicit assumptions that do not hold and have proposed modifications to correct this, thereby increasing the cost and complexity of the algorithm. Even with these modifications, WATCHERS requires explicit communication between routers. The algorithm cannot detect packets with forged source addresses. Even worse, such packets could be used by the attacker to misidentify a target as a bad router. WATCHERS can only detect compromised routers; it is helpless against misbehaving hosts. It also assumes that every router knows the topology of the network, which is not a feasible solution for large networks.

Several traceback mechanisms have been proposed to locate attacking nodes ([29], [30], [33] and [34]). These systems provide information about the identity of attacking machines, but do not themselves stop DDoS attacks. Tracing becomes ineffective when the volume of attack traffic is small or the attack is distributed [35]. Traceback techniques are also susceptible to attempts by the attackers to deceive them.

Several filtering mechanisms have been proposed to prevent spoofing of the source address in IP packets ([23], [24], and [31]). While IP spoofing is not necessary for DDoS attacks, it helps attackers to hide the identity of attacking machines so they can reuse them for future attacks. Eliminating IP spoofing would facilitate easy distinction of normal flows from attack flows, and would help greatly to identify attacking machines. However, with highly distributed attack sources, the knowledge of the identity of attack machines still does not prevent an efficient attack.

In [18] Floyd et al., have proposed to augment routers with the ability to detect and control flows that create congestion, which is frequently a sign of a DDoS attack. Flows are detected by monitoring the dropped packets in the router queue and identifying high-bandwidth aggregates that are responsible for the majority of drops. A rate limit is then imposed on the aggregate. If the congested router cannot control the aggregate by itself, it can ask the adjacent upstream router to also impose the rate limit on that aggregate. This upstream rate limiting is called pushback and can be recursively propagated until it reaches the routers in the source networks. This approach offers a powerful tool to not only combat DDoS attacks but to also locate and remove network congestion caused by any other reason. However, it requires significant augmentation of the routers on the whole path from the victim to the sources. A single legacy router on the path complicates the scheme and imposes the need for secure communication of non-adjacent routers, which makes the system vulnerable to attacks. The approach also requires cooperation of different administrative domains, which is currently hard to achieve.

Response to a DDoS attack by the intermediate network is obviously more effective than a victim network's response since large volumes of attack traffic can be handled easily and attacks can be

cooperatively traced back to the sources. However, there are several problems that prevent wide deployment of these approaches:

(a) Intermediate network's performance - The intermediate network usually handles large traffic volumes to which it dedicates most of its resources. Requiring additional resources for traffic profiling and rate limiting is likely to degrade the network's performance, which might in turn degrade Internet service as a whole.

(b) Attack detection - It is very difficult for an intermediate network to detect the attack and identify the victim, since it usually does not feel any effect of the attack. Thus, most proposed systems rely on a signal from the victim to trigger the response. The victim must be authenticated to assure that the attackers cannot misuse the system, which imposes an additional cost. Also, frequently the victim host is so damaged by the attack that it cannot send the signal to the intermediate network.

(c) Lack of interdomain cooperation - There is currently very little cooperation between different administrative domains. Adding a service that would enable intermediate domain A to restrict traffic from B to C based on some signal or on its own observations would require a large restructuring of interdomain relations. Highly secure and authenticated communication between all participants is also necessary to prevent misuse, which imposes high additional cost due to the large number of participants.

(d) General deployment - All proposed systems rely on cooperation between participants to combat the attack. If there are non-participating domains, or the participants are not adjacent, the performance of the system degrades rapidly.

3.2.3. Source Network

One might argue that it suffices to take a well-developed network-based intrusion detection system and deploy it on the source side, reversing its functions to examine outgoing traffic, to achieve effective DDoS defense system. However, the attack appears different at the source than at the target network. At the target network all DDoS flows converge and affect the system greatly so that detection is inevitable; at the source end those flows are still dispersed and can appear as a set of perfectly valid transactions. It is usually the sheer amount of these transactions that saturates the target network.

We are currently aware of only one research effort that investigates DDoS defense at the source network. In [19] Gil and Poletto propose a heuristic and a data-structure (MULTOPS) that network devices can use to detect DDoS attacks. Each network device maintains a multi-level tree that monitors certain traffic characteristics and stores data in nodes corresponding to subnet prefixes at different aggregation levels. The tree expands and contracts within a fixed memory budget. The system is designed so that it can operate as either a source-end or victim-end DDoS defense system. The attack is detected by abnormal values of packet ratio, and offending flows are rate-limited. Non-TCP flows in systems using MULTOPS can either be misclassified as the attack flows by the detection mechanism, or recognized as special and rate-limited to a fixed value. In the first approach, harm is done to a legitimate flow, while in the second approach a sufficiently distributed attack can still make use of the allowed rate to achieve the effect. The paper does not offer many details on the rate limiting mechanism, and does not address the removal of the rate limit after the attack.

Placing DDoS defense close to the sources of the attack has many advantages over placing it further downstream. The attack flows can be stopped before they enter the Internet core and before they aggregate with other attack flows, thus achieving the power to create network congestion and exhaust resources. Being close to the sources can facilitate easier traceback and investigation of the attack. As with an intermediate network, the source network has a hard time detecting the occurrence of the attack on one of its peers, since it itself does not experience any difficulties. However, a source network can sacrifice some of its resources and performance for better DDoS detection, since it does not handle such large volumes of traffic as does an intermediate network. Due to the low degree of flow aggregation,

more complex detection strategies can be deployed to achieve higher accuracy. Since a source network only sees a small number of total flows, policing these flows will have a small effect on sufficiently distributed attacks. A high degree of deployment would offer a better effectiveness guarantee. However, the motivation for deployment of DDoS defense system in the source network is low, since such a system does not directly protect deploying network resources and might also restrict legitimate traffic from this network in the case of unreliable attack detection.

4. Requirements for a DDoS Defense System

The design of an effective DDoS defense system faces a great challenge in successfully detecting and stopping attacks, while at the same time preserving current performance guarantees to legitimate traffic. The following are requirements that such a system has to meet to be widely accepted:

1. **High security** - A DDoS defense system must ensure that it cannot be misused to degrade or deny service to legitimate clients. It also must be resistant to attempts by attackers to bypass or to disable it.

2. **Reliable attack detection** - We define an attack as a set of packet streams that consume some critical resource at the target thus disrupting its normal operation.[1] Ideally we would like a DDoS defense system to detect every DDoS attack on a set of machines it is protecting and to have no false positives (legitimate connections that are misclassified as an attack). Such ideal detection usually incurs high cost, so this requirement is frequently relaxed, resulting in a less radical attack response.

3. **Independent attack detection and response** - While cooperation among multiple machines in attack detection and response offers several benefits (as we have discussed in Section 3), it imposes a requirement for highly secure and authenticated communication between participants. A DDoS defense system should offer sufficient functionality in independent operation and leave cooperation as optional property.

4. **Low performance cost** - Resource requirements of the system must not degrade the performance of the deploying network.

5. **Incremental benefit with incremental deployment** - Design of a DDoS defense system must be such that partial deployment still offers some protection from DDoS attacks, and that the benefits grow incrementally as number of deployment points increases.

6. **Efficiency** - When an attack is detected, the system should engage in a response that significantly reduces the effectiveness of the attack, regardless of the attack characteristics.

7. **Realistic design** - The system should require small or no changes to the existing Internet infrastructure

Desirable, but not required features of a DDoS defense system are:

8. **Higher penalty for recurring attacks** - Regardless of detection and response strategy, attackers will always have an option to repeat the attack after a long period of time. We would like the DDoS defense system to be able to recognize some of these attempts early and act more restrictively to prevent recurring damage to the victim.

9. **Traceback** - It is desirable to identify the attack machines and assure that they are cleansed, to prevent future misuse.

---

[1] Note that this definition encompasses not only malicious attacks but also "flash crowds," situations when a multitude of legitimate requests overwhelms a popular Web site.

