Classification of UDP Traffic for DDoS Detection

Alexandru G. Bardas
Kansas State University
bardasag@ksu.edu

Loai Zomlot
Kansas State University
lzomlot@ksu.edu

Sathya Chandran Sundaramurthy
Kansas State University
sathya@ksu.edu

Xinming Ou
Kansas State University
xou@ksu.edu

S. Raj Rajagopalan
HP Labs
raj.rajagopalan@hp.com

Marc R. Eisenbarth
HP TippingPoint
marc.r.eisenbarth@hp.com

Abstract

UDP traffic has recently been used extensively in flooding-based distributed denial of service (DDoS) attacks, most notably by those launched by the Anonymous group. Despite extensive past research in the general area of DDoS detection/prevention, the industry still lacks effective tools to deal with DDoS attacks leveraging UDP traffic. This paper presents our investigation into the proportional-packet rate assumption, and the use of this criterion to classify UDP traffic with the goal of detecting malicious addresses that launch flooding-based UDP DDoS attacks. We conducted our experiments on data from a large number of production networks including large corporations (edge and core), ISPs, universities, financial institutions, *etc.* In addition, we also conducted experiments on the DETER testbed as well as a testbed of our own. All the experiments indicate that proportional-packet rate assumption generally holds for benign UDP traffic and can be used as a reasonable criterion to differentiate DDoS and non-DDoS traffic. We designed and implemented a prototype classifier based on this criterion and discuss how it can be used to effectively thwart UDP-based flooding attacks.

1 Introduction

Denial-of-Service (DoS) attacks have been studied extensively and a large number of detection/prevention methods have been proposed by researchers [7, 8, 9, 11, 14, 18, 21, 24, 27, 29]. Despite this effort, DoS attacks remain a major threat facing enterprise networks of all sizes. Only recently have DoS attacks shifted to causing significant burdens for companies with the financial means to mitigate them. Previously, DoS attacks were rampant in for example the Asia Pacific geo-theater as well as between rival underground organizations. Such attacks have also threatened small- to medium-sized companies which had little technical or financial means to deal with the threats, often for the purpose of financial extortion. Today, however, DoS attacks have been carried out successfully against high-profile targets such as governmental agencies and major corporations, causing the cry for a practical, efficient solution to the hard problem of DoS to reach a fever pitch.

There are three major classes of DoS attacks: flooding, vulnerability, and reflector attacks [15, 22]. In this paper we focus on flooding attacks, which are attacks designed to “flood” a network or application with a large enough volume of packets or service requests so that legitimate packets or requests have to wait abnormally long in buffers and eventually get timed out or dropped. Flooding attacks are rarely launched from a single source, but rather attackers or compromised hosts act in a coordinated fashion towards a common target, forming a distributed denial of service attack (DDoS). DDoS flooding attacks using TCP protocol are well known [16, 25]. But flooding attacks using UDP have become prominent, thanks to the number of high-profile attacks launched by the Anonymous hacker group [5, 6]. Although some previously proposed methodologies for handling DDoS attacks apply to UDP, there is a lack of understanding on whether key assumptions made in these works hold for UDP traffic. Of the testing approaches described by *Mirkovic et al.* [20], namely theory, emulation, and deployment, almost all existing literature on DDoS attacks uses simulation (theory) or emulation. There has been very little if at all work that tries to understand UDP traffic characteristics in production environments for the purpose of DDoS detection and prevention.

While it is important to have re-producible experiments through emulation [10] using carefully designed realistic experimental settings, there is no substitute for experimenting a DDoS methodology on real network topologies and data. This type of experimentation has been extremely challenging, due in large part to the fact that data sharing is actively discouraged by commercial solutions or for privacy reasons. Having access to a live Internet Service Provider network as well as data from over twelve distinct networks to aid in the development of our approach, we focused our efforts on experiments on data from these production networks.

