THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Multifaceted Defense Against Distributed Denial of Service Attacks: Prevention, Detection, Mitigation

ZHANG FU

Division of Networks and Systems
Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2012

Multifaceted Defense Against Distributed Denial of Service Attacks: Prevention, Detection, Mitigation

Zhang Fu

ISBN: 978-91-7385-733-8

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie 3414
ISSN 0346-718X

Technical report 85D
Department of Computer Science and Engineering
Distributed Computing and Systems

Division of Networks and Systems
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31-772 10 00

Author e-mail: zhafu@chalmers.se

Printed by Chalmers Reproservice
Gothenburg, Sweden 2012

ABSTRACT

Distributed Denial of Service (DDoS) attacks can be so powerful that they can easily deplete the computing resources or bandwidth of the potential targets, by flooding massive packets. Internet infrastructures and network applications, including social services and communication systems for emergency management, are under the threat of the DDoS problem. This thesis aims at providing efficient methods which can detect and mitigate DDoS attacks, meanwhile keeping the network performance degradation as little as possible.

Dealing with DDoS attacks is challenging, due to their multifaceted properties: dynamic attack rates, various kinds of targets, big scale of botnets, etc. Multifaceted nature of DDoS attacks justifies the need for multifaceted defense. Thus we address the DDoS problems from different aspects. In particular, in the thesis we present an adaptive *port-hopping* method to address application-level DDoS problems. The method enables multiparty applications to communicate via ports changed periodically. Thus, the adversary cannot effectively attack the communication ports of the targets. The proposed method can deal with clock drifts among the communication parties without the need of acknowledgments or time server. To address the bandwidth-flooding attacks, in the thesis, we propose and present SIEVE, a lightweight distributed filtering method. Depending on the adversary's ability, SIEVE can provide a standalone filter for moderate adversary models and a complementary filter which can enhance the performance of strong and more complex methods for stronger adversary models. SIEVE uses an overlay network to form a distributed "sieve" and uses *lightweight authenticators* (e.g. source IP addresses) to filter packets. SIEVE includes also a simple solution to protect connection setup procedures between legitimate clients and protected servers, which can also be applied to address the Denial-of-Capability (DoC) problem. In this thesis we present how to complement network-capability mechanisms by addressing the Denial-of-Capability problem. Mitigating DDoS attacks are challenging not only for the end hosts, but also for the network. By building on earlier work and improving on distribution of control aspects, a proactive method, which we call *CluB*, is proposed in this thesis to mitigate DDoS attacks. The method balances the effectiveness-

overhead trade-off by addressing the issue of granularity of control in the network. *CluB* can collaborate with different routing policies in the network, including contemporary datagram options. We estimate the effectiveness of the method and also study a set of factors for tuning the granularity of control. The thesis also studies the problem of monitoring high-speed traffic and detecting DDoS-related anomalies in a data streaming fashion to offer a detection at an early stage in the core network, thus activating appropriate DDoS mitigations only when necessary. We propose an IP-prefix based aggregation method to monitor and detect DDoS-related anomalies. Furthermore, we investigate the design space of combining parallel-distributed data streaming with both online detection and baseline profile maintenance, and give detailed solutions for achieving this. The proposed data streaming based DDoS defense solutions are implemented upon a parallel-distributed data stream engine.

Keywords: Distributed Denial-of-Service, System Design, Network Security, Distributed Computing, Data Streaming, Overlay, DDoS Detection

Acknowledgments

First of all, I would gratefully and sincerely appreciate my supervisors: Prof. Marina Papatriantafilou and Prof. Philippas Tsigas. Their inspiring guidance, rich experience and sustained encouragement enabled me to develop an intensive understanding of my research area. Without the generous help of my supervisors, this thesis would not have been possible.

I am honored to have Prof. Felix Freiling from University of Mannheim as my opponent. I thank Prof. David Sands, my examiner, for his kind support and helpful suggestions during the discussions in my PhD study follow-up meetings. I thank Vincenzo Gulisano, Mar Callau-Zori, Prof. Ricardo Jiménez-Peris and Prof. Marta Patiño-Martínez from Universidad Politécnica de Madrid for the collaboration and discussions.

I would like to thank the Swedish Civil Contingencies Agency (MSB) for financial support to my PhD employment. I thank the European Union Seventh Framework Program for funding the SysSec project.

I would like to give my appreciation to the members of Distributed Computing and Systems group: Elad Michael Schiller, Magnus Almgren, Olaf Landsiedel, Andreas Larsson, Daniel Cederman, Georgios Georgiadis, Nhan Nguyen Dang, Farnaz Moradi, Ioannis Nikolakopoulos, Bapi Chatterjee, Valentin Tudor. It is my honor to be a member in such a great research group. I enjoy the excellent working environment and many fruitful discussions, especially in the weekly group seminar.

I am very grateful to all my colleagues in the Department of Computer Science and Engineering. The atmosphere has always been a perfect source of

motivation. I feel lucky and happy to work here. And I really had a nice time with people of our department. They have been always friendly and supporting. Especially, I want to express my gratitude to Peter Lundin, Eva Axelsson, Erland Jonsson, Jan Jonsson, Tomas Olovsson, Wolfgang John, Pierre Kleberger, Dennis Nilsson, Ulf Larson, Phu Phung, Vilhelm Verendel, Ming Xiao and Ran Ji.

I also would like to thank my friends Wei Wei, Shan Liu and Feng Liu. Thanks to them for their good comments on this thesis.

Last, I want to thank my parents, without whom I would never have been able to achieve so much. I cordially thank my sister for taking the responsibility of taking care of my parents during my time studying abroad. I'm very proud to be her brother. I especially wish to express my love to Fang Su who only knows the real price of this thesis as we suffered and paid it together. I thank for her endless love, patience, and understanding.

Zhang Fu
Göteborg, August 2012

List of Appended Papers

I **Zhang Fu**, Marina Papatriantafilou, Philippas Tsigas, "*Mitigating Distributed Denial of Service Attacks in Multiparty Applications in the Presence of Clock Drifts*," in *IEEE Transactions on Dependable and Secure Computing,* Volume 9, Number 3, pages 401-414, 2012. A preliminary version of the paper was published in the proceedings of 27^{th} *IEEE International Symposium on Reliable Distributed Systems (SRDS)*, pages 63-72 Napoli, Italy, 6-8 October, 2008

II **Zhang Fu**, Marina Papatriantafilou, Philippas Tsigas, "*Off-the-Wall: Lightweight Distributed Filtering to Mitigate Distributed Denial of Service Attacks*," in the proceedings of 31^{st} *IEEE International Symposium on Reliable Distributed Systems (SRDS)*, Irvine, California, 8-11 October, 2012

III **Zhang Fu**, Marina Papatriantafilou, Philippas Tsigas, Wei Wei "*Mitigating Denial of Capability Attacks Using Sink Tree Based Quota Allocation*," in the processings of 25^{th} *ACM Symposium on Applied Computing (SAC)*, pages 713-718, Sierre, Switzerland, 22-26 March, 2010

IV **Zhang Fu**, Marina Papatriantafilou, Philippas Tsigas, "*CluB: A Cluster Based Framework for Mitigating Distributed Denial of Service Attacks*," in 26^{th} *ACM Symposium on Applied Computing (SAC)*, pages 520-527 TaiChung, Taiwan, 21-24 March, 2011

V Vincenzo Gulisano, **Zhang Fu**, Mar Callau-Zori, Ricardo Jiménez-Peris, Marina Papatriantafilou, Marta Patiño-Martínez, "*STONE: A Stream-based DDoS Defense Framework*" Technical Report no. 2012-07, ISSN 1652-926X, Chalmers University of Technology, 2012

List of Acronyms

AS Autonomous system
BGP Border Gateway Protocol
CAIDA The Cooperative Association for Internet Data Analysis
DDoS Distributed Denial of Service
DoC Denial of Capability
DoS Denial of Service
ECR Egress Checking Router
EH Exponential Histogram
EIR Ingress Checking Router
GTT Graphical Turing Test
HTTP Hypertext Transfer Protocol
ICMP Internet Control Message Protocol
IPFIX Internet Protocol Flow Information Export
ISP Internet Service Provider
MAC Massage Authentication Code
MSS Maximum Segment Size
NID Network Intrusion Detection
PCA Principal Component Analysis
RTT Round Trip Time
SPE Stream Processing Engine
SSH Secure Shell
SSL Secure Sockets Layer
SUNET Swedish University Computer Network

TCP Transmission Control Protocol
UDP User Datagram Protocol

Contents

Abstract i

Acknowledgments iii

List of Appended Papers v

I INTRODUCTION 1

1 Overview 3

1.1 Background 3

1.2 Categories of DDoS Attacks 6

1.2.1 Semantic attacks 7

1.2.2 Brute Force attacks 8

1.2.3 Attack Tools 10

1.3 Challenges for DDoS Defense 12

1.4 State-of-the-art of DDoS defense 14

1.4.1 Network-defense mechanisms 15

1.4.2 Application-defense mechanisms 21

1.5 Contribution of the thesis 23

1.5.1 Applications defend DDoS attacks using port hopping . 24

1.5.2 Lightweight filter balancing throughput and protection trade-off 25

1.5.3 Complementing the network-capability mechanism . . . 26

1.5.4 Tuning the granularity of traffic control 27
1.5.5 Scalable online DDoS detection 28
1.6 Chapters outline 29
1.7 Conclusions and future research questions 31
Bibliography 33

II PAPERS 41

2 PAPER I: Mitigating Distributed Denial of Service Attacks in Multiparty Applications in the Presence of Clock Drifts 45
2.1 Introduction 46
2.2 Problem and System Model Definitions 51
2.3 Protocol for Single Client Case 53
2.3.1 Overview 53
2.3.2 Contact-Initiation Part 54
2.3.3 Sending the Application Data 56
2.3.4 Adaptive Hopping Period: the HoPerAA Algorithm . . 57
2.4 Supporting Multiple Clients: the BigWheel Algorithm 62
2.5 Analysis 65
2.6 Experimental Study 73
2.7 Discussion 78
2.8 Conclusions 79
Bibliography 81