10. **Cooperation with the victim -** If it is possible to achieve reliable and secure communication with the victim at an acceptable cost, this would offer the benefit of reliable feedback on the effectiveness of the DDoS response.

5. Source Router Approach

In this thesis we propose to design, implement and evaluate a DDoS defense system that is deployed in the source network. This system uses the router (hereafter called the *source router*), that serves as a gateway between the source network and the rest of the Internet, to detect and limit DDoS streams long before they reach the target. The proposed design assumes that the source network is an edge network, i.e., most of the traffic passing through the edge routers either originates within this network or is destined for some machines in this network. We assume that the source router is able to identify the interfaces on which it receives the source network's incoming and outgoing traffic, either through some protocol or through manual configuration. We further assume that all machines on the source network use this router as the "exit router" to reach a particular set of destinations. The source router is thus in a unique position to protect these destinations from denial-of-service attacks originating at the source network by arbitrating communication with them.

Detection of denial-of-service attacks is particularly hard at the source end, since the attacking flows might not be heavily aggregated at this point in the network, and the effect of the attack is usually not so prominent. As described in Section 2, typical denial-of-service attacks lack common characteristics that enable us to detect and filter flows by traditional methods. These attacks therefore have to be detected after their effect is felt by the victim, or after the heavy aggregation of attack flows creates congestion somewhere in the network.

We argue here that there is a single common characteristic of DDoS attacks; this is the goal of consuming significant quantities of victim resources. This characteristic cannot be modified by the attacker without compromising the success of the attack.[2] Thus, we can use this characteristic to detect the occurrence of the attack and identify the victim. We propose that the source router detect the attack by observing two-way communication between machines on the source network and the outside hosts. The source router monitors the behavior of each destination with which the source network communicates, looking for difficulties in communication, such as reduction of number of response packets or longer inter-arrival times. These difficulties can be a sign of denial-of-service attacks against the specific destination or of severe network congestion on route to the destination. These difficulties are detected by periodically comparing the parameter values of the two-way traffic for each destination against a predefined model of normal traffic. If the comparison reveals the possibility of a DDoS attack, the source router responds by imposing a rate limit on all outgoing traffic flows for this destination. The outcome of subsequent observation intervals either confirms or refutes this hypothesis. Confirmation further restricts the allowed rate limit, whereas refutation leads to a slow increase of the allowed outgoing traffic rate, according to the degree of well-behavior of outgoing flows.

The source router detects the attack and responds to it autonomously, without communication with other routers or human intervention. More accurate attack detection might be achieved if the source router relied on a signal from the victim instead of its own incomplete observations. This approach requires reliable, secure and authenticated communication between the victim and the source router, which cannot

[2] An attacker could decide to degrade a victim's services instead of denying them, and thus avoid detection. While this is still undesirable behavior, it does not disable the victim's services. Attackers have historically aimed at inflicting visible damage to the victim to demonstrate their power. It is possible that unnoticeable attack methods, such as gradual degradation of victim's services, would not appeal to them.

always be guaranteed. We plan to investigate, as a part of this thesis, the benefits and costs that this communication would add to the system, but the basic system design should still offer reasonably reliable autonomous detection. A more complete observation and more appropriate responses might be possible if all border routers in the source network were allowed to exchange information. The benefits and cost of this approach will also be investigated in this thesis.

5.1. System Architecture

The proposed DDoS defense system consists of *observation* and *throttling* components, which can be part of the source router itself or can belong to a separate unit that interacts with the source router to obtain traffic statistics and install filtering rules. Figure 1 depicts the architecture corresponding to the second approach. The observation component monitors the outgoing and incoming packets and gathers statistics on two-way communication between the source network and the rest of the Internet. These statistics are classified per destination. Periodically, statistics are compared to a predefined model of normal traffic, and the result of this comparison is passed to the throttling component. The throttling component adjusts the filtering rules based on this characterization and on flow behavior.

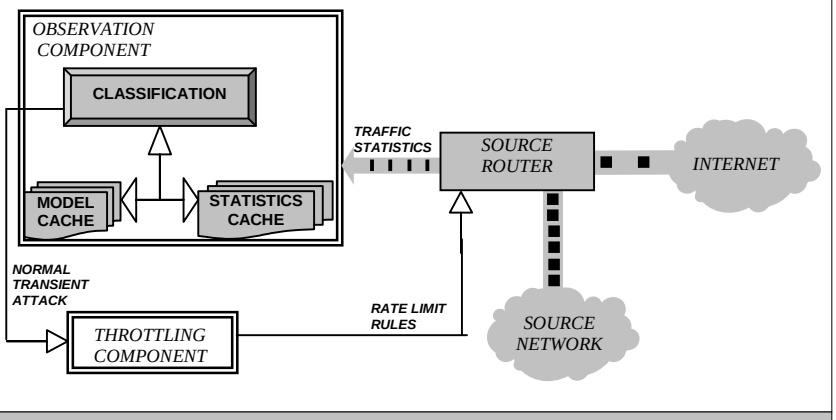

Figure 1: Architecture of the proposed DDoS defense system.

A prototype of the system incorporating all the elements described here has been built and tested. The following sections describe the design followed in this implementation and present results of experiments. We anticipate that many details of the system will change during the course of the research.

5.1.1. Observation Component

The observation component monitors all traffic passing through the source router. Each packet is classified as *incoming* or *outgoing* based on its source or destination address, or arriving interface.[3] Information in the packet header is then used to update statistics on current flows. Periodically, statistics are compared with a model of normal traffic, and flows are characterized as being *normal*, *transient* or *attack* flows. Normal flows are those flows whose parameters match those of the model and which have not been recently classified as attack flows. Attack flows are those flows whose parameters are outside of the model boundaries. Transient flows are those whose parameters match those of the model but which have been recently classified as attack flows; thus their rate-limit must be relaxed slowly to avoid recurring attacks.

[3] The exact classification method depends on how the router is connected to the source network. It is obvious that classification by source address cannot be performed reliably if spoofing is possible. If a router does not relay transit traffic, or it can clearly identify "incoming" and "outgoing" interfaces with regard to its source network, then classification by destination address or arriving interface can be performed. In our testbed we used classification by destination address.

5.1.1.1. Statistics Gathering

The purpose of statistics gathering is to characterize the current behavior of each Internet host with which the source network is communicating. The statistics provide the information needed to discover difficulties in communication caused by denial-of-service attacks on the destination host, or by network congestion along the route to the destination. The statistics are classified per IP address of the *foreign*[4] Internet host. Keeping a record for each existing IP address is infeasible, so the size of the statistics cache is limited and the cache is purged periodically. Figure 2 shows the number of cache records in subsequent observation intervals obtained by gathering statistics from different traces: (1) a trace collected at the "exit" router of the UCLA Computer Science Department network, (2) a LBL-PKT trace from Lawrence Berkeley Laboratory [20], (3) a DEC-PKT trace from Digital Equipment Corporation [21], and (4) a NLANR-auckland trace from NLANR trace archive [22]. Even though the cache size varies among different traces, it remains fairly stable across time for a particular trace. This justifies our assumption that the maximum cache size can be preset for a given source network and thus the storage requirement can be limited while not losing important information. The cache is purged after flows are categorized by deleting the records for flows that have been classified as normal during the comparison. Records for the attack and transient flows are kept, but the statistics are reset. If the cache overflows during the observation interval, the record for the destination receiving the smallest number of bytes is expunged.

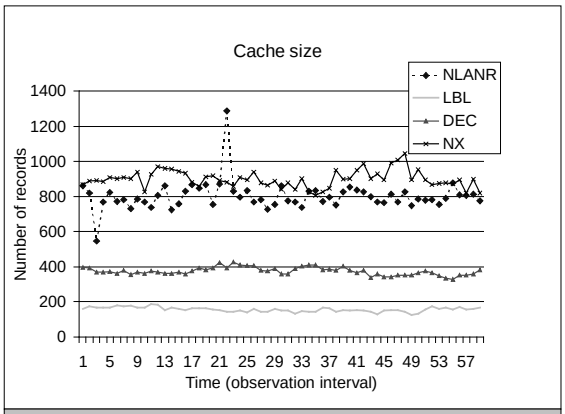

Figure 2: Cache size for several traces.