Detection and mitigation of DDoS attacks can occur close to the attack source, close to the victim, or throughout the whole Internet infrastructure. Approaches that thwart the source of the attacks have significant practical complications due to the inherent collateral damage that comes with any prevention mechanisms and lack of incentives for service providers to throttle outbound traffic that may or may not be the source of a DDoS attack. Approaches that need global cooperation from higher-level groupings of networks, known as autonomous systems, on the infrastructure level could be attractive but requires significant efforts in deployment and coordination. Compared with these, approaches that address the target (victim) networks or hosts have the unique advantage from a practical perspective, since owners of an enterprise network have the sole authority to deploy such a solution in their network and as the main beneficiaries of the protection will have the motives to do so. Thus our approach is based on a classification system that monitors an enterprise network to detect possible DDoS flooding attacks targeted at it. Our main contribution is the following:

1. We examine the "proportional packet rate" assumption proposed by past research to verify whether it holds for benign UDP packets. We do this through extensive empirical analysis of UDP traffic from a large number of production networks as well as traffic generated by popular attack tools.
2. We designed a classification algorithm for UDP traffic that aims at differentiating benign and flooding UDP flows based on the proportional-packet rate assumption. We tested a prototype implementation of the algorithm on the above mentioned traffic to understand its effectiveness.
3. We propose two operation modes of using the algorithm for the purpose of thwarting UDP-based DDoS flooding attacks, and analyze its effectiveness based on the experimental results.

The rest of the paper is organized as follows: Sections 2 and 3 cover the background and related work. Section 4 describes the main classification algorithm and its two operation modes (mirrored and inline). In Section 5 we present our experimental results including accuracy and false positive percentages of the classification algorithm. We discuss our approach's limitations and future work in Section 6, followed by the conclusion.

2 Background

Denial-of-Service attacks are a major threat to the Internet community because it causes loss of service and network connectivity to legitimate users by consuming services' network bandwidth and/or exhausting their resources [15, 22]. Attacks are usually launched from multiple coordinated machines (usually botnets controlled by malicious users) and have evolved over the years from limited and naive to sophisticated and large-scale. Attackers can rent or purchase from other attackers botnet armies, which can be made of millions of compromised hosts [1].

On the other hand, machines used to perform DDoS attacks can be controlled by individual users that coordinate their actions. Well-known examples are the attacks against Paypal, Mastercard and VISA at the end of 2010 and in 2011 [1, 2, 4]. Besides the botnet armies that they were controlling, the Anonymous group posted instructions and coordination details for other users who were willing to help. These DDoS attacks were viewed by the attackers as a form of protest, whose motives were political and ideological rather than financial [2, 5]. The last well-known successful hacktivism DDoS attacks were performed against various government and media websites in January 2012 [5]. All these recent attacks adopted UDP traffic as the main weapon.

UDP (User Datagram Protocol) is a transport-layer protocol that offers minimal transport service. The protocol does not provide reliability or datagram ordering and therefore is less onerous to the end-points. It gives applications direct access to the datagram service of the IP layer. An application program that runs over UDP has to deal with end-to-end communication issues that a connection-oriented protocol like TCP would have handled. Some of these issues include retransmission for reliable delivery, segmentation and reassembly, congestion avoidance, and flow control. UDP can almost be viewed as a null protocol because it provides only checksumming of data and multiplexing by port number. A UDP packet header contains four fields: source port, destination port, length and checksum. Only two of these fields are required (length and destination port); the other two fields (source port and checksum) are optional. UDP flooding attacks which are in active use today employ different strategies compared to their TCP and ICMP counterparts and it is due to these differences that existing DoS sensors and prevention mechanisms are either ineffective or non-applicable. Nevertheless, by attacking critical UDP services such as DNS, or even flooding the network routers and switches, these attacks can be crippling. It has become imperative that effective sensing and remediation techniques be developed to deal with UDP-based flooding attacks.

UDP flooding attacks have been used since the beginning of the DDoS attacks era (*e.g.*, Trin00 attack at the University of Minnesota). Nowadays launching a UDP-based flooding attack has become a trivial task whereas detection and response can be a painfully slow and often manual process [7]. Tools that perform flooding attacks using UDP packets (*e.g.* LOIC - Low Orbit Ion Canon or even Stacheldraht) are widely available on the web and can easily be used by even non-sophisticated computer users. Due to the stateless feature of the UDP protocol, DDoS prevention method based on connection state (*e.g.*

SYN cookies) are not applicable. Existing solutions that were proposed and implemented so far are specific to a number of UDP flooding attacks that have happened and especially to the applications that were generating the UDP traffic (*e.g.* Trin00, Stacheldraht, *etc.*), and hence can be easily evaded by modifying the UDP traffic generation pattern. To have generic and effective UDP-based DDoS attack detection and prevention, it is important to understand the characteristics of UDP traffic and the potential differences between benign flows and flows that are part of a flooding attack. If such differentiation factors can be found, they could be used to classify UDP traffic for the purpose of detecting/preventing DDoS attacks.