3 Paper II: Off The Wall: Lightweight Distributed Filtering to Mitigate Distributed Denial of Service Attacks 87
3.1 Introduction 88
3.2 Problem and System Model Definitions 92
3.3 SIEVE Overview 93
3.4 Design Details 95
3.4.1 Bootstrapping connection setup 95
3.4.2 Lightweight traffic filtering 98

3.4.3 Using SIEVE as a complementary filter 101
3.5 Analysis 102
3.5.1 Effectiveness of protecting connection setup 102
3.5.2 Filter effectiveness 106
3.5.3 Induced latency by routing via overlay nodes 109
3.6 Simulation Study 111
3.6.1 Connection setup simulation 111
3.6.2 Traffic filtering simulation 114
3.7 Related work 116
3.7.1 Overlay-based DDoS mitigation 117
3.7.2 Other solutions 118
3.8 Conclusion 119
Bibliography 119

4 PAPER III: Mitigating Denial of Capability Attacks Using Sink Tree Based Quota Allocation 125
4.1 Introduction 126
4.2 Sink Tree-Based Quota Allocation 129
4.2.1 Quota Allocation 130
4.2.2 Dealing with IP Spoofing and Early Acquiring within Domains 131
4.3 Analysis of the Algorithm 135
4.4 Overhead Evaluation 138
4.5 Experimental Study 139
4.6 Discussion and Future Work 141
Bibliography 142

5 PAPER IV: CluB: A Cluster Based Framework for Mitigating Distributed Denial of Service Attacks 147
5.1 Introduction 148
5.2 Related Work 150
5.3 System Model 152
5.4 Overview of *CluB* Framework 153

5.4.1 Coordination authorities . . . 154
5.4.2 Authentication tokens . . . 155
5.4.3 Egress/Ingress Checking Routers . . . 156
5.5 The Basic Protocol in *CluB* . . . 157
5.5.1 Permission Requesting . . . 158
5.5.2 Packet Encapsulation . . . 159
5.5.3 Packet Forwarding . . . 159
5.5.4 Filtering replayed packets . . . 162
5.5.5 Token-Refreshing . . . 164
5.6 Analysis and Evaluation . . . 164
5.6.1 Filtering efficiency . . . 164
5.6.2 Filtering efficiency comparison between *CluB* and the capability-based mechanism . . . 166
5.6.3 Effect of packets flooding to the checking routers . . . 167
5.7 Controlling the Granularity of Clusters . . . 171
5.8 Further Discussion . . . 176
5.8.1 Controlling the checking process . . . 176
5.8.2 Dealing with valid packets flooding . . . 177
5.9 Conclusion and Future Work . . . 178
Bibliography . . . 178

6 Paper V: STONE: A Stream-based DDoS Defense Framework 183
6.1 Introduction . . . 184
6.2 System Model . . . 187
6.2.1 Basic System Model . . . 187
6.2.2 Threat Model . . . 188
6.2.3 Anomaly-based Detection . . . 189
6.2.4 Stream Processing Engine (SPE) Model . . . 190
6.3 Overview of STONE Structure . . . 192
6.4 Building Profiles and Detection Methods . . . 193
6.4.1 Comparison system . . . 196
6.5 STONE- Detailed Description . . . 198

6.5.1 Data aggregator 198
6.5.2 Detection Control Center 200
6.5.3 Filter Component 211
6.6 Evaluation 212
6.6.1 Data Description and Evaluation Setup 213
6.6.2 Memory Consumption 214
6.6.3 Scalability evaluation 216
6.6.4 Mitigation effectiveness 218
6.6.5 Detection accuracy 220
6.7 Related Work 221
6.8 Conclusion 223
Bibliography 223
Example of the Exponential Histograms 227
Example of weighted features 228

List of Figures

1.1 The largest bandwidth attacks reported from 114 Internet service providers 5
1.2 Average number of DDoS attacks per month during October 2010 to September 2011, based on the reports from 114 Internet service providers throughout the world 6
1.3 Stages of a typical DDoS attack 10
1.4 A typical DDoS network 11
1.5 Example of CAPTCHA images 22

2.1 Worker ports' open interval with overlap 57
2.2 Client's clock is slower than the server, which means $\rho_c < 1$. 59
2.3 Client's clock is faster than the server, which means $\rho_c > 1$. 59
2.4 Messages exchange with timestamps and the associated time points. . 60
2.5 The open intervals of port p_0^i, p_1^i and p_2^i. 64
2.6 Arrival duration covers γ changing periods of guard ports. 64
2.7 The average number of contact-initiation trials in one contact initiation part 75
2.8 The length of HOPERAA execution interval grows with the number of HOPERAA executions 76
2.9 The receiving percentage of the server; μ is set to 100 milliseconds. . 77
2.10 The receiving percentage of the server; μ is set to 40 milliseconds. . . 78

3.1 Cookies and random scheduling are used to protect the connection setup phase 97

3.2 The average ratio between the shortest path length of a random node-pair through an overlay node and the original shortest path length . . 110
3.3 The proportion of legitimate requests that can be forwarded by an overlay node 110
3.4 The average waiting time for a response of a legitimate request to be sent, compared with the analytical result 110
3.5 The percentage of legitimate packets received by the server, when there are 32 overlay nodes in SIEVE 113
3.6 The percentage of legitimate packets received by the server, when there are 64 overlay nodes in SIEVE 113
3.7 Bandwidth used at the server's incoming link with different number of overlay nodes used in the protection, compared with no protection . 114
3.8 Fraction of the attack traffic that can pass through the lightweight filter 116

4.1 Quota allocation within the sink tree of s. 129
4.2 Time sequence diagram of random-delay response. 131
4.3 The average number of trials to establish a capability for a legitimate host 141

5.1 Example for describing *CluB*'s architecture and packet Forwarding . 157
5.2 AS Neighborhood Distribution 164
5.3 Filtering efficiency comparison between *CluB* and the capability-based mechanism 166
5.4 The probability that at least one of the ECRs is compromised 173
5.5 Average traffic stretch for routing packets via ECRs. 173
5.6 Average traffic stretch for routing packets via ECRs in the clusters of power-law topology. 173
5.7 Trade-off between security and path diversity 175

6.1 Structure of STONE 194
6.2 The source clusters are partitioned into 8 groups 197
6.3 Data Aggregator processing steps 199

6.4 Example of *Monitoring Period*, *Historic Period* and basic functionalities implemented by DCC modules 202
6.5 Example of interaction between sliding window model and monitoring periods 204
6.6 Illustration of filtering incoming packets when attacks are detected . . 213
6.7 Data Aggregator memory consumption 215
6.8 Data Aggregator scalability 217
6.9 Filter scalability 218
6.10 Traffic volume passing through the filter VS the total traffic load during the attack 219
6.11 Percentages of legitimate and attack traffic that is forwarded by the filter component 220
6.12 False negative ratio with different normalized thresholds and attack rates 221
6.13 Exponential Histograms example 229

Part I

INTRODUCTION

1 Overview

1.1 Background

According to the news on August 6, 2009, Twitter was shut down for hours, silencing millions of Tweeters [21]. From a user's point of view, the first indication of this outage was "site is down". Actually, not only was the site down, but client applications that depended on the Twitter API could also not connect to the service, creating a complete Twitter blackout. According to the Twitter's admission, this outage was caused by a *Denial of Service Attack*.

A Denial of Service (DoS) attack is an attempt by the adversary[1] to prevent the legitimate users of a service from using that service. Generally speaking, any attack that can saturate or exhaust system resources or get the system into

[1] The terms attacker and adversary are used interchangeably throughout this thesis.

fault status sometimes even crashes should be identified as a DoS attack. DoS problems are not new, they have been there for more than 20 years and keep evolving over time. The first well-known DoS is the Morris Worm [72] which is an Internet worm developed by a graduate student. This worm can exploit and infect vulnerable systems automatically and then replicate itself. The network can be easily congested with the massive messages created by this worm, so Morris Worm is definitely a DoS, even though it was developed without malicious intention. From then on, more and more DoS incidents were observed and reported. Following the rapid growth of the Internet, by taking its amazing ability of information searching, retrieving and exchanging, more and more social services rely on the network applications and communications. These services include scheduling travel itineraries, receiving/reporting information about severe weather or potential disasters, e-commerce, on-line medical diagnostics and scheduling emergency management events, etc. Any denial of such services can cause huge damage, not only loss of money but may also loss of human lives.

Nowadays, DoS attacks are usually launched in a distributed way: the attack traffic is from many attacking sources and the aggregated traffic volume is so big that it can easily deplete the victim's key computing resources, such as bandwidth and CPU time. When the adversary compromises multiple machines to launch a Denial-of-Service attack, this becomes a *Distributed Denial-of-Service* (DDoS) attack. The period between year 2000 to year 2004 was the period of the highest known frequency of DDoS attacks. In 2001, researchers from CAIDA observed 12000 attacks against more than 5000 distinct targets from a three week-long dataset [56], using backscatter analysis. The motives of DDoS attacks can be various. Some people may just show off their skills or prove that they found some system vulnerabilities, like the situation of Morris Worm. Economic incentive is another motive of DoS attacks. Companies can get benefits by launching attacks to their competitors or they can hire attackers to do that [11]. On the other hand, if the attackers have the ability to launch DDoS attacks, they can blackmail the victim, so the potential victim has to pay for avoiding that [85]. Political reason is also an important motive. Incidents re-

ported in [46] and [52] can be examples. In this sense, DoS, especially DDoS, not only threatens the Internet, but also threatens the civil security, due to its prevalent usage in cyber-crimes. Thus to understand well the characteristics of DDoS problems and investigate corresponding defense mechanisms have significant contributions not only for academia and industry, but also for the social security and emergency management agencies, since they can use such knowledge to enhance their abilities of risk assessments and help the stakeholders to make appropriate decisions when facing DDoS threats.

Since DDoS problems can make huge damage and have big impact on legitimate Internet usage and civil security, in the last decade, more and more researchers from academia, industry and also government organizations devoted themselves into this research area. Though many solutions were proposed to solve the DDoS problem and some types of the attacks are indeed countered, DDoS attacks continue to be a main threat in the Internet. According to the report from Arbor Networks [58] the scale of the DDoS attacks evolved a lot that it was observed a significant increase in the prevalence of attack rates in the 10 Bbps range. The frequency of DDoS attacks, though is not as high as year 2000 to year 2004, is still far from extinction. Figure 1.1 and Figure 1.2 show the scale and frequency respectively.