Only data from the packet's IP header is used for statistics gathering. This allows fast operation since most routers are optimized for fast lookup of IP header fields and classification of packets based on these fields. This choice also reduces the size of the state kept for each record. Finally, it provides the possibility of verification of our detection technique against many published packet traces.

A record in the statistics cache corresponding to some peer address PA consists of the following fields:

p_{sent} - number of packets sent to PA
p_{rec} - number of packets received from PA
B_{sent} - number of bytes sent to PA
B_{drop} - number of bytes dropped due to rate limit
int_{sent} – inter-arrival time of packets sent to PA
int_{rec} – inter-arrival time of packets received from PA
mean_rto – smoothed mean of the ratio of number of packets sent to and received from PA
timestamp – timestamp when fields were last reset

5.1.1.2. Traffic Models

Most of communications between Internet hosts involve two-way traffic. The TCP protocol uses a two-way communication paradigm to achieve reliable delivery. During a TCP session, the data flow from source to destination host is controlled by the constant flow of acknowledgments in the reverse direction.

[4] *Foreign* in this context means the host that does not belong to the source network.

Thus, normal TCP communication is modeled by small values of the ratio of number of packets sent to a specific destination and number of packets received from this destination. Ideally, this ratio should be one, but network congestion and different TCP implementations using delayed and selective acknowledgements push it to slightly larger values.

Other communication protocols (UDP, ICMP) do not explicitly require reverse traffic for proper operation. Some applications that use these protocols expect responses from the peer host. The *ping* application sends out `ICMP_ECHO` packets and expects to receive `ICMP_ECHO_RESPONSE` from the peer host. *ICQ* and *NetMeeting* use UDP packets to convey text, audio and video between humans; during normal operation messages travel in both directions. *DNS* requests carried in UDP packets usually produce a DNS reply in the reverse direction. For non-TCP traffic, servers usually experience smaller amounts of reverse traffic than clients, since their service consists of streaming real-time data and occasionally receiving control packets or streaming requests. However, with regard to a specific destination, those servers send a fairly stable volume of traffic per time interval, and we use this feature to characterize normal non-TCP traffic.

Models of normal non-TCP traffic are created during the *training phase* before the DDoS defense system is started. During this phase the observation component gathers statistical records whose structure is similar to those gathered during the *attack detection phase*, but not necessarily in consecutive observation intervals. Traffic can be sampled instead of constantly monitored, and the observation interval need not be the same length as the observation interval during the attack detection phase. In addition to per-destination classification, in this phase traffic is also profiled per type, and records are kept in separate caches describing different types of non-TCP traffic. In our experiments, we differentiated between four types of traffic: TCP, UDP, ICMP and *other*.[5] Each incoming packet is classified per type and destination and a corresponding cache record is updated.

At the end of each interval, the cumulative traffic parameters (mean and standard deviation) for a specific destination PA are updated based on the data gathered during the interval. After the update, the cache is purged, and the statistics gathering process is restarted at a later time.

After the training phase is finished, a destination is classified as TCP, UDP, ICMP or other based on the traffic type of the majority of packets sent to this destination. If the dominant type of traffic is non-TCP, the cumulative parameters corresponding to it are stored in a file. This file is used to initialize models of non-TCP traffic prior to the recognition phase. Destinations that have no dominant type of traffic (i.e. no type of traffic is represented by more than 50% of the packets sent) are conservatively classified as TCP destinations. Records that have a small number of observations are discarded as statistically insignificant, and the corresponding destinations are treated as TCP destinations.

Classification of destinations as strictly one type (e.g., TCP or UDP but not both) will introduce errors in the detection process. It would be more appropriate to use the complete profile of the destination to detect the attacks, but this would consume four times more memory than the proposed approach, and would require online classification of incoming packets per traffic type and four times larger online statistics. The cost of various classification methods and the improvement they bring to classification accuracy will be investigated in this thesis.

Although the traffic models are generated prior to system startup, they need to be updated periodically to reflect new trends in network traffic, such as the deployment of a new service. The update of models

[5] We anticipate that more distinct traffic types may be identified during the research.

needs to be controlled to prevent attackers from training the system over time to regard the attacks as normal traffic. The automatic update of models of normal non-TCP traffic is also part of our future work.

5.1.1.3. Classification of Traffic Flows

The purpose of the flow classification is to detect if the associated destination experiences communication difficulties that could be a sign of a DDoS attack. During classification, flows are first matched with the models of normal traffic and classified as *compliant* or attack flows. Compliant flows are further examined and classified as either normal or transient flows based on their behavior.

The gathered flow statistics are the aggregate statistics for both TCP and non-TCP traffic in the flow. As a first classification step these statistics are compared with the more restrictive model of normal TCP traffic. The smoothed ratio of number of packets sent and number of packets received for a flow destination is compared with the maximum allowed packet ratio for TCP traffic. Smoothing is performed by exponentially averaging the sampled values of this parameter to accommodate temporary peaks. If the smoothed packet ratio for the specific destination is smaller than maximum allowed, the flow is classified as compliant. Otherwise, further classification is needed using the less restrictive models of normal non-TCP traffic. If the model exists for the corresponding destination, and observed parameter values are within limits defined by the model, the flow is classified as compliant. Otherwise, if no model exists or observed parameter values are outside limits defined by the model, the flow is classified as attack.

Flows that are classified as compliant are further checked to see if they are well behaved.[6] To facilitate this check, a *well-behaved* counter is associated with each cache record. The purpose of this counter is to slow down the recovery of flows that have been classified as attack flows and keep their records in the cache long enough to assure they are well behaved. The counter is originally unset, and can be modified after the flow has been matched with the model of normal traffic. The counter is set to 0 every time a flow is classified as attack; and incremented, if it is set, every time the flow is classified as compliant. Compliant flows that have an unset value of this counter are classified as normal, otherwise they are classified as transient. The value of the *well-behaved* counter indicates the number of consecutive observation intervals when the flow has been punished and behaved well. The counter is unset after the flow has endured an appropriate penalty.

5.1.2. Throttling Component

As a response to the attack, the throttling component restricts the allowed sending rate to the victim of the attack. Since the detection of the attack is unreliable and may have false-positives, rate limiting is a better-suited response than complete filtering. Filtering out all the traffic to the victim would greatly damage misclassified flows, whereas rate limiting still allows some packets to reach the destination and thus keeps connections alive. Allowing some attack packets through is acceptable, since the attack's overall impact depends on the volume of attack packets.

The throttling component defines the allowed sending rate to a particular destination based on the history of destination's behavior and current classification of its associated flow. The detection of the attack is based on incomplete observations and is not reliable. Therefore, the throttling component has the hard task of defining a strategy that should be able to effectively limit attacking flows and reduce the effect of

[6] This checking is necessary since compliance of the flow parameters with the model can stem from the restrictive rate limit placed on the flow. If this is the case, misclassifying this flow as "normal" would lead to its removal from the cache at the end of the current observation period, and to removal of the imposed rate limit. This would quickly open a new opportunity for the attack.

the attack on the victim, while at the same allowing misclassified flows to recover within a reasonably short time period. The trade-off here is between the potential harm to legitimate flows by applying a too-restrictive throttling strategy and the lower protection provided to the victim by designing a fast recovery mechanism.