3 Related Work

Denial-of-service and distributed denial-of-service attacks have been pressing the Internet security and reliability for more than a decade. Therefore a significant amount of effort has been put in designing effective solutions [7, 8, 9, 11, 14, 18, 24, 27, 29] and also on how to test DDoS defenses [19, 20, 23]. Our work is inspired by these previous works and we tried to conduct thorough analysis of UDP traffic (both benign and flooding traffic) on a large number of production systems to understand the validity of key assumptions made and effectiveness of our approach.

Gil and Poletto [11] hypothesized that under normal operations, the packet rate of traffic going in one direction is proportional to the packet rate of traffic going in the opposite direction. For example, under TCP protocol an acknowledgement packet is sent for every n packets received. They designed the MULTOPS [11] system based on this assumption which enables routers or network monitors to detect outgoing or incoming bandwidth attacks. Our approach is inspired by this assumption and we conducted extensive empirical study on data from a large range of production networks and that generated by popular DDoS attack tools, to validate using this assumption in UDP-based flooding attack detection. The MULTOPS work tried to address all types of bandwidth attacks and was designed to be deployable on the whole Internet infrastructure. Some important details and use cases were not discussed. For instance whether IP addresses are blocked for an indefinite amount of time. Moreover this approach was tested mainly in a simulated environment and not significantly on live data from production networks.

The NOX/OpenFlow platform [7], DCD [8], Secure Overlay Services (SOS) [14], SIFF [28], and TVA [29] are various DoS-limiting protocols or network architecture setups that require special network devices and functions. Our classifier is a local detection method that is meant to work on existing networks without changing the architecture. Stop-It [18] is a DoS-resistant network architecture that is focused on how to block a DoS attack after it was detected. This architecture complements our classifier and can be used for forensics purposes. DefCOM [24] is a framework for DDoS defense cooperation. This framework enables source, core and destination defenses to securely communicate during attacks and plan collaborative defenses. For detection purposes this framework also considers proportional packet rates. Speak-Up [27] is a defense against application-level DDoS attacks performed using legitimate-looking requests that consume computational resources like CPU cycles or disk space. The underlying assumption for the Speak-Up protocol is that compromised machines (bots) have lower available bandwidth than legitimate hosts. Proactive Surge Protection (PSP) [9] aims to provide a broad first line of defense against bandwidth DDoS attacks and minimize collateral damage (blocked legitimate users) by providing bandwidth isolation between traffic flows. This defensive mechanism is designed to protect traffic on ISP links by observing an ISP topology as a set of origin-destination pairs. Another useful solution that can be utilized at the ISP level is described by Sengar *et al.* [26]. This paper proposes a practical answer using behavioral distance to the problem of online detection of traffic anomalies at high speed. Li and Lee [17] present another interesting approach based on the assumption that abnormal traffic behavior imposed by DDoS attack can be detected via energy distribution based on wavelet analysis. However this method cannot deal with unknown behavior in a proactive way and therefore for better performance a cooperation with a detection mechanism is encouraged by the authors. Hussain, *et al.* [12] propose a framework to classify DDoS attacks into single- and multi-source attacks based on header analysis, ramp-up behavior and spectral analysis. Furthermore hop-count filtering [13] can be used to detect spoofed DDoS traffic. These methods together with the existing ingress/egress filtering can be used with our classifier for making sure that the collateral damage is minimized. All in all the above mentioned DDoS defensive solutions have been mostly tested and adjusted on synthetic data. Moreover, the proportional packet rates assumption used in some of the above mentioned works has been applied in general to TCP packets and not to UDP. In addition to emulation experiments our classifier was tested on datasets from a large number of real production networks and the off-line version is currently running on span links (mirrored traffic) on a couple of production networks.