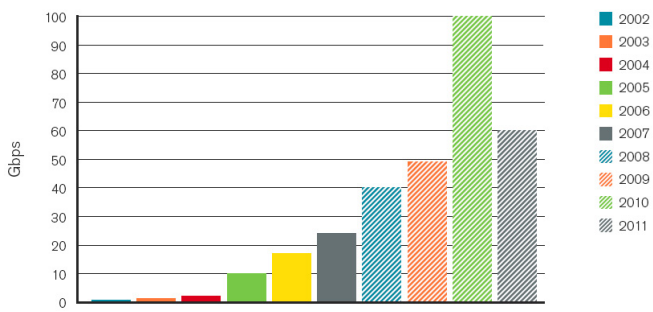

Figure 1.1: *The largest bandwidth attacks reported from 114 service providers. Source: Arbor Networks [58]*

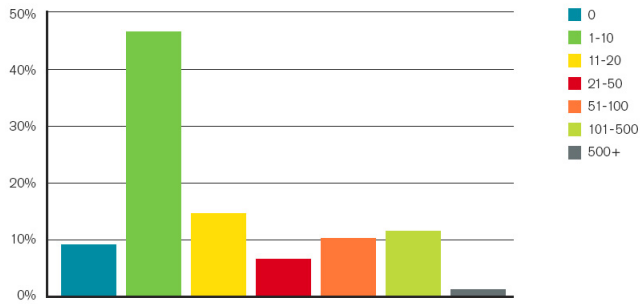

Figure 1.2: *Average number of DDoS attacks per month during October 2010 to September 2011, based on the reports from 114 service providers throughout the world. We can see that there are around* 45% *of the respondents encountered DDoS attacks 1-10 times per month. More than* 10% *of the respondents encountered DDoS attacks 101-500 times per month. Source: Arbor Networks [58]*

1.2 Categories of DDoS Attacks

Preventing or mitigating DDoS attacks is not an easy job. First we have to understand how the attacks work. To achieve the goal of DDoS, the adversary can adopt various means: It can study the flaws of communication protocols (or their implementations) and insert malformed or bogus packets to subvert the legitimate communications. This kind of attacks can be called *semantic attacks*. In semantic attacks, a single machine can complete the attack goal, since one malformed packet is enough to imped the service. Semantic attacks can be prevented by fixing the corresponding bugs in the protocols or applications. While usually the adversary does not need to inspect the implementation of protocols, it may just flood seemingly legitimate traffic to congest the victim's network or keep the victim busy processing the packets, so that the legitimate clients cannot get served. This kind of attacks can be called *brute force attacks*. To successfully flood packets to overwhelm the victim's network, usually the adversary needs to recruit many compromised machines or zombies to flood packets simultaneously. Brute force attacks are the most common forms of the

distributed denial of service attacks. The focus of this thesis is to defend against brute force attacks.

1.2.1 Semantic attacks

Some typical examples of semantic attacks are:

Teardrop

The adversary sends incorrect IP fragments to the target. The target machine may crash if it does not implement TCP/IP fragmentation re-assembly code properly [14]. This kind of attack can be prevented by fixing the IP implementation bugs in operating systems.

Ping of Death

A ping of death is an attack that the adversary sends the victim a ping packet which has more than 65535 bytes. Since many systems can not handle ping packets larger than 65535 bytes, handling packets of this size may cause a buffer overflow which may cause a system crash. More information can be found in [37].

BGP Poisoning

The BGP protocol is used to establish routing paths between networks in Autonomous system level. The routing information is updated by exchanging the BGP advertisement between routers. Usually, the routers update their routing tables without verification of the BGP advertisements. The adversary can subvert the network communication by announcing a better route to some destinations, then all the packets to the destinations are routed to the adversary. Also the adversary can disturb the BGP routing by announcing fake BGP advertisements with addresses of other routers. Then the corresponding traffic will be routed to those routers which do not have optimal routes to the destination.

These weak points in BGP are the basic motivation of designing secure BGP (S–BGP) and other revised versions of BGP [38].

1.2.2 Brute Force attacks

Brute force attacks aim at exhausting the victim's network bandwidth or computing resources by means of flooding massive malicious packets. To deplete the victim's computation resources, the adversary usually uses the packets of Internet protocols which have *request-reply* scheme, such as TCP, HTTP. During the attacks, massive spurious requests are flooded to keep the target busy serving them, thus impeding the legitimate usage. To deplete the bandwidth, the adversary can basically flood any types of packets to congest the target network link. Examples can be UDP flooding and ICMP flooding.

SYN Flood

In a SYN flood attack, the adversary takes the advantage of the three-way handshake for a TCP connection. In normal execution, when a TCP server receives a SYN packet, it opens a session for this new connection and sends back a SYN/ACK packet to the initiator. When it reaches a timeout and there is no ACK packet received from the corresponding initiator, the session will be closed and the corresponding resources for the session are released. During the attack, the adversary continues sending SYN packets without sending back the final ACK packets for the TCP handshakes, the server's resource (e.g. memory) can be quickly depleted by maintaining many half open sessions, thus legitimate connection requests cannot be served. We refer to [27] for more details of SYN flooding.

HTTP Flood

In HTTP flood attacks, the adversary floods massive spurious HTTP requests for downloading a web file from the target server. This file is usually a large file that the server may need to load from the hard disc and spend considerable CPU time to transfer it via packets. However, continuously requesting big files

can be suspicious. To avoid being detected, the adversary can instruct zombie machines to get a specific web page as the start and then follow the links on that page recursively, which can mimic the normal web browsing behaviors [63].

ICMP Flood (Smurf Attack)

In an ICMP flood attack, the adversary floods ICMP Echo packets to some network which broadcasts these messages to all the hosts in the network. These ICMP Echo packets have the victim's IP address. All the hosts who receive the echo packet will send Echo reply packets to the victim, which exhaust the victim's bandwidth. Actually, this kind of attack is a mixture of a semantic attack with brute force. The way the attack works is based on response mechanism in ICMP. However, from the perspective of the victim, it is brute force, as the type of the attack is just flooding packets from many machines. Similar to ICMP flooding attack, the adversary can take advantages of any reply-based protocol to launch *reflected attack*, by spoofing requests from the victim to a large set of Internet servers, resulting in a big volume of reply messages towards the victim network. Common protocols used in this kind of attacks include DNS queries [77], ICMP. For more information, one may refer to an analysis about reflected attacks [62].

UDP Flood

During a UDP flood attack, the victim's network is overwhelmed by a large volume of UDP packets [13]. The attack packets are usually with random port numbers. When the victim receives a packet, if there is no application listening at the corresponding port, then the victim may generate an ICMP packet of "destination unreachable" to the sender. Thus massive UDP packets to the victim's inactive ports may exhaust both incoming and outgoing capacities of the victim.

Note that, although we classify the DDoS attacks into different types, it does not mean that every time the adversary only launches one type of attack. Basically, the adversary can flood any type of packets in a DDoS attack. For example, the

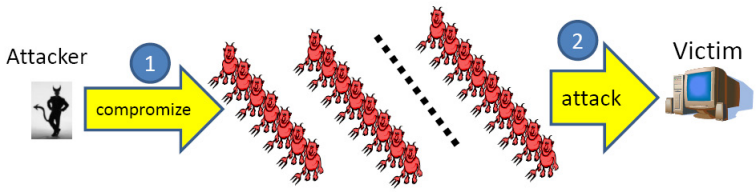

Figure 1.3: *Stages of a typical DDoS attack*

adversary can combine TCP SYN/ACK, ICMP Echo, ICMP Echo Reply, UDP packets together to flood the victim. In this case, the attack is really hard to be classified, so a filtering mechanism against a specific type of attacks may not be sufficient to solve the problem.

1.2.3 Attack Tools

Basically, there are two phases in a DDoS attack. First the adversary exploits the Internet to recruit machines which can be used as attacking agents (zombies or bots). The procedure of finding vulnerable machines and making them attacking agents are usually conducted by Trojans and worms, which can scan remote machines for security holes and infect them with attack code. Examples for such Trojans and worms can be found in [55]. After the recruiting phase, when the adversary wants to launch an attack, the attacking commands can be sent to the attacking agents. A typical procedure of a DDoS attack is shown in Figure 1.3. To cover its real identity, the adversary usually recruits zombie machines with the help of handler (master) machines. The adversary first compromises and infects one or more masters with attacking codes, according to which the masters can further compromise other zombies. A typical DDoS network is shown in Figure 1.4, which can also be referred to as a *botnet*. There are several tools used for maintaining the DDoS network and launching attacks, such as Trinoo [24], TFN [23], TFN2K [18], stacheldraht [22], Shaft [73] and mstream [74]. All of these tools are based on the architecture shown in Figure 1.4. The differences among them are basically the communication patterns

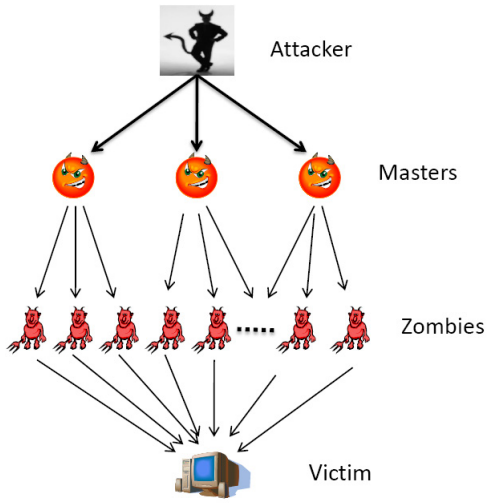

Figure 1.4: *A typical DDoS network*

between the adversary and the masters, and also between the masters and the zombies. The types of attack packets are also different from different tools. For a brief explanation about these attacking tools, one may refer to [10]. For more information about the architecture of the botnets and their communication patterns, we refer to [19, 33].