When the flow is classified as an attack flow for the first time after a long period, its rate is limited to a fraction of the offending sending rate. The size of the fraction is specified by the configuration parameter `DEC_SPEED`. Subsequent classification of a flow as an attack restricts the rate further, according to the formula:

$$rateLimit = \min(rateLimit, rate_{actual}) * DEC\_SPEED * \frac{B_{sent}}{B_{sent} + B_{drop}}$$

$rate_{actual}$ is the realized sending rate for the flow during the previous observation interval. B_{sent} represents the number of bytes sent from this flow during the interval and B_{drop} is the number of dropped bytes because of the imposed rate limit. Thus the last factor in the equation describes the degree of misbehavior of the flow, and defines the restrictiveness of the rate-limit. Flows that have worse behavior are quickly restricted to very low rates, whereas this restriction is more gradual for better-behaving flows. The minimal rate limit that can be imposed is defined by the `MIN_RATE` configuration parameter so that at least some IP packets can reach the destination and trigger a recovery phase. Rate decrease progresses until the flow is classified as transient, at which point the slow recovery mechanism is triggered.

The recovery phase is divided into *slow-recovery* and *fast-recovery* based on the values of the well-behaved parameter associated with the flow. During the slow-recovery phase, a flow is penalized for having been classified as an attack flow by a linear increase in the allowed rate according to the formula:

$$rateLimit = rateLimit + INC\_SPEED * \frac{B_{sent}}{B_{sent} + B_{drop}}$$

The speed of the recovery is defined by `INC_SPEED` parameter. The duration of a slow-recovery phase is defined by two configuration parameters, `MIN_RECOVERY_PERIOD` and `MAX_RECOVERY_PERIOD`. The variable `period` is set randomly between these two values to avoid synchronization of flows in the case of "pulsing attacks." After the flow has been classified as transient for `period` consecutive observation intervals (detected by the well-behaved counter having values higher than `period`), the fast-recovery phase is triggered.

During the fast-recovery phase the rate is increased exponentially according to the formula:

$$rateLimit = rateLimit * \left(1 + \frac{B_{sent}}{B_{sent} + B_{drop}}\right)$$

The rate increase is limited by the `MAX_RATE` configuration parameter. As soon as the rate limit becomes greater than `MAX_RATE`, the recovery phase is finished, the flow is transferred from transient to normal class, and the rate limit is removed.

There is a trade-off between low values of `MIN_RECOVERY_PERIOD`, which facilitate fast recovery of misclassified flows, and vulnerability to "pulsing attacks." Higher values of `MAX_RECOVERY_PERIOD` reduce this vulnerability by allowing more values for the `period` variable. They also keep the flow in the cache and enforce the rate limit. There is also a trade-off between the high values of `MAX_RECOVERY_PERIOD` and the frequency of cache overflows, which reduce system performance.

The rate limiting of the throttling component is very similar to TCP's congestion control mechanism. It also performs an exponential decrease of the sending rate as a response to detected communication problems. Its slow-recovery period resembles the congestion-avoidance phase in the TCP during which the sending rate increases linearly, and the fast-recovery phase resembles the TCP's slow-start phase when the rate increases exponentially. A difference is that the rates regulated are the cumulative rates for the domain that correspond to many TCP and non-TCP flows. Another difference is that the speed of the rate change is guided by the flow's compliance to the imposed rate limit, thus the misbehaving flows endure a greater penalty.

The behavior of the system obviously depends on the proper parameter setting. The following section gives some guidance on the trade-offs involved, but the detailed evaluation of the effect of parameter values on system functioning is part of our thesis research.

5.2. Testing a DDoS Defense System

Denial-of-service attacks occur in distributed and complex environments and can involve vast numbers of participating machines. Reproduction of similar conditions in a testbed environment is a challenging task. Testing systems for DDoS defense must include reproduction of real denial-of-service attacks. However, all the traffic produced during this testing must be limited to testbed machines to protect the rest of the Internet from congestion and avoid inflicting great damage if the experiment is badly designed. Few testbeds include more than several machines in a sufficient physical proximity to satisfy the previous requirement. Thus, reproduction of the scale of the attack is infeasible, and scalability and cost of any DDoS defense system cannot be well assessed through experiments.

Also, denial-of-service attacks disrupt legitimate traffic, either by denying the service to the victim or by creating congestion on intermediate routers. To assess the benefit of a source-end-based DDoS defense system, we should measure the benefit or damage that such a system inflicts on legitimate traffic. This calls for generation or reproduction of legitimate traffic between the source network and the victim.

In our test runs the proposed DDoS defense system builds models of normal traffic from `tcpdump` traces collected at the exit router of the UCLA CSD network. The models of normal TCP traffic are likely to be location independent. The models of normal non-TCP traffic, on the other hand, must be generated from traces observed by the same source router that will later use them for flow classification.[7] Thus the source router in our testbed has to play the role of UCLA CSD exit router to be able to use these models. Also, during the test runs the source router must observe normal background traffic whose volume and dynamics correspond closely to the real traffic between UCLA CSD and the rest of the Internet. We handle these problems by profiling a different set of `tcpdump` traces than that used for model generation, and replaying them during the experiment.

Ideally, every line in a `tcpdump` trace should produce a packet that passes through the source router on its way to the destination and updates traffic statistics. However, the volume of traffic observed by the UCLA CSD exit router is larger than testbed machines can efficiently handle; this replay process would quickly use up the source router's resources. To scale down this problem, we assume that the real router has a sufficient amount of resources to relay the traffic as efficiently during the attack as in normal operating conditions. Thus, the only flows that feel the impact of the attack and the source router's response are those flows that are either destined for the victim or generated by the victim. Only these

[7] Different "exit" routers are likely to observe different volumes of normal non-TCP traffic. For example, the level of normal UDP traffic is much higher for a domain containing large DNS-zone name-servers than for a college network.

flows need to be replayed through the network, since the attack and the response are likely to cause a dropping of their packets, change of packet timing and change in the duration of connections. The rest of the `tcpdump` records can be read in directly from the trace and used to update the corresponding statistics.

The replay of flows associated with the victim starts with the choice of victim IP address and extraction of corresponding packet details from the trace. Two files are generated as the result of this process: `victim.trace`, which contains information for all packets sent by the victim to UCLA CSD hosts, and `ntg.trace` that contains all the information for UCLA CSD traffic to the victim. Two testbed machines are then chosen to play the role of the *victim* and the *normal_traffic_generator*. They are placed on different interfaces of the source router and replay the traffic from the generated files during the test runs.

Non-TCP packets can be described by timing, size, source and destination IP addresses and port numbers; and they are sent during replay according to the timing in the trace. TCP packets are further dependent on the state of the connection they belong to. To reproduce those connections correctly, we need to set up a TCP session between the victim and the normal_traffic_generator for every TCP connection in the trace. TCP connections are initiated at the time specified by the trace, and data is delivered to the connection according to the timing of trace packets. TCP itself handles their exact sizes and the times they are injected into network through TCP's congestion window and acknowledgment mechanism. If the sending of some data packets is delayed, we shift the generation time of subsequent data packets from the same connection by this delay. In the test runs that do not involve attack packets, traces gathered during the replay process resemble, to a great extent, the real `tcpdump` traces used for their reproduction. During the attack, the TCP control mechanism responds to packets that are lost due to rate limiting or congestion, by regulating the sending of new packets. We measure this impact and use this measurement as one of the indicators of the effect of our system on normal traffic.

5.3. Experimental Results

Our testbed consisted of a single source router connected to eight testbed machines via Ethernet links. Out of those, one machine was designated to act as the normal_traffic_generator and the other seven are used for the attack. On another interface, the source router is connected to a machine acting as a victim. The source router is a Linux router, and it uses a Netlink interface to communicate with our DDoS defense system which is implemented as a user application on the same machine. All testbed machines are identically configured with 1.4 GHZ AMD Thunderbird CPUs, 1GB RAM, and 100Mbs Ethernet NICs. To model the real network in which the router has greater resources than the victim, we artificially reduce the available bandwidth in the link from the source router to the victim. We generate the DDoS attacks by using real DDoS attack tools, such as TFN and Trinoo, and we intentionally test the attacks that invoke responses from the victim, such as TCP SYN and ping flood attacks. We use 5 seconds as the observation interval, 3 as the maximum allowed packet ratio, 0.5 for the value of `DEC_SPEED` and `INC_SPEED` parameter, 60 for `MIN_RECOVERY_PERIOD`, 300 for `MAX_RECOVERY_PERIOD`, 20 for `MIN_RATE` and 100000 for `MAX_RATE`.