4 Methodology

A naive approach to deal with flooding traffic generated by automated tools is to match the packet content with well-known patterns from these tools. For example, most DDoS tools that generate UDP flooding traffic (*e.g.*, LOIC, Stacheldraht, Trin00) use the same payload for every packet. However without too much effort these tools can be modified to use a different payload for each packet which makes pattern-based detection very hard. We were able to modify LOIC (v.1.0.6) so that it uses a unique random payload for every UDP packet. Our tests showed that our modified version sends at approximately half the speed as the original version, which is acceptable performance. Thus to effectively detect DDoS flooding attacks, one has to go beyond simple packet payload patterns.

Proportional-packet rate assumption

Even though UDP does not provide delivery confirmation, in benign UDP-based applications there is usually a mechanism implemented in the application layer that makes sure the other party is still live and/or received the sent packets, similar to TCP ACK. Our approach is based on the assumption that under normal operations, the packet rate in one direction is proportional to the packet rate in the opposite direction [11]. Under normal circumstances there are dialogues where two or more parties communicate (even if one side is "dominating" the communication). In our experiments we found that the vast majority of UDP-based applications follow this proportional-packet rate rule.[1] However we did observe exceptions in some UDP-based applications, which adopt a short sender-receiver initial two-way communication, followed by a one-way burst of UDP packets (*e.g.*, the Session Initiation Protocol - SIP and the T.38 protocol for fax). This needs to be handled which is explained in our algorithm below. Moreover, our approach must monitor traffic in both directions. Therefore the case of asymmetric routes has to be addressed separately.

4.1 Basic Approach

Our basic classification algorithm (Algorithm 1) works as follows. For every source IP address that sends UDP packets to a monitored destination IP address (or subnet), we calculate a ratio function within a pre-set time window. The ratio function is defined as follows.

Let $\alpha(src\_addr)$ be the number of packets sent *from* src_addr to the monitored address and $\beta(src\_addr)$ be the number of packets sent *to* src_addr from the monitored address, then:

$$ratio(src\_addr) = \begin{cases} \dfrac{\alpha(src\_addr)}{\beta(src\_addr)} & \text{if } \beta(src\_addr) \neq 0 \\ \alpha(src\_addr) & \text{if } \beta(src\_addr) = 0 \end{cases}$$

[1]Note that packet rate is different from data rate. When measuring packet rates we only count the number of packets sent per time unit; the sizes of these packets do not affect the metric.

The intuition is that under normal operations, the number of packets sent from the source to the monitored target will be proportional to the number of packets sent from the target to the source, due to the proportional-packet rate assumption. Thus the ratio function shall be below some maximum threshold. If a source IP is launching a flooding attack against the target, the number of packets sent from the source will far exceed the number sent to it by the target. Thus the ratio function will be higher for a flooding source address than a benign one.[2] The $\beta(src\_addr) = 0$ case captures the benign one-way communication pattern as illustrated by the SIP and T.38 protocols described above. In such cases the ratio function becomes the packet rate sent from the source to the monitored target. In those benign applications the packet rate will be limited as well during the one-way communication period.

Algorithm 1 Basic Approach

```
1: for each $src\_addr$ do
2:     $r \leftarrow ratio(src\_addr)$
3:     if within $t$ seconds $r \geq max\_ratio$ then
4:         $waiting\_queue \leftarrow src\_addr$ {For inline mode}
5:         OR
6:         print "Alert($src\_addr$)" {For mirrored mode}
7:     end if
8: end for
```

The algorithm keeps track of the ratio function for every source address that has sent UDP packets to the monitored target. If the ratio sampled in a time window t (called bucket TTL, or BTTL) exceeds the predefined threshold, this signals that the source address may be launching a flooding attack. In the mirrored mode an alert will be issued; in the inline mode the source IP will be put into a "waiting queue" where further packets sent from it are blocked or rate limited for a certain time period. Rate limiting is accomplished by dropping a portion of the packets, but still giving the offending IP address a chance to communicate if it adjusts its behavior, *e.g.*, by slowing down the sending speed and waiting for the application's acknowledgment, something that many legitimate UDP applications would normally do. One could also employ a backoff scheme in which the first infraction causes an IP address to be blocked or rate-limited for a short time and the waiting time increases if the address continues to send packets in an uncontrolled manner. The ratio function is updated every time a packet is sent or received by the monitored target. It is reset when the BTTL expires and a new sample begins.