It is noticeable that for some situations the adversary can assemble the bots army with volunteers, if they have some common purpose. One example is an orchestrated attack on organizations such as Mastercard.com, PayPal, Visa.com and PostFinancesuch in 2010. This attack was launched by a group of people called ”anonymous” who wanted to express their indignation to the injustice against Wikileaks [1]. The attack is launched using an attack tool called LOIC [4], which can direct the volunteers to attack the indicated websites with massive HTTP requests. It is easy to join in such attacks. Volunteers only have to open an on-line web page created by the attack initiator and click the attack button on the page, then the web browser will repeatedly and rapidly send HTTP requests to the target [35].

1.3 Challenges for DDoS Defense

DDoS attacks are powerful, since the adversary can get very big aggregated bandwidth from many compromised machine, it is very easy to overwhelm the victim's network due the asymmetry of the bandwidth resources. Defending against DDoS is a difficult task, not only because of its big attack scale, but also because of the various strategies that the attacker can adopt. In particular, challenges for DDoS defense include:

The attacking sources are distributed In DDoS attacks, the attacking traffic consists of many malicious flows originating from hosts which are usually scattered among the Internet. If the malicious traffic is composed of a huge number of small malicious flows, it is difficult to distinguish the good and malicious traffic. Besides, the Internet infrastructure seldom provides services to the end hosts for controlling traffic sent to them. Instead, based on the *end-to-end paradigm* or *simple core and complex edge* [53], when packets are forwarded in the Internet, the intermediate networks (especially the core routers) will do the best-effort to deliver the packets to their destinations. So when DDoS attacks happen the victim usually does not have the ability to prevent the traffic of others from reaching its network. It is frequently necessary to have a distributed, possibly coordinated response system [55]. It is also desirable that the defense mechanism can be deployed in a distributed manner at many points of the Internet, thus filtering the malicious traffic as much as possible before it reaches the target. However, global deployment is always hard to achieve due to the distributed administration of the Internet.

IP spoofing is quite common in DDoS attacks IP spoofing refers to assigning an IP packet with a source address which is not the address of the sender. The attacker can use spoofed IP addresses in the malicious packets to cover the real identities of the attacking sources. IP spoofing may affect the accuracy of the counter measures of DDoS attacks. Since some of the attack detection mechanisms identify the abnormal traffic by source addresses, if the source

addresses are spoofed, traffic from innocent hosts may be blocked but the malicious traffic may still reach the victim if the attacker changes addresses in the malicious packets. There are many solutions against IP spoofing, however any solution that relies on message authentication is potentially vulnerable to DoS attacks [28]. Ingress-filtering [48, 59] or router based filtering [47] also cannot solve the problem completely, due to the global deployment requirement and the multipath routing scheme of the Internet.

The attack rate can be dynamic During a DDoS attack, all zombie machines may flood the victim simultaneously with all their sending capacities. The aggregated attack traffic can suddenly grow far beyond the target link capacity. However, big abrupt changes of the traffic can be easily detected, so the adversary may gradually increase the attack traffic rate to consume the link bandwidth slowly to avoid being discovered. The adversary can also attack the target link with periodic short *traffic bursts* (which refers to *pulsing attack* [55]) that keep protocols who have congestion-control mechanisms, such as TCP, sending packets with low rates [43].

The attacker may have various attack abilities It would be even worse, if the adversary can combine other malicious behaviors with DDoS attacks. For example, the adversary may eavesdrop some network links and it can analyze packets sent by legitimate hosts (which is referred as *sniffing*); the attacker may also pretend to be other network hosts (which is referred as *man-in-the-middle*) so that it may subvert the corresponding communications of other hosts; several malicious hosts may even collude to confuse or deceive legitimate hosts or network administrative entities. So the problem is even more complicated and difficult, if we want to deal with such powerful attackers.

The legitimate traffic should be affected as little as possible Mitigating is always a two-fold project: On one hand, the illegitimate traffic should be filtered as much as possible, while on the other hand, the network performance for the legitimate traffic should be degraded as little as possible. Actually, there

is a trade-off between the two aspects of the project: to effectively and accurately filter out malicious traffic, fine-grained traffic inspection or classification algorithms are needed. However, the time complexity of such inspection or classification procedures is not trivial. Continuously operating such algorithms will definitely harm the throughput of the legitimate traffic, since packets may be queued in the checking entities and need more time to be forwarded. Due to the big scale of the DDoS attacks (e.g. the attack rate can reach multi-10Gbps [58]), the network checking entities which execute packet inspections and filtering may run out of resources and become potential targets of the DDoS attacks.

1.4 State-of-the-art of DDoS defense

According to the activity level, the DDoS defense mechanisms can be classified as *proactive defense*, *reactive defense*, and *DDoS detection*. Proactive defense aims at preventing the malicious packets from reaching the victim, thus they cannot impact the protected service. Proactive defense methods are always turned on so that the malicious packets can be dropped as soon as possible when they are identified. Reactive defense aims at minimizing the loss caused by DDoS attacks. Reactive defense methods are always activated after the attacks are detected or the targeted service has already impacted. DDoS detection aims at detecting DDoS attacks when they are launched. It is important, since the detection results may impact when and which defense methods should be activated.

Based on the deployment layer, the defense mechanisms can be classified as *application defense* and *network defense*, depending on whether network entities, e.g. routers, other than the end hosts are involved for mitigating the attacks. In this section, to connect with our work, we give a brief overview of the exiting DDoS defense mechanism based on the classification of *application defense* and *network defense*. For more classifications and surveys of DDoS defense mechanisms, we refer to [50, 53, 55, 63].

1.4.1 Network-defense mechanisms

Network-defense mechanisms can be mainly categorized into *proactive solutions*, *reactive solutions* and *DDoS detections*:

Proactive network-defense

Proactive solutions aim at mitigating or preventing the DDoS attacks before they affect the performance of targets. Most of them focus on finding methods to identify the malicious packets and drop them as early as possible when they are transfered in the network. Some proactive solutions try to mitigate DDoS attacks by filtering out packets with spoofed IP addresses [29, 47, 59], due to the fact that many DDoS attacks involve IP spoofing. The basic idea is that routers keep information about the set of the IP addresses that each interface is expected to see. If the router receives a packet with the source address not belonging to the expected address-set of the corresponding interface, the packet is identified as malicious and it will be dropped. For example, a leaf network[2] has the network address $129.16.22.0/24$. When the access router receives a packet leaving this leaf network, the packet should have source address with the prefix $129.16.22.*$. The malicious packets whose addresses do not have prefix $129.16.22.*$ cannot leave the network and will be dropped by the access router. However, such filtering mechanism is not sufficient to solve DDoS problem. The attacker can still spoof the addresses within the expected range. Sometimes when the "attack army" consists of a huge number of zombie machines, the attacker does not even have to use IP spoofing, since each zombie machine can just flood a small volume of traffic with its own address, then the aggregated malicious traffic is enough to overwhelm the victim network.

To effectively distinguish malicious packets from legitimate packets, *Capability* based mechanisms [3, 49, 60, 82, 83] against DDoS were proposed as an alternative. The essential component in this kind of solutions is a hash-based message authentication token (capability) which is binded with the addresses of

[2] A leaf network does not offer service of forwarding packets from other networks. In the graphical network topology, a leaf network appears as a leaf node in the graph.

the packet sender and receiver. When a sender wants to send data to a receiver, it should first send a request to the receiver to get the permission for sending. After checking the sender's identification, the receiver will grant the capability to the sender, and this capability will be added in the following packets from the sender to the receiver. Capabilities can be generated by the receiver or by the routers along the path. In order to avoid the possibility that the attacker may break the capability by intensive searching, capabilities are changed periodically. Packets from a specific sender have to be always forwarded along the same path, since only the routers along that path can check the validities the packets which implies that every capability is *path-dependent*. If the path is broken, the capability should be regenerated. Also, intermediate routers do not take charge of issuing capabilities, they just forward all the packets with valid capabilities. This implies that malicious hosts can allow each other to send traffic and flood a part of the network. Another important issue is that attackers can flood capability-request packets to the request channel to prevent the legitimate requests being delivered. The attack against the initial capability request was referred to as *Denial of Capability* (DoC) [5]; i.e. capabilities need to be used together with mechanisms against DoC [30, 61].

Overlay networks are also proposed to act as the filters against DDoS attacks. The overlay nodes can be either routers or regular machines. Since overlay networks are easy to deploy and do not require global changes of the current routing protocols, many proactive defense mechanisms are built on overlay networks [2, 25, 39, 65, 70, 71]. The basic idea is to use an overlay network as a proxy for the protected victim. Any packet that goes to the victim should be checked and forwarded by the overlay nodes, thus expanding the receiving and filtering capacity of the protected victim. However, most of the overlay-based solutions still face the problems as the capability-based solutions: First, unforgeable authentication tokens (e.g. MACs) are used for packets filtering, which may undermine the throughput of the filtering entities (i.e. overlay nodes). Second, the attacker can flood the request channel to prevent the legitimate hosts from getting the authentication tokens. Early overlay-based solutions, such as SOS [39] and Mayday [2], do not consider the issue of se-

curing request channel, since they focus on protecting the traffic of *confirmed users*, which by assumption have prior permissions to communicate with the server. MOVE [71] uses SSL and Multipath-Overlay [70] uses UMAC [9] to authenticate packets. For protecting the connection setup phase, MOVE and Multipath-Overlay push the task of distinguishing spurious requests to overlay nodes who may accept all the connection requests, due to the lack of the local knowledge of the server. These two solutions also suggest to use Graphic Turing Test (GTT) for filtering out requests from remotely controlled zombie machines. However, GTT needs humans involved and is not transparent to users. OverDose [65] protects the request channel by using crypto-puzzles. Each request packet should contain a correct solution of a puzzle generated with the current puzzle seed (which is changed periodically), and packets with correct solutions of puzzles in higher levels have priority for being forwarded. OverDose proposes to use *flow cookies* [12] to authenticate and regulate flows. However, it requires the overlay nodes to keep states for all the clients who have legitimate flow cookies. Phalanx [25] tries to provide a system that is easy to deploy, yet is powerful enough to mitigate DDoS attacks. Phalanx uses puzzle-based solution as OverDose to protect request channel. Instead of using overlay nodes to filter malicious packets, Phalanx uses overlay nodes as temporary *mailboxes* of the server. Legitimate clients send different packets to different overlay nodes according to a pseudo-random pattern. In order to get legitimate packets, the server has to know which client sends which packet to which overlay node, and has to send requests to the correct overlay nodes as soon as possible to get the packets before the buffer of the overlay node becomes overloaded. With big number of clients and legitimate packets, the overhead can be too high for the server to perform those operations.