To measure the effectiveness of our system and the impact it has on attack and normal traffic, we ran several experiments in the identical setup, varying the level of normal and attack traffic. Figure 3 presents the result of two test runs. The defense system is started at the beginning of every run and given time to initialize. After 10 seconds, trace replaying is triggered on the victim and the normal_traffic_generator. The attack is started at 210 seconds and it lasts until 330 seconds. We continue measurement for 300 seconds after the end of the attack. During the first run, normal traffic to the victim was being replayed from `tcpdump` traces, and was on the order of several packets per second, containing no more than a thousand bytes each. The second run interleaved the same trace traffic and an FTP transfer of a 6.5MB

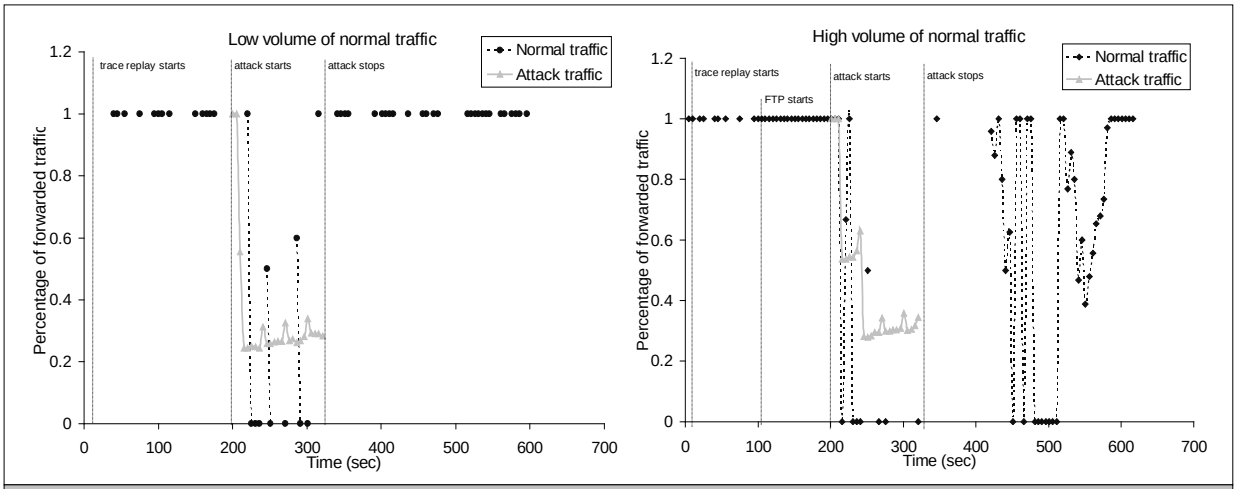

Figure 3: Effect of the defense system on legitimate and attack traffic.

file from normal_traffic_generator to the victim. The transfer is started 110 seconds after the beginning of the run. Figure 3 depicts the percentage of packets that were allowed to pass through the filter in every observation interval. In the case of low volume of background traffic, the attack is suppressed more efficiently and quickly (within three observation intervals) to about 30% of its capacity, where it remains until the end of the attack. At the same time, normal traffic suffers certain damage in some observation intervals but restores quickly after the attack has passed. In the case of high background traffic, the attack is suppressed more gradually, taking nine observation intervals to reach the 30% level, and normal traffic suffers greater damage in the recovery phase since its desired transfer rate does not comply with the restrictive rate limit. The difference in the speed of the response comes from the fact that high TCP background traffic in the second run generates a large number of response packets from the victim, which facilitate the classification of the offending flow as compliant (210 to 230 seconds) and slow down rate limiting. Figure 4 shows the change of rate limit over these two runs and the desired and achieved rate values for the total flow to the victim (normal and attack traffic combined). In the case of high-volume traffic, the sending rate violates the linearly increasing rate limit at 450 seconds, thus slowing the recovery of normal traffic. The rate limit also reflects the misclassification of the attack as a compliant flow in the second run, from 210 to 230 seconds, where it increases linearly.

We then evaluated the impact of the proposed DDoS defense system on the duration of legitimate TCP

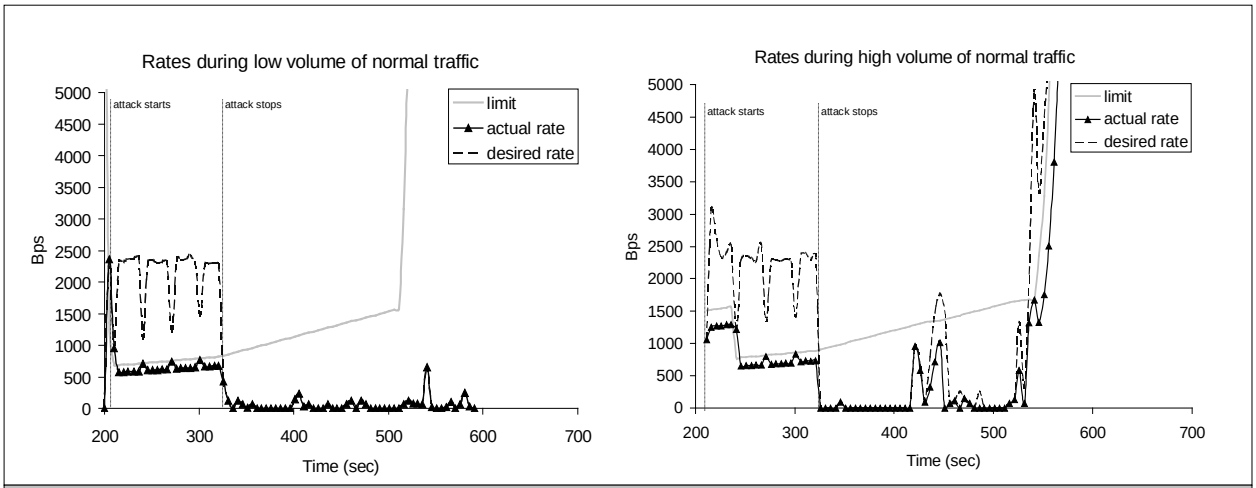

Figure 4: Rate limit, the desired and actual rate for low and high volume of normal traffic.

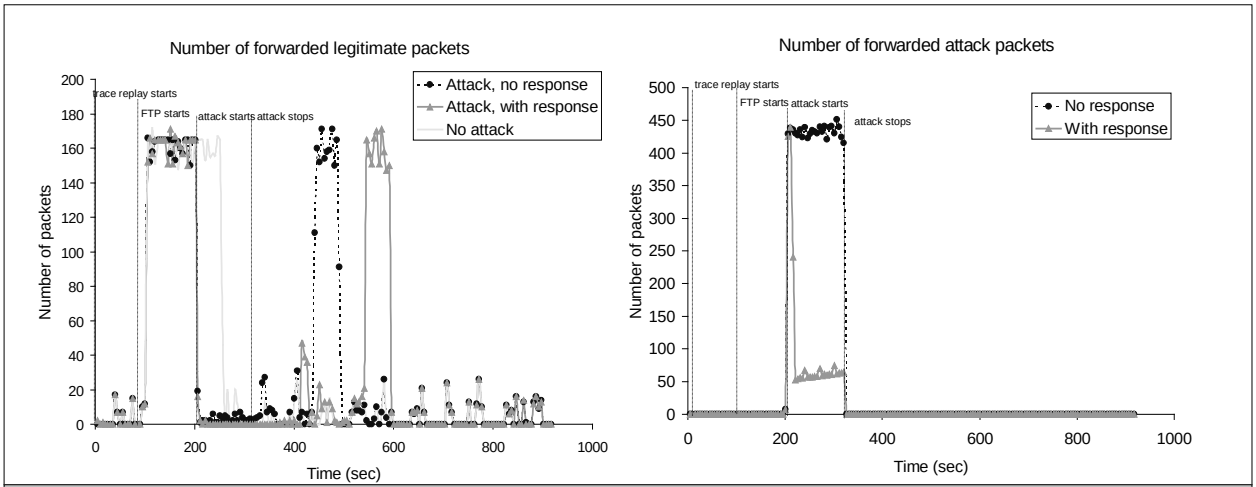

Figure 5: Number of forwarded packets in the case with and without response to the attack.

connections. We generated a high volume of background traffic by interleaving the FTP transfer of the same 6.5 MB file with replayed traces, and observed the interaction of the normal with the attack flow and with the imposed rate limit. The number of normal and attack packets allowed to pass the filter in each observation interval for three test runs is shown in Figure 5. In all runs, the FTP transfer is started at 110 seconds. In the first run there is no attack and no rate limit is imposed. In the second run the attack occurs from 210 to 330 seconds, and our system is not deployed on the router. In the third run, our system is deployed and does respond to the attack by rate limiting the total flow to the victim. The FTP transfer in the first run ends at approximately 250 seconds. The onset of the attack in the second run takes up the bandwidth from the normal flow, and FTP reduces the sending rate in response to lost packets. After the end of the attack, normal communication is resumed and FTP increases the sending rate quickly and finishes transfer at 500 seconds. In the third run, the restrictive rate limit delays the recovery of FTP flow by an additional 100 seconds, and the transfer finishes at 600 seconds. The delay stems from the aggressive increase in the sending rate by FTP, which does not comply with a linear increase of rate limit in the slow recovery phase. On the other hand, the benefit of rate limiting the attack flow is evident from the second graph; it is quickly reduced to one tenth of its original sending rate, and it remains there until the end of the attack.

We further evaluated the improvement in attack detection by using models of normal non-TCP traffic

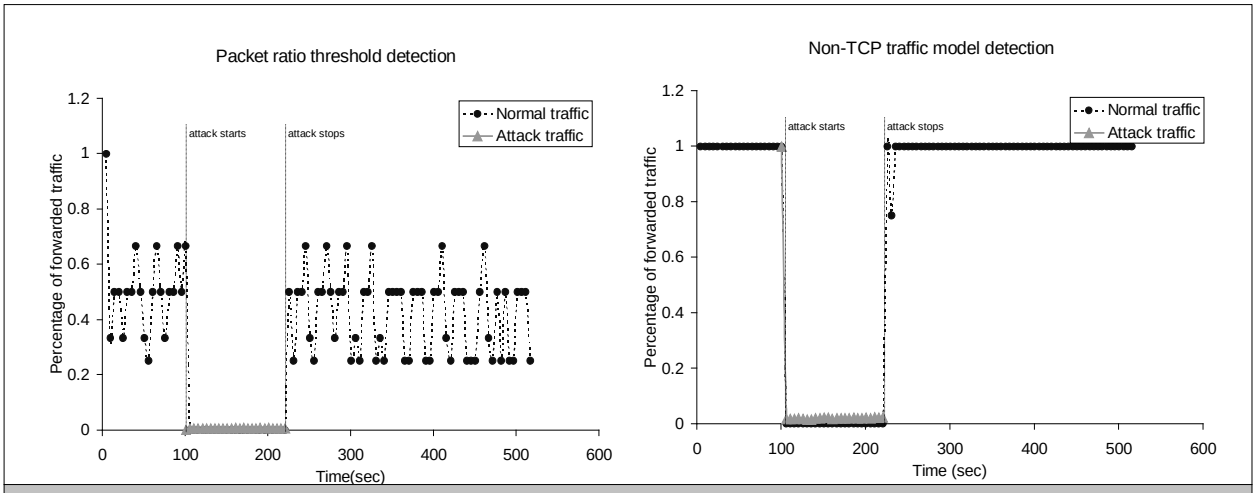

Figure 6: Effect of different detection mechanisms on normal and attack traffic

compared to detection based on the disproportion in numbers of sent and received packets. We targeted an attack on an UDP destination and measured the levels of legitimate and attack traffic that were allowed to pass the filter during the attack and recovery phase. Trace replaying is triggered 10 seconds after the start of the run. The attack is started at 100 seconds and lasts until 210 seconds. We continue measurement for 300 seconds after the end of the attack. In the first run we use a disproportion in the number of sent and received packets to classify the flow as an attack flow, whereas in the second run we use precalculated models of normal non-TCP traffic for the given destination. Figure 6 presents the results of the experiment. In both runs, the throttling strategy is identical, and the attack traffic is quickly detected and dropped. When a fixed packet ratio threshold is used for detection, legitimate UDP packets are dropped constantly, since they do not produce enough reverse packets to comply with the allowed packet ratio. The drop rate is around 50% even when the attack is not going on. In the case where models of non-TCP traffic are used, legitimate traffic is dropped only during the attack and for a short time after the attack has finished, due to the slow recovery phase.

Experimental results show that the proposed mechanism can effectively detect and handle individual DDoS flows at a router close to their source by comparing their two-way traffic parameters to models of normal activity. Even in the presence of a high level of non-attack traffic, the system can throttle attack flows down within a few seconds, although the response is slowed down if non-attack traffic invokes a lot of replies from the victim. If applied to all flows in a DDoS attack, this form of throttling would make the attack much less effective. While the mechanism also throttles good flows for the same destination, it does not entirely shut them off.

The recovery of legitimate TCP connections after the attack can be slowed down by the system. The delay imposed on legitimate connections depends on the length of the slow-recovery phase and the aggressiveness of connections. Less aggressive connections experience small or no delay since they do not try to send over the imposed rate limit, which facilitates a faster increase of the rate limit. More aggressive connections violate the rate limit, which in turn slows down recovery process, and is amplified by the conservative reaction of the TCP congestion control mechanism to lost packets. The behavior of the system can be tuned by changing the values of `MIN_RECOVERY_PERIOD` and `MAX_RECOVERY_PERIOD` configuration parameters. Lower values of these parameters facilitate fast recovery of legitimate TCP connections. However, they also endanger the victim’s recovery during the attack, because the attack flows can recover quickly and deliver another full blow at the victim before their rate is limited again. The system is friendly to non-TCP traffic and does not misclassify it as an attack.

While the results are promising, they point to the need for further research. More aggressive throttling strategies need to be deployed in cases of continued bad behavior to avoid recurring attacks. Distinguishing between normal and attack flows within the aggregate flow to the victim would offer better fairness to normal flows. It would also facilitate a lower delay in recovery of legitimate TCP connections after the attack, while still providing high protection to the victim. A separate strategy should be defined to handle non-TCP flows whose model does not exist in the model cache, and an automated mechanism for model updating should be designed to capture dynamics in Internet communication.

6. Attack Detection

The proposed system monitors a set of parameters that describe peer behavior and compares them to predefined models to detect a DDoS attack on the peer. A goal of this thesis is to resolve the following issues connected to the detection process:

1. **Choice of monitored parameters** - Monitored parameters should offer enough information to detect difficulty in the peer's operation. On the other hand, the parameter space must not be so large that the

monitoring and update process degrades network performance. We plan to investigate this tradeoff and find the detection reliability that each set of parameters can offer. Also, monitored parameters should enable the system to differentiate individual legitimate flows destined for the victim from the attack flows, even in the face of IP spoofing. We will investigate different strategies for flow identification and the benefit that elimination of IP spoofing would add to identification accuracy.

2. **Creation and update of models -** Predefined models should describe normal peer behavior. This thesis will investigate various modeling approaches and evaluate the detection accuracy that each of them offers. In addition to this it will define an automatic strategy for model update. This strategy must be resilient to attackers' attempts to train systems to misclassify attack flows as normal or the normal flows as attack.[8] It should also define appropriate system behavior when a peer model cannot be found in the model database.

3. **Cooperation with other source routers -** In the case of asymmetric routing between source network and the victim, the source router might have incomplete observation of parameters. In such a case, cooperation with other source routers in the same administrative domain would benefit the detection accuracy. We plan to assess this benefit and define secure and low-cost communication protocol.