Reactive network-defense

Generally speaking, reactive solutions aim at mitigating DDoS attacks after the attacks have been detected. This detection is usually a task of the targets, since the targets are the real witnesses of the attacks. The malicious traffic can be identified by the paths through which it is forwarded using various *IP traceback*

methods [20, 64, 66, 67]. Roughly speaking, in the IP traceback methods, the routers in the network put marks in the packets traversing them. When DDoS attacks happen, the victim can use these marks to assemble the paths through which the malicious traffic has been forwarded, then the victims can require the routers along those paths to filter the corresponding malicious traffic. These filtering processes can also be done recursively from the victims' gateway routers to the attacking sources' gateway routers [6, 51].

There are some other reactive solutions whose objective is to achieve some kinds of fairness (e.g. *max-min* fairness) when DDoS attacks happen [17, 75, 84]. These mechanisms will adjust the rate limits for different links according to their flow volumes in the normal situation and their recent flow fluctuations. The ideal goal is that when an attack happens, the malicious flow must be controlled within its part of the network, and this part of the network should be identified and isolated as much as possible; the network used mostly by legitimate users should not be affected.

The reactive defense mentioned above can effectively mitigate DDoS attacks when the attack paths can be successfully identified. But when the attacking sources are quite scattered among the Internet, it is not easy to identify the path through which the attack traffic is forwarded, especially when the attacker compromises millions of zombie machines and each of them sends a small volume of malicious traffic which is similar to the legitimate traffic.

Reactive network-defense methods also include solutions using congestion puzzles. Wang et al. [79] proposed countermeasures based on *congestion puzzles* (CP) to mitigate bandwidth-exhaustion attacks. CP mechanism attempts to force the attacker to invest vast computation resources in order to effectively perform DoS(DDoS) attacks. Once a link is congested, the adjacent router requires the traffic flow to be accompanied by a corresponding *computation flow*– a continuous flow of puzzle solutions. Since solving a puzzle requires a brute-force search in the solution space, the computation flow imposes a computational burden on the clients who transmit traffic via the router. The authors also extend CP mechanism to a *distributed puzzle mechanism* (DPM). In DPM, a router can ask its upstream routers to help with controlling the traffic

flows before they converge to the congested link. The same authors also extend the method to a multi-layer framework that uses puzzles to solve both connection requests flood (e.g HTTP flood and SYN flood) and bandwidth-based flood [80].

There are two issues of puzzle-based solutions: First, throughput of legitimate traffic can be affected due to the computation burden to solve the puzzles. Second, the proposed mechanism can cause unfairness in puzzle-solving time over different hardware platforms.

DDoS detection

Systems used to detect DDoS attacks are considered as a type of *Network Intrusion Detection Systems* (NIDs). The latter employ two distinct approaches to detect malicious activities: *signature-based detection* and *anomaly-based detection* [42]. Signature-based detection is used to detect known security threats (e.g., a virus or malformed packets) looking for specific patterns (signatures) to appear in individual packets. On the other hand, anomaly-based NID is used to detect potential security threats based on abnormal behaviors over a set of packets. Due to the nature of flooding-based DDoS attacks, where every malicious packet may seem legitimate if analyzed individually but where the overall traffic behavior may suffer abrupt variations (e.g. abrupt increases of traffic volume), anomaly-based detection is always used to detect flooding-based DDoS attacks.

In the recent decade, many anomaly-based detection methods were proposed to identify DDoS attacks from network traffic. Basically, these detection methods can be classified into two categories: *off-line DDoS mining* and *on-line* DDoS detection. Off-line DDoS mining usually try to find attacks by analyzing the main characteristics of feature distributions of the network traffic with some systematic methods, such as PCA (Principal Component Analysis)) [44, 45] and *dominate states analysis* [81]. The basic idea of PCA is to embed the multi-dimensional data into lower dimensional subspace in which normal instances and the anomalies appear significantly different. The basic idea of the dominate states analysis is to explore the interaction or dependence among the dimen-

sions of the data by identifying subset of values (dominate states) to represent or approximate the original data in their probability distribution. Anomalies can be identified since their dominate states deviate significantly from the normal ones. When the network anomalies are identified, data clustering methods, such as k-means clustering [34], are applied to group different types of anomalies together for further identification, correlating anomalies to attacks. To achieve accurate analysis results, the processing procedures of off-line methods are executed on the whole data trace, and these methods usually involve expensive computations, e.g PCA involves matrix computations for computing principal components of the data. So the off-line DDoS mining methods can hardly be used in online detection, due to time and space complexities. However, the analysis results from the off-line anomaly mining can help build the base-line profile for the real time detection.

Considering the big scale of the DDoS attacks, detection over big volume of traffic (e.g. multi-10Gbps) is really challenging. Computation over massive data streams is being studied in the emerging field of *data streaming*, aiming at methods for processing massive amounts of data in a real-time fashion, such that each tuple in the data stream is only processed once. Data streaming computation has been adopted in applications such as financial markets and mobile phones or credit card fraud detection applications [16]. Recently, data streaming has also been proposed for DDoS detection at high-speed network links, where streams of packets are processed by *continuous queries* to find anomalous DDoS-related traffic patterns in real time. Data streaming queries are referred to as continuous as they are constantly "standing" over the streaming tuples and continuously producing output results.

Most data-streaming based DDoS detection methods focus on using space-efficient and time-efficient algorithm to keep track of the heavy hitters, e.g. a source sending lots of packets to many destinations, in the monitored traffic. One particular algorithm used is *sketch* algorithm [31, 32, 40, 41, 86]. Sketch is a probabilistic summary technique which can sustain large streaming datasets. It keeps the summary updates using projection along random vectors to achieve space efficiency with guaranteed probabilistic reconstruction accuracy. How-

ever, sketch based solutions do not support continuous monitoring with sliding window, since the random vectors used for maintaining the sketches are reset when some anomalies are detected or some predefined period expires. Thus sketch-based solution may miss the anomalies spanning consecutive periods.

Considering the types of DDoS attacks that are in the focus of the previous work in DDoS detection, SYN flooding is the most common one, since such attacks usually cause imbalance between the number of SYN packets and the SYN/ACK or FIN packets [31, 32, 40]. However, monitoring such imbalance to detect SYN flooding may require the monitor to be deployed at the edge routers, due to the routing asymmetry. So solutions which can detect DDoS attacks at the early stage, i.e. at the backbone links, are desired. However, monitoring high speed traffic in backbone links is challenging [68].

To detect bandwidth flooding attacks, *change-point* detection [76] and *wavelet analysis* [8] were proposed. Change-point detection maintains a *moving average* of each flow and compares the current flow rate against the moving average; if the changing ratio exceeds the threshold, then the flow is identified as suspicious. Wavelet detection maps the series of the flow rates into a spectral domain. Since the attack flows and the legitimate flows have distinguishable frequency components, the presence of attack flows can be detected. However, most of the change-point based and wavelet-based detections only focus on detecting the abrupt changes of the traffic rate, so they may be insufficient for detecting connection requests flooding, like SYN flooding, since the traffic rate may not increase so much in such attacks.

1.4.2 Application-defense mechanisms

Most application-defense mechanisms focus on dealing with SYN flood and HTTP flood attacks. Typical reactive methods include *SYN cookies* [26] and *client puzzles* [78]. SYN cookies prevent the victim system from keeping half-open states for incomplete TCP connections. The server only allocates resources when it receives an ACK packet with a valid cookie. Client puzzles can slow down the attacker's rate of generating valid connection requests, since it

has to solve the puzzles. Typical proactive methods include *active tests* such as CAPTCHAs [57] which can tell whether the suspicious connection requests are generated by human beings or automatically by the zombie machines by showing the letters in a CAPTCHA's image. Figure 1.5 gives an example for such images. These solutions are designed for specific types of DDoS attacks, they cannot defend general packets-flooding attacks to the communication channels of the network-based applications (puzzle-based solutions may help, but as mentioned before, they affect the throughput of the legitimate traffic).

Figure 1.5: *Example of CAPTCHA images*

When considering general solutions for network-based applications to defend DDoS attacks, Badishi et al. [7] proposed a *port-hopping* scheme that enables the applications to defend DDoS attacks by changing their communication ports. This solution is motivated by the fact that network-based applications commonly provide some *open port(s)* for communication, making themselves become targets for DoS attacks. Adversaries who have the ability of eavesdropping messages exchanged by the applications can identify open ports and launch *directed* attacks to those ports.

Badishi et al. presented a acknowledgment-based port-hopping protocol in [7]. The protocol in that paper is focused on the communication between two parties, modeled as sender and receiver. The receiver sends back an acknowledgment for every message received from the sender, and the sender uses these acknowledgments as the signals to change the destination port numbers of its messages. Since this protocol is acknowledgment-based, time synchronization is not necessary. But note that the acknowledgments can be lost in the network, and this may keep the two parties using a certain port for a longer time. If the attacker gets the port number during this time, then a *directed attack* will be launched under which the legitimate communication can hardly

survive. To cope with that, a solution that reinitializes the protocol is presented in the same paper. The latter solution depends on clocks having the same rate; it allows for bounded drift in the clock phases (resulting in bounded differences of clock values) but not their rates (which would imply arbitrary differences of clock values). In [7] the authors also present a rigorous model and analysis of the problem of possible DoS to applications (ports) by an adaptive adversary, i.e. one that can eavesdrop, as in our case, too. The analysis, besides the parts that involve the port-hopping protocols proposed in that paper, also includes a part analyzing the effect of adversary's different strategies for launching blind attacks.