4. **Cooperation with the victim -** We will investigate the ways to securely and reliably communicate with the victim, which should result in the corresponding improvement of attack detection.

5. **Recurring attacks -** We will investigate different techniques for early detection of recurring attacks.

7. Attack Response

The unreliable detection process imposes the need for a carefully designed attack response. The rate limiting decisions need to be adjusted based on direct (through communication) or indirect (through observed parameters) feedback from the victim. The goal of this thesis is to resolve the following issues connected to the attack response:

1. **Effectiveness vs. fairness of response -** The response to the attack should be able to effectively constrain attacking flows and reduce the effect of the attack on the victim, while at the same allowing misclassified flows to recover within a reasonably short time period. We plan to investigate different response strategies and define the best-suited strategy depending on the reliability of the detection mechanism.

2. **Traceback of the attack -** The deployment of the system at the source network should offer a good opportunity for attack traceback. We will investigate different traceback approaches and assess their performance, in the case when IP spoofing is allowed. We will further assess additional benefits if the IP spoofing is eliminated.

8. Security

Attackers have so far demonstrated the ability and willingness to develop more and more sophisticated attack tools that defeat current approaches to combating DDoS attacks. It is therefore expected that they will try to defeat the system proposed in this thesis or to misuse it for denial-of-service attacks. Special attention must be paid to secure the system against such attempts.

[8] This could be done by the attacker sending gradually more and more UDP traffic over a long time period to train the system to regard such high levels of UDP traffic as normal for a given destination.

The system operates autonomously and does not rely on communication with other entities. Thus its operation can only be affected by packet flows observed by the source router. An attacker that can control the reverse flow (from the rest of the Internet to the source network) could either prevent detection of the attacks by generating fake response packets, or even deny any service to clients from the source network by dropping the legitimate response packets. Spoofing of packets is still possible at a large number of domains, and therefore the former threat seems very realistic. Wider deployment of mechanisms that prevent spoofing ([23], [24], [31]) may help reduce this problem.

The attackers could also perform “pulsing attacks.” Once the attack is suppressed by rate limiting, they could abort it for a sufficient interval, and then restart it with full force. The victim would thus be periodically overwhelmed with attack packets for the short interval, until the DDoS system detects and limits their flows. The random choice of a penalty period can help to break up the synchronization of attack flows from different source networks, thus possibly taking the full force out of the attack. However, more sophisticated techniques are needed to solve this problem completely.

The attackers could choose to merely use up a lot of the victim's resources, thus degrading the service instead of denying it completely. Our system would not be able to detect such attacks since there would not be any signs of difficulties visible at the source end. The extension of the system with a capability to act on the request of the victim, if sufficiently secured, could handle this problem.

Finally, the attackers might try to perform a DDoS attack on the source router. While the size of memory structures is limited and cannot be controlled by the attacker, it is possible that a vast number of packets from the source network would exhaust a router’s processing resources. However, the gain from this attack is small. By disabling the source router, attackers lose their connectivity to the world and cannot perform any more attacks. Furthermore, the administrative domain that is sufficiently security aware to deploy a source router mechanism, would be likely to have various traceback mechanisms to quickly detect and stop attack machines.

9. Implementation Considerations

9.1. Partial Deployment

If the system proposed here were deployed in a small number of source networks, the DDoS threat would not be significantly reduced. Since only a few source networks would throttle attack flows, attackers would merely exploit networks where the system was not deployed. Thus, a high degree of deployment is a condition for high effectiveness. In our thesis we will investigate different deployment strategies and evaluate the security guarantee that each offers.

To encourage source networks to deploy the DDoS defense system proposed in this thesis, we attempt to introduce minimal changes to the source router and offload the processing and storage tasks to separate units, as shown in Figure 1. The following are necessary functions that the source router should provide: (1) passing of the whole packets or packet headers to the observation component on a separate link and (2) implementation of per-destination rate limits. The source router should further be able to communicate with the throttling component to obtain new rate limits and implement them in its filtering mechanism. It is desirable for the source router to also provide some feedback as to which packets were dropped due to rate limits, to facilitate monitoring of the functioning of the defense system. To the best of our knowledge, today's routers already possess this functionality, or can be easily extended to provide it.

Additional incentives to deploy the defense system might be the low cost and easy installation of its components, small or no changes to the existing routers, good performance, and low impact on normal functioning of the source network. However, these factors merely make deployment less painful. The

source networks do not protect their own nodes from DDoS attacks by deploying this or similar systems, and thus currently have low motivation to do so. However, it can be expected in the future that legal or governmental factors will lead network operators to install systems that make their networks harder to use as attack points. The Internet has become such a vital part of business that denial-of-service attacks and network outages cost companies immense amounts of money, and something will be done to ensure that those connecting to networks behave well. In such an environment, implementation of the proposed DDoS defense system would be highly beneficial to the source networks as well as to their peers.

9.2. Deployment on Core Routers

Deploying a DDoS defense system on core routers would have the significant advantage of policing a large fraction of the total Internet traffic; thus a large coverage could be achieved with a small number of deployment points. However, core routers must operate at high speeds, and they have very few resources to spare for tasks other than routing. The memory and processing overhead that a defense system introduces per packet would have to be quite small in order to avoid performance degradation. On the other hand, faster processing and smaller state lead to poor detection of the attacks, and misclassification of flows at the core router would damage a significant portion of the legitimate Internet traffic.

We plan to investigate a separate design of a DDoS defense system that would be implemented on a core router. We will assess the ability of such a system to independently detect attack and design an appropriate response strategy.

10. Summary of Thesis Goals

In this section we summarize the goals we plan to achieve in this thesis.

1. **Analyze DDoS attacks** - We plan to examine the existing tools for DDoS attacks and identify vital features of these tools (those features that cannot be omitted without endangering the success of the attack). We will investigate the impact of each attack on the victim, depending on the number of attack machines, victim resources, the attack strategy and type of the attack. This phase should help us understand the problem fully so that we can design effective solutions.

2. **Investigate monitoring strategies -** We will investigate various monitoring and attack detection strategies and evaluate detection reliability and cost of each. We will investigate different methods of separating legitimate flows from attack flows, investigate various modeling approaches, define reliable automatic strategy for model update, and define the appropriate system behavior when a peer model cannot be found in the model database. We will also investigate different techniques for early detection of recurring attacks.

3. **Investigate response strategies -** We will investigate various response strategies that will quickly constrain the attack and offer satisfactory fairness to legitimate flows. We will also investigate traceback strategies with and without IP spoofing in the network.

4. **Investigate cooperation possibilities -** We will investigate cooperation within the source network and with the victim, and evaluate its benefits and cost.

5. **Design and implement the system -** Based on our investigations we will choose the best monitoring and response strategy and design and implement a system that effectively stops DDoS attacks at the source network. This system will operate independently. As an extension, we will add cooperation modules that will increase the effectiveness of the system. The first implementation will be done in a Linux software router, and extensive performance measurements will be performed. Results from

these measurements should help us correct and refine our design. The improved system will be implemented in Intel's IXA IXP programmable router.

6. **Investigate deployment strategies** - We will investigate various partial deployment strategies at different points in the Internet and evaluate the coverage that each of them offers based on the deployment degree. We will also investigate deployment of a (possibly significantly modified) system at core routers.

7. **(Optional) Simulation** - It is possible that the scalability of our system can only be evaluated through simulation. If so, we will develop a simulation of the proposed system.

Figure 7 gives the anticipated timeline of the thesis work.

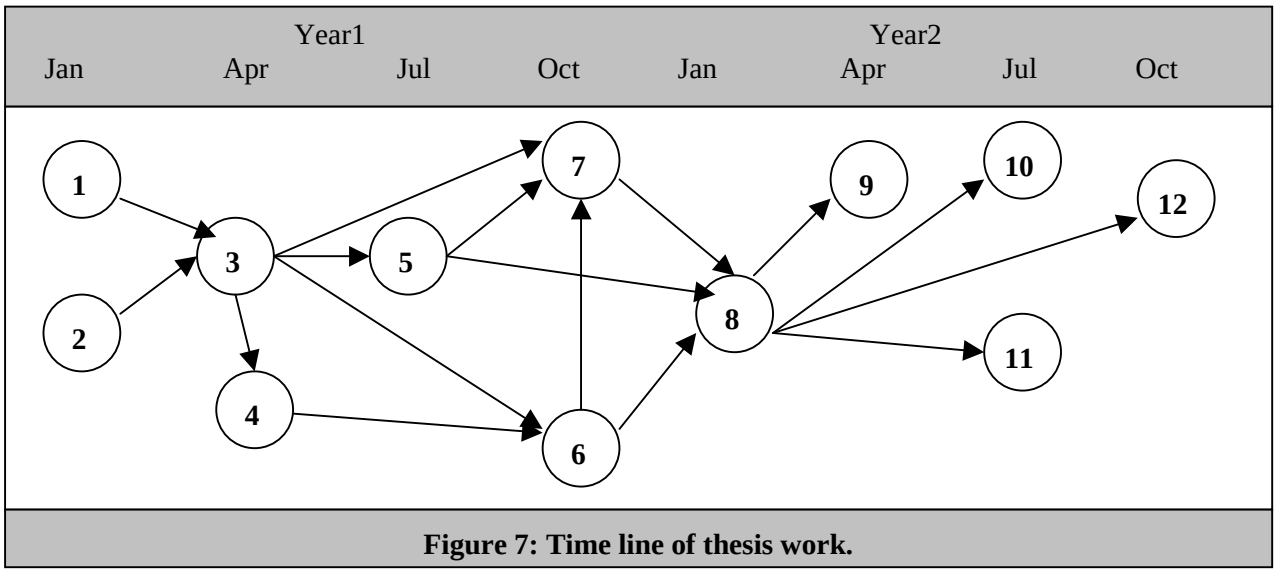

Figure 7: Time line of thesis work.

1. Analysis of Internet traffic patterns
2. Analysis of DDoS attacks
3. Initial implementation of a software router
4. Experimentation
5. Investigation of monitoring strategies
6. Investigation of response strategies
7. Investigation of cooperation possibilities
8. Security design
9. Implementation in programmable router (joint work with other research group members)
10. Investigation of partial deployment strategies
11. Investigation of core router deployment
12. Simulation (optional)

11. Conclusion

Denial of service attacks present a serious threat to all Internet hosts, and design of an incrementally deployable system that can effectively detect and constrain them is necessary. In this Ph.D. thesis such a system that is deployed at the source network. This thesis will investigate various components of the proposed system and evaluate benefits and cost of different design approaches. We will implement this system in a software router and in a real router. We will define the best incremental deployment strategy and investigate a separate design that could be deployed in core routers. If necessary we will simulate the

proposed system to evaluate its scalability. We hope that when this thesis is completed, it will offer a viable, low-cost and effective solution that will remove the DDoS attack threat from the Internet.

12. References

[1] Cisco, "NetRanger Overview," http://www.cisco.com/univercd/cc/td/doc/product/iaabu/csids/csids1/csidsug/overview.htm

[2] Cisco, "Characterizing and Tracing Packet Floods Using Cisco Routers," http://www.cisco.com/warp/public/707/22.html#2b

[3] K. A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R. A. Olsson, "Detecting Disruptive Routers: A Distributed Network Monitoring Approach," Proceedings of the 1998 IEEE Symposium on Security and Privacy, May 1998.

[4] J.R. Hughes, T. Aura, and M. Bishop, "Using Conservation of Flow as a Security Mechanism in Network Protocols." Proceedings of the 2000 IEEE Symposium on Security and Privacy, May 2000.

[5] P.G. Neumann and P.A. Porras, "Experience with EMERALD to DATE," 1st USENIX Workshop on Intrusion Detection and Network Monitoring, April 1999.

[6] U. Lindqvist and P.A. Porras, "Detecting computer and network misuse through the Production-Based Expert System Toolset (P-BEST), " Proceedings of the 1999 IEEE Symposium on Security and Privacy, May 1999.

[7] P.A. Porras and P.G. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," Proceedings of the Nineteenth National Computer Security Conference, October 1997.

[8] P.A. Porras and A. Valdes, "Live traffic analysis of TCP/IP gateways," Proceedings of the Symposium on Network and Distributed System Security, March 1998.

[9] S.Dietrich, N. Long and D. Dittrich, "An Analysis of the "Shaft" distributed denial of service tool," http://www.adelphi.edu/~spock/shaft_analysis.txt

[10] S. Liu, Y. Xiong, and P. Sun, "On Prevention of the Denial of Service Attacks: A Control Theoretical Approach," IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop, June 2000.

[11] Computer Incident Advisory Capability, "Network Intrusion Detector Overview," http://ciac.llnl.gov/cstc/nid/intro.html.

[12] MimeStar.com, "SecureNet PRO Feature List," http://www.mimestar.com/products/

[13] CERT Coordination Center, " CERT Advisory CA-2001-19 'Code Red' Worm Exploiting Buffer Overflow In IIS Indexing Service DLL," http://www.cert.org/advisories/CA-2001-19.html

[14] Internet Security Systems, "Intrusion Detection Security Products," http://www.iss.net/securing_e-business/security_products/intrusion_detection/index.php

[15] CERT Coordination Center, " CERT Advisory CA-1999-17 Denial-Of-Service Tools," http://www.cert.org/advisories/CA-1999-17.html

[16] NFR Security, "NFR Network Intrusion Detection," http://www.nfr.com/products/NID/

[17] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, August 1993.

[18] S.Floyd et al, “Pushback Messages for Controlling aggregates in the Network," Internet draft, Work in progress, http://search.ietf.org/internet-drafts/draft-floyd-pushback-messages-00.txt, July 2001.

[19] T. M. Gil and M. Poleto, "MULTOPS: a data-structure for bandwidth attack detection," Proceedings of 10th Usenix Security Symposium, August 2001.

[20] Internet Traffic Archive, "LBL-PKT Traces," http://ita.ee.lbl.gov/html/contrib/LBL-PKT.html

[21] Internet Traffic Archive, "DEC-PKT traces," http://ita.ee.lbl.gov/html/contrib/DEC-PKT.html

[22] National Laboratory for Applied Network Research, "NLANR Packet Header Traces," http://pma.nlanr.net/Traces/Traces/long/auck/2/

[23] J. Li, J. Mirkovic, M. Wang, P. Reiher and L. Zhang, "SAVE: Source Address Validity Enforcement Protocol," Proceedings of INFOCOM 2002, June 2002.

[24] P. Ferguson and D.Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing," RFC 2827.

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

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

[27] D. Dittrich, "The 'Stacheldraht' distributed denial of service attack tool," http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt

[28] D. Dittrich, "The 'mstream' distributed denial of service attack tool," http://staff.washington.edu/dittrich/misc/mstream.analysis.txt

[29] S. Bellovin, M. Leech and T. Taylor, "ICMP Traceback Messages," Internet draft, Work in progress, http://search.ietf.org/internet-drafts/draft-ietf-itrace-01.txt, October 2001.

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

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

[32] CERT Coordination Center, "CERT Advisory CA-2001-01 Ramen Worm," http://www.cert.org/advisories/CA-2001-01
[33] H. Burch and W. Cheswick, "Tracing Anonymous Packets to Their Approximate Source," Proceedings of 2000 Systems Administration Conference, December 2000.
[34] R. Stone. "CenterTrack: An IP Overlay Network for Tracking DoS Floods," 9th USENIX Security Symposium, August 2000.
[35] H. Lee and K. Park, "On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack." Proceedings of Infocom 2001, Anchorage, Alaska, April 2001.
[36] K. J. Houle and G. M. Weaver, "Trends in Denial of Service Attack Technology," CERT Coordination Center Report. http://www.cert.org/archive/pdf/DoS_trends.pdf