There is a client-transparent approach proposed in [69] which is quite similar to port hopping. This approach uses JavaScript to embed authentication code into the TCP/IP layer of the networking stack, so the messages with invalid authentication code would be filtered by the server's firewall. In oder to defend the DoS attacks, the authentication code changes periodically. There is a challenge server in charge of issuing keys, controlling the number of clients connected with the server and synchronizing the clients with the server as well. Since this approach relies on the challenge server, the protection of the challenge server is quite important. The paper mentions that a cryptographic based mechanism can be used to protect the challenge server, but this is not discussed in detail.

1.5 Contribution of the thesis

Distributed Denial of Service attacks are really hard to defend, due to their multifaceted natures: dynamic attack rates, various kinds of targets, big scale of botnets, etc. A complete and comprehensive defense solution (if there is any) for DDoS problem might even not be achievable. Instead, the multifaceted attack natures justify the need for multifaceted defense. In this thesis the research is focused on addressing the DDoS problem from different aspects which are motivated by the following questions:

1. *Can network applications be provided with stronger anti-DDoS methods, less dependent on the network and host characteristics such as timing*

uncertainties?

2. *Can we provide a method that is easy to deploy and has low overhead for filtering packets, yet is powerful enough to mitigate DDoS bandwidth-based attacks? Can we provide simple solutions to mitigate the request channel flooding problem which is encountered by many proactive defense mechanisms?*

3. *Can the network core provide protection of individual domains or end-hosts, thus managing related trade-offs? How to distribute the responsibility and knowledge of controlling the network traffic in order to address the DDoS threats to both the end hosts and the core of the network?*

4. *Can we provide efficient on-line detection methods that can detect different types of DDoS attacks at high speed links?*

Below we explain the thesis contributions towards answering the questions above. Basically, we address the DDoS problem from two aspects: DDoS mitigation and DDoS detection. For DDoS mitigation, we focus on both application-defense and network-defense. For DDoS detection, we focus on on-line detection of different types of DDoS attacks over big volume of data streams.

1.5.1 Applications defend DDoS attacks using port hopping

A critical issue of the port-hopping scheme is synchronizing the communication parties. Two main kinds of synchronization mechanisms are presented in the previous work. One is acknowledgment-based and the other one depends on synchronized clocks (cf. section 1.4.2). As acknowledgment loss can cause a situation where a port may remain open for a time interval long enough for an eavesdropping adversary to identify it and launches a *directed attack* to it. Having synchronized clocks may imply need for synchronization server, which could be the weak point in the system. An interesting middle ground exists for investigation and for employing the method on common networking systems. We investigate such questions opened in the earlier work and improve the port hopping scheme, so that it can be used (i) in the presence of *timing uncertainty*,

i.e. *clock-rate drifts*, implying that clock values can vary arbitrarily much with time; and (ii) in *multi-party communication*.

In particular, in order to deal with hopping in the presence of clock-rate drifts, we propose an adaptive algorithm, executed by each client when its hopping period length and alignment drift apart from the server's. To enable multi-party communication with port-hopping, we present an algorithm for a server to support hopping with many clients, without the server needing to keep state for each client individually. The basic idea in both algorithms is that each client interacts independently with the server and considers the server's clock as the point of reference; moreover, the server does not need to maintain a state for each client, since the main responsibility for the coordination is assigned to the client(s). As in e.g. TCP, a client of one session can be a server for another (possibly concurrent) session, hence the solution fits for symmetric use. According to the properties of our algorithm, there is no need for group synchronization which would raise scalability issues.

1.5.2 Lightweight filter balancing throughput and protection trade-off

By studying the communication architecture of the modern botnets [33], it is observed that the adversary usually needs some time to configure the bots with new attack commands. We reflected on this and on earlier research, i.e. the "hopping" paradigm where it is shown that smart updating of secret knowledge enables to both allow legitimate entities to use resources and leave non-legitimate ones "in the dark". Inspired from this reflection, we extended our "hopping" paradigm into network level aiming at mitigating bandwidth flood attacks.

Unlike other proactive network defense solutions that use message authentication codes (MACs) to distinguish malicious packets from legitimate packets, we use lightweight authenticators, i.e. IP addresses. According to its address, a legitimate host sends its packets to different filtering points in different time

periods. Since it takes time for the adversary to find out the filtering rules and to configure an attack using zombie machines, e.g. a botnet, if the filtering rules can change with appropriate timing, the adversary can be kept in the "dark" and cannot launch flooding attacks effectively. Using IP addresses as the authenticators needs neither extra fields in the packet header nor extra processing time for each incoming packet. This implies saving of many orders of magnitude of CPU cycles compared with MAC authentication.

SIEVE, our proposed lightweight filter uses an overlay network for building distributed checking points, since it does not require global changes of the current routing protocols. To send packets to the correct overlay nodes, each legitimate hosts still have to request this information from the protected server. We provide a simple solution to solve the problem of requests flooding attacks. Roughly speaking, in our solution, network hosts are partitioned into domains. Each domain has its quota per time period for sending connection requests to the server. Even if the adversary can compromise many sending sources in some domains, the connection requests from the uninfected domains will not be affected. In each infected domain, the solution provides guaranteed probability for legitimate clients whose requests will be forwarded by the overlay node.

1.5.3 Complementing the network-capability mechanism

As mentioned in section 1.4, we found that the network-capability mechanism is a novel approach to prevent DDoS proactively. However, this approach may suffer from the Denial-of-Capability (DoC) problem. This problem can be viewed as a problem of denial-of-connection. The DoC problem is that the adversary may flood connection requests to a specific server, so that the requests from legitimate users can not be received by the server. We give a sink-tree based quota allocation method to mitigate this kind of problems. Our basic solution is to distribute the quota of the *tickets* for the connections of the server. An analogy could be the tickets for a concert: You can only buy the tickets in your own area, e.g. city or country, so the adversary can only exhaust the quota in its area, while other areas will not be affected. We also address several important

issues, such as controlling the number of tickets that one host can get, how to distribute the tickets and how to maintain the distribution.

1.5.4 Tuning the granularity of traffic control

Considering malicious traffic, we would like to ideally disallow it completely from consuming network resources as what the network-capability mechanisms do. Ideally in an imaginary world, if we have a global network monitor that is extremely powerful and can observe and control every flow between any pair of hosts, DDoS attacks could be mitigated by having this monitor identify every potential DDoS attack and stop the corresponding traffic as soon as possible before it reaches the target. However, it is well-understood that such a centralized counter-measure is practically impossible due to huge implied overhead —it would actually resemble a DoS attack itself! On the other hand, a completely distributed low-overhead solution, where every entity has only local information, can have limited effect in stopping malicious traffic at some distance from the target. Seeing the problem from this perspective, we observe a trade-off in the achievable protection level of the network and the efficiency/overhead of the protecting method, depending on the granularity of control.

Roughly speaking, to balance this trade-off, we would wish to have some distribution of control and responsibility. This observation led us to think of an analogy in real-life; namely the *exit and entry control* problem between countries in the world. A citizen of one country needs a passport and the corresponding visa to go to another country. The passport stands for a permission that the local country allows this person to go abroad and recognizes this person as a good citizen. The visa is the permission from the destination country to allow this person to enter in. If this person will transfer a flight in a third country, a transit permission for passing by that country may be needed.

Inspired by the above idea, we propose *CluB*: a Cluster-Based architecture that prevent DDoS attacks proactively. In *CluB*, the network consists of a set of clusters —in the Internet, these can be e.g. Autonomous Systems (AS), or neighborhoods of ASes. Packets need permissions to exit, enter, or pass-by

different clusters.

We addressed the following issues in our architecture: We present and analyze the right level of distribution so that this can be feasible and scalable, how the permissions are issued, how the permission-control is carried out, how the permissions will be implemented so that they are hard to be faked by adversaries, what is the level of protection that can be achieved. We observe and study trade-offs in the achievable protection level of the network and the efficiency/overhead of the protecting method, depending on the granularity of control.

1.5.5 Scalable online DDoS detection

As discussed in section 1.4, reactive network defense always relies on the victim to identify malicious traffic by deploying detection mechanisms at the edge of the network. So it can not detect the attacks before the malicious traffic reaches the victim network. It also may fail to detect the attacks targeted at the core network. On the other hand, proactive network defense may degrade the network performance due to its continuous operations of checking packets. Therefore, it is desirable to have a powerful detection mechanism which can provide online DDoS detection at the *vantage points*, e.g. backbone links, thus offering a protection at an early stage in the core network and activating appropriate DDoS mitigations only when necessary.

Motivated by the above consideration, we focused on online DDoS detection over big volume of network traffic in a data streaming manner where each monitored packet is only processed once. In particular, we identified a small set of traffic features that can capture the anomalies reflecting most of important types of DDoS attacks. To keep the baseline profile (used for anomaly-based DDoS detection) updating with the evolution of the normal traffic, we provided solutions for both online DDoS detection and the baseline profile maintenance in a data streaming fashion. To meet the time and space constraints, we proposed scalable prefix-based aggregation and coordinate-based measurement for monitoring traffic and detecting DDoS-anomalies from high-

speed network links in an efficient data streaming fashion. Furthermore, we also defined a filtering mechanism that, upon detection of a DDoS attack, maximizes the forwarding of legitimate traffic while discarding as much illegitimate traffic as possible, by prioritizing traffic from the more familiar sources based on the monitoring and profiling information.

1.6 Chapters outline

Chapter 2, Paper I: Mitigating Distributed Denial of Service Attacks in Multiparty Applications in the Presence of Clock Drifts

By identifying the synchronization problem of the port-hopping method, we study the case where the communication parties have clock with rate drifts, which is common in networking. We propose and present an algorithm, BIGWHEEL, for servers to communicate in a scalable way with multiple clients in a port-hopping manner, thus enabling support to multi-party applications as well. The algorithm does not rely on the server having a fixed port open in the beginning, neither does it require from the client to get a "first-contact" port from a third party. We also present an adaptive algorithm, HOPERAA, for hopping in the presence of clock-drift. The algorithm is simple, based on each client interacting with the server independently of the other clients, without the need of the server keeping a state for each client separately, or issuing acknowledgments or using any time server. We show the properties of our solutions both analytically and experimentally.

Chapter 3, Paper II: Off-the-Wall: Lightweight Distributed Filtering to Mitigate Distributed Denial of Service Attacks

To balance the security/throughput trade-off of filtering malicious packets, we propose and present SIEVE, a lightweight distributed filtering method. Depending on the adversary's ability, SIEVE can provide a standalone filter for

moderate adversary models and a complementary filter which can enhance the performance of strong and more complex methods for stronger adversary models. SIEVE uses an overlay network to form a distributed "sieve" to filter malicious traffic aimed at servers. Overlay nodes use *lightweight authenticators* (e.g. source IP addresses) to filter packets. SIEVE includes also a simple solution to protect connection setup procedures between legitimate clients and protected servers, which provides guaranteed probability for the legitimate packets to receive service. We present analytical and simulation-based studies of the filtering efficiency and overhead of SIEVE and give a cost guideline on configuring the distributed filter based on the customized demand, thus balancing trade-offs.

Chapter 4, Paper III: Mitigating Distributed Denial of Capability Attacks Using Sink Tree Based Quota Allocation

As mentioned in section 1.4, most proactive defense mechanisms encounter the problem of request-channel flooding. Here we mitigate such problem by offering a solution to a typical instance of such attacks. In particular, we propose an algorithm to mitigate Denial of Capability (DoC) attacks. The algorithm divides the server's capacity for handling capability requests into quotas. Quotas are allocated based on a sink tree architecture. Randomization and Bloom filtering are used as tools against identified attacking scenarios. We both theoretically and experimentally demonstrate the properties of our algorithm, showing that legitimate hosts get service with guaranteed probability.

Chapter 5, Paper IV: CluB: A Cluster Based Framework for Mitigating Distributed Denial of Service Attacks

In this paper we study the problem of distributing the traffic control abilities among network entities, in order to achieve different granularities of DDoS mitigation. In particular, we propose a proactive cluster-based method to mitigate DDoS attacks, which we call *CluB*. This method treats the DDoS problem in a distributed manner, possibly at the granularity of Autonomous System level,

which are natural administrative domains. We addressed the malicious traffic control problem in two levels: intra-cluster level and inter-cluster level. *CluB* can collaborate with different routing policies in the network, including contemporary datagram options. Our method balances the effectiveness-overhead trade-off by addressing the issue of granularity of control in the network which we studied in the paper. We present the results of our studies and outline new questions for future research.

Chapter 6, Paper V: STONE: A Stream-based DDoS Defense Framework

In this paper we study the problem of monitoring high-speed traffic and detecting DDoS-related anomalies in a data streaming fashion. In particular, we present the design and implementation of a scalable online DDoS defense architecture, which can detect and mitigate DDoS attacks at high-speed network links, e.g. backbone links, thus offering a protection at an early stage in the core network. By investigating the traffic patterns of different forms of DDoS attacks (including connection requests flooding and bandwidth flooding), we propose an IP-prefix based aggregation method to monitor and detect DDoS-related anomalies. Furthermore, distinguishing from the prior DDoS detection solutions, we investigate the design space of combining parallel-distributed data streaming with both online detection and baseline profile maintenance, and give detailed solutions for achieving this. The proposed data streaming based DDoS defense solutions are implemented upon a parallel-distributed data stream engine. According to the evaluation with traffic based on real network datasets, the proposed solution achieves good scalability to the growing incoming traffic load, and the lightweight built-in filter can effectively block malicious packets from reaching the target when attacks are detected.

1.7 Conclusions and future research questions

Defending distributed denial of service attacks is challenging, due to their multifaceted natures: dynamic attack rates, various kinds of targets, big scale of

botnets, etc. Providing a complete and comprehensive defense solution (if there is any) for the DDoS problem could be very difficult. The project involves the investigation into subjects of multiple research fields, including network security, distributed computing, algorithm/protocol design. Instead, based on the study in this thesis, multifaceted attack natures justify the need for multifaceted defense. In particular, at a coarse-grained level, *CluB* framework can help autonomous systems to cooperatively regulate their traffic among themselves, thus flooding attacks targeted at core network can be mitigated. SIEVE can be used to protect individual servers from bandwidth flooding, achieving good balance between throughput and protection. If the target of the DDoS attack is a specific application channel, then network defense may seem expensive and impractical. Port hopping mechanism can help to mitigate the problem. Meanwhile, an effective DDoS detection mechanism is needed to detect attacks as early as possible and activate appropriate mitigation methods when necessary. Note that solutions proposed in this thesis are complementary, it is interesting and useful to continue the research on the methods for combining different approaches towards integrated solutions.

Defense against DDoS aatacks is always a two-fold project: On one hand, the illegitimate traffic should be filtered as much as possible, while on the other hand, the network performance for the legitimate traffic should be degraded as little as possible. In distributed control of unwanted traffic building on the work in this thesis, there are interesting trade-offs to study between efficiency/overhead and security level. Intuitively, increasing the security level for a certain protocol implies using stronger control mechanisms which may lead to degradation of some performance, such as throughput. Since different applications may have different performance requirements [54], an application-specific metric is needed for balancing the trade-offs between efficiency and security level. This future direction may shed light on the open problem about designing an anti-DDoS mechanism that is adaptive to different network-based applications that have different quality of service (QoS) requirements.

To make a DDoS defense mechanism work, one important issue is the deployment. Due to the need for adding new functions to the routers and the

modifications of the current routing schemes, many anti-DDoS solutions are not adopted in the practical usage. So far overlay-based solutions seem to have the minimum changes to the Internet, but still it does not mean that all the ISPs want to adopt such methods to protect their customers. It is interesting and important to offer a generic framework which can show the dependency between the filtering effectiveness and the deployment percentage, i.e. the proportion of the ISPs that deploy the filter mechanism. It is also important to investigate into the metrics, such as the user utility and deployment cost measurements presented in [15, 36], that measure the benefits and costs for adopting new network architectures (including anti-DDoS mechanisms), offering incentives for the corresponding deployment. The findings of this thesis not only contribute to the academic and industrial domains with new networking technologies and anti-DDoS solutions, but also to the social security area. Since more and more social services, including Internet banking, e-government and on-line medical diagnostics, rely on the network applications and communications. It is critical to deal with the problems which may prevent these services from functioning. The lessons learned about the multifaceted natures of the DDoS problem and multifaceted defense mechanisms offer fruitful knowledge for improving the current network-based emergency management systems. This thesis offers multiple strategies countering the DDoS problem. Based on the characteristics of the (potential) threats, the crisis management system administrators are able to choose appropriate strategies for the stakeholders to control the loss or prevent from being attacked. In the social security area, it is also worthy for the society actors to investigate legislative responses to the DDoS problem, due to their tight bind to the cyber-crimes and cyber-wars.

Bibliography

[1] E. Addley and J. Halliday. Operation payback cripples mastercard site in revenge for wikileaks ban, Dec. 2010 http://www.guardian.co.uk/media/2010/dec/08/operation-payback-mastercard-website-wikileaks.

[2] D. G. Andersen. Mayday: distributed filtering for internet services. In *USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*, pages 3–3, Berkeley, CA, USA, 2003. USENIX Association.

[3] T. Anderson, T. Roscoe, and D. Wetherall. Preventing internet denial-of-service with capabilities. *SIGCOMM Comput. Commun. Rev.*, 34(1):39–44, 2004.

[4] Anonymous Wikileaks supporters explain web attacks. http://www.bbc.co.uk/news/technology-11971259, Dec. 2010.

[5] K. Argyraki and D. Cheriton. Network capability: The good, the bad and the ugly. In *In Proceedings of Workshop on Hot Topics in Networks (HotNets-IV)*, November 2005.

[6] K. Argyraki and D. R. Cheriton. Active internet traffic filtering: real-time response to denial-of-service attacks. In *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 10–10, Berkeley, CA, USA, 2005. USENIX Association.

[7] G. Badishi, A. Herzberg, and I. Keidar. Keeping denial-of-service attackers in the dark. *IEEE Trans. Dependable Secur. Comput.*, 4(3):191–204, 2007.

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

[9] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. Umac: Fast and secure message authentication. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 216–233, 1999.

[10] S. Bosworth and M. Kabay. *Computer security handbook, Chapter 11*. John Wiley & Sons, 2002.

[11] Botnet Economy. http://dos-attacks.com/2010/10/26/botnet-economy/.

[12] M. Casado, P. Cao, A. Akella, and N. Provos. Flow-cookies: Using bandwidth amplification to defend against ddos flooding attacks. In *IWQoS 2006. 14th IEEE International Workshop on Quality of Service*, pages 286 –287, june 2006.

[13] CERT. Advisory CA-1996-01 UDP Port Denial-of-Service Attack, SEP. 1997 http://www.cert.org/advisories/CA-1996-01.html.

[14] CERT Advisory CA-1997-28 IP Denial-of-Service Attacks. http://www.cert.org/advisories/CA-1997-28.html, 2010.

[15] H. Chan, D. Dash, A. Perrig, and H. Zhang. Modeling adoptability of secure bgp protocol. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, pages 279–290. ACM, 2006.

[16] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[17] J. Chou, B. Lin, S. Sen, and O. Spatscheck. Proactive surge protection: A defense mechanism for bandwidth-based attacks. In *Proceedings of the USENIX Security Symposium*, pages 123–138, July 28-Aug 1 2008.

[18] Computer Emergency Response Team (CERT). Cert advisory ca-1999-17 denial-of-service tools, Mar. 2000 http://www.cert.org/advisories/CA-1999-17.html.

[19] E. Cooke, F. Jahanian, and D. McPherson. The zombie roundup: understanding, detecting, and disrupting botnets. In *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*, SRUTI'05, pages 6–6, 2005.

[20] D. Dean, M. Franklin, and A. Stubblefield. An algebraic approach to ip traceback. *ACM Trans. Inf. Syst. Secur.*, 5(2):119–137, 2002.

[21] Denial-of-Service Attack Knocks Twitter Offline. http://www.wired.com/epicenter/2009/08/twitter-apparently-down/, August 2009.

[22] D. Dittrich. The "stacheldraht" distributed denial of service attack tool, Dec. 1999 http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.

[23] D. Dittrich. The "tribe flood network" distributed denial of service attack tool, Oct. 1999 http://staff.washington.edu/dittrich/misc/tfn.analysis.

[24] D. Dittrich. The dos project's "trinoo" distributed denial of service attack tool, Oct. 1999 http://staff.washington.edu/dittrich/misc/trinoo.analysis.

[25] C. Dixon, T. Anderson, and A. Krishnamurthy. Phalanx: withstanding multimillion-node botnets. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 45–58, 2008.

[26] D.J. Bernstein. Syn cookies $http : //cr.yp.to/syncookies.html$, accessed in 2012.

[27] W. M. Eddy. RFC 4987. TCP SYN Flooding Attacks and Common Mitigations http://tools.ietf.org/html/rfc4987, 2007.

[28] T. Ehrenkranz and J. Li. On the state of ip spoofing defense. *ACM Trans. Internet Technol.*, 9(2):1–29, 2009.

[29] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 251–262, New York, NY, USA, 1999. ACM.

[30] Z. Fu, M. Papatriantafilou, P. Tsigas, and W. Wei. Mitigating denial of capability attacks using sink tree based quota allocation. In *ACM SAC'10: ACM Symposium on Applied Computing*, page To appear. ACM, 2010.

[31] S. Ganguly, M. Garofalakis, R. Rastogi, and K. Sabnani. Streaming Algorithms for Robust, Real-Time Detection of DDoS Attacks. In *Proceedings of the 27th International Conference on Distributed Computing Systems*, ICDCS '07, pages 4–, 2007.

[32] Y. Gao, Z. Li, and Y. Chen. A dos resilient flow-level intrusion detection approach for high-speed networks. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, ICDCS '06, pages 39–, 2006.

[33] G. Gu, R. Perdisci, J. Zhang, and W. Lee. Botminer: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proceedings of the 17th conference on Security symposium*, pages 139–154, Berkeley, CA, USA, 2008. USENIX Association.

[34] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, Sept. 1999.

[35] Joining Pro-WikiLeaks Attacks Is as Easy as Clicking a Button. http://www.wired.com/threatlevel/2010/12/web20-attack-anonymous/, Dec. 2010.

[36] D. Joseph, N. Shetty, J. Chuang, and I. Stoica. Modeling the adoption of new network architectures. In *Proceedings of the 2007 ACM CoNEXT Conference*, CoNEXT '07, pages 5:1–5:12, 2007.

[37] M. Kenney. Ping of death, http://insecure.org/sploits/ping-o-death.html, accessed in 2012.

[38] S. T. Kent, C. Lynn, J. Mikkelson, and K. Seo. Secure border gateway protocol (s-bgp) - real world performance and deployment issues. In *NDSS*, 2000.

[39] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: secure overlay services. *SIGCOMM Comput. Commun. Rev.*, 32(4):61–72, 2002.

[40] R. R. Kompella, S. Singh, and G. Varghese. On scalable attack detection in the network. *IEEE/ACM Trans. Netw.*, 15(1):14–25, Feb. 2007.

[41] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: methods, evaluation, and applications. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, IMC '03, pages 234–247. ACM, 2003.

[42] S. Kumar. Survey of current network intrusion detection techniques, http://www.cse.wustl.edu/~jain/cse571-07/ftp/ids/index.html.2007.

[43] A. Kuzmanovic and E. W. Knightly. Low-rate tcp-targeted denial of service attacks: the shrew vs. the mice and elephants. In *Proceedings of the 2003 Con-*

ference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM '03, pages 75–86, 2003.

[44] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '05, pages 217–228, 2005.

[45] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft. Structural analysis of network traffic flows. In *Proceedings of the joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '04, pages 61–72, 2004.

[46] M. Lesk. The new front line: Estonia under cyberassault. *Security Privacy, IEEE*, 5(4):76 –79, july-aug. 2007.

[47] J. Li, J. Mirkovic, M. Wang, P. Reiher, and L. Zhang. SAVE: source address validity enforcement protocol. In *IEEE INFOCOM 2002.*, volume 3, pages 1557–1566, June 2002.

[48] X. Liu, A. Li, X. Yang, and D. Wetherall. Passport: Secure and adoptable source authentication. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 365–378, 2008.

[49] X. Liu, X. Yang, and Y. Xia. Netfence: Preventing internet denial of service from inside out. *SIGCOMM Comput. Commun. Rev.*, 40(4):255–266, Aug. 2010.

[50] G. Loukas and G. Oke. Protection against denial of service attacks: A survey. *The Computer Journal*, 2009.

[51] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. *SIGCOMM Comput. Commun. Rev.*, 32(3):62–73, 2002.

[52] J. Markoff. Before the gunfire, cyberattacks, Aug. 2008 http://www.nytimes.com/2008/08/13/technology/13cyber.html?em.

[53] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher. *Internet Denial of Service: Attack and Defense Mechanisms (Radia Perlman Computer Networking and Security)*. Prentice Hall PTR, 2004.

[54] J. Mirkovic, A. Hussain, S. Fahmy, P. Reiher, and R. K. Thomas. Accurately measuring denial of service in simulation and testbed experiments. *IEEE Trans. Dependable Secur. Comput.*, 6(2):81–95, 2009.

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

[56] D. Moore, G. Voelker, and S. Savage. Inferring internet denial-of-service activity. In *In Proceedings of the 10th Usenix Security Symposium*, pages 9–22, 2001.

[57] W. G. Morein, A. Stavrou, D. L. Cook, A. D. Keromytis, V. Misra, and D. Rubenstein. Using graphic turing tests to counter automated ddos attacks against web servers. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, pages 8–19. ACM, 2003.

[58] A. Networks. Worldwide infrastructure security report, Volume VII 2011 http://www.arbornetworks.com/worldwide-infrastructure-security-report.html.

[59] D. S. P. Ferguson. Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing. rfc 2827. may 2000.

[60] B. Parno, A. Perrig, and D. Andersen. SNAPP: Stateless network-authenticated path pinning. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, Tokyo, Japan, mar 2008. IEEE Computer Society.

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

[62] V. Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *SIGCOMM Comput. Commun. Rev.*, 31:38–47, July 2001.

[63] T. Peng, C. Leckie, and K. Ramamohanarao. Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Comput. Surv.*, 39(1):3, 2007.

[64] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for ip traceback. *SIGCOMM Comput. Commun. Rev.*, 30(4):295–306, 2000.

[65] E. Shi, I. Stoica, D. Andersen, and A. Perrig. Overdose: A generic ddos protection service using an overlay network. *Technical report, Carnegie Mellon University, CMU-CS-06-114*, 2006.

[66] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Hash-based ip traceback. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '01, pages 3–14, 2001.

[67] D. X. Song and A. Perrig. Advanced and authenticated marking schemes for ip traceback. In *IEEE INFOCOM 2001.*, volume 2, pages 878–886 vol.2, 2001.

[68] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller. An overview of ip flow-based intrusion detection. *IEEE Communications Surveys and Tutorials*, 12(3):343–356, 2010.

[69] M. Srivatsa, A. Iyengar, J. Yin, and L. Liu. A client-transparent approach to defend against denial of service attacks. In *SRDS '06: Proceedings of the 25th IEEE Symposium on Reliable Distributed Systems*, pages 61–70, Washington, DC, USA, 2006. IEEE Computer Society.

[70] A. Stavrou and A. D. Keromytis. Countering dos attacks with stateless multipath overlays. In *Proceedings of ACM CCS*, pages 249–259, New York, NY, USA, 2005. ACM.

[71] A. Stavrou, A. D. Keromytis, J. Nieh, V. Misra, and D. Rubenstein. MOVE: An end-to-end solution to network denial of service. In *In Proceedings of the ISOC Symposium on Network and Distributed System Security (SNDSS)*, pages 81–96, 2005.

[72] A. Sudduth. The what, why, and how of the 1988 internet worm, Nov. 1988 http://www.snowplow.org/tom/worm/worm.html.

[73] D. D. Sven Dietrich, Neil Long. An analysis of the "shaft" distributed denial of service tool, Mar. 2000 http://home.adelphi.edu/\simspock/shaft_analysis.txt.

[74] N. L. Sven Dietrich, George Weaver and D. Dittrich. The "mstream" distributed denial of service attack tool, May. 2000 http://staff.washington.edu/dittrich/misc/mstream.analysis.txt.

[75] C. W. Tan, D.-M. Chiu, J. C. S. Lui, and D. K. Y. Yau. A distributed throttling approach for handling high bandwidth aggregates. *IEEE Trans. Parallel Distrib. Syst.*, 18(7):983–995, 2007.

[76] A. Tartakovsky, B. Rozovskii, R. Blazek, and H. Kim. A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods. *Signal Processing, IEEE Transactions on Signal Processing*, 54(9):3372 –3382, sept. 2006.

[77] R. Vaughn and G. Evron. DNS Amplification Attacks http://www.isotf.org/news/DNS-Amplification-Attacks.pdf, 2006.

[78] X. Wang and M. Reiter. Defending against denial-of-service attacks with puzzle auctions. In *Proceedings of Symposium on Security and Privacy*, pages 78 – 92, may 2003.

[79] X. Wang and M. K. Reiter. Mitigating bandwidth-exhaustion attacks using congestion puzzles. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 257–267, New York, NY, USA, 2004. ACM.

[80] X. Wang and M. K. Reiter. A multi-layer framework for puzzle-based denial-of-service defense. *Int. J. Inf. Secur.*, 7(4):243–263, July 2008.

[81] K. Xu, Z.-L. Zhang, and S. Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '05, pages 169–180, 2005.

[82] A. Yaar, A. Perrig, and D. Song. SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks. *IEEE Security and Privacy Symposium*, page 130, 2004.

[83] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting network architecture. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 241–252, New York, NY, USA, 2005. ACM.

[84] D. K. Y. Yau, J. C. S. Lui, F. Liang, and Y. Yam. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. *IEEE/ACM Trans. Netw.*, 13(1):29–42, 2005.

[85] Your Money or Your Site. http://www.wired.com/wired/archive/14.06/posts.html?pg=2.

[86] Q. Zhao, J. Xu, and A. Kumar. Detection of super sources and destinations in high-speed networks: Algorithms, analysis and evaluation. *Selected Areas in Communications, IEEE Journal on*, 24(10):1840 –1852, oct. 2006.