[2]The implicit assumption here is that the attacking traffic will try to occupy a higher bandwidth compared with a benign one. If this is not the case, then the attacking source becomes indistinguishable from the benign ones. We believe this is a fundamental limit of any DDoS detection method.

4.2 Advanced Inline Approach

When dealing with a reliable network where packet loss rate is considered very low, a benign application may opt to use NACK packets instead of ACK packets for delivery confirmation. That is, instead of sending an ACK packet for every *n* packets received, the receiver sends a non-acknowledgement packet (similar to the TCP NACK packets) in case it does not receive anticipated data. The NACK behavior would break the proportion-packet rate assumption if the network communication is highly reliable (and thus very few NACK packets are sent). Although we have not observed such NACK behavior in our analysis of UDP traffic in production network data, we believe it is a legitimate concern that needs to be addressed to reduce the likelihood of false positive produced by the classifier. Thus we designed an advanced algorithm (Algorithm 2) for the inline mode that can appropriately handle NACK behaviors in benign UDP traffic.

Algorithm 2 Advanced Inline Approach

```
1:  for each $src\_addr$ do
2:    $r \leftarrow ratio(src\_addr)$
3:    if within $t$ seconds $r \geq max\_ratio$ then
4:      if $src\_addr$ has exhibited NACK behavior before then
5:        $NACK\_queue \leftarrow src\_addr$
6:      else
7:        $waiting\_queue \leftarrow src\_addr$
8:      end if
9:    end if
10: end for
```

The main difference in the advanced approach is that when the ratio threshold is exceeded, rather than putting the source IP to the waiting queue, we will put the source IP into a “NACK_queue” if it has exhibited NACK behavior before. This can be known by keeping a per-flow flag which is set when a packet is sent *from* the monitored target to the source after the source is blocked. The difference between the NACK_queue and the waiting_queue is that source IP’s in the NACK_queue will be immediately released if a packet is sent from the target to the source (presumably a NACK packet), whereas source IP’s put in the waiting_queue will be blocked or rate-limited for the specified waiting time.

5 Experimentation

5.1 Experimental Setup

We implemented both the mirrored and inline versions of the classifier in C++ and currently were able to extensively test only our mirrored version that runs on captured traffic, including that from a large number of production networks.

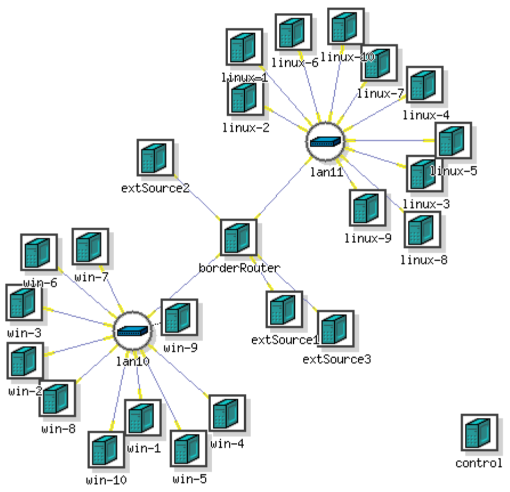

Figure 1: DeterLab Setup

DeterLab (http://deter-project.org/) We used the DeterLab to construct a setup with two subnets connected to a border router, which is also connected to three external resources (Figure 1). On this setup we generated benign traffic, attack traffic and a mix of the two. We used the Security Experimentation EnviRonment (SEER) [3] to generate benign traffic and installed LOIC on some Windows machines for generating attack traffic.

Argus testbed We used a testbed built in the Argus cybersecurity lab at Kansas State University that consists of 20 hosts (mainly i3-2100T CPU @2.5GHz with 4GB of RAM). On this testbed we conducted various experiments and measurements. First we configured and installed various UDP applications and network games so we can measure the ratio function of these benign UDP applications’ traffic. Next we installed DoS tools (mainly LOIC and modified LOIC) and measured the ratio function for the UDP attack traffic. We tested our classifier on the mixed benign and attack traffic.

Production networks We captured traffic from Kansas State University Computing and Information Sciences departmental network to test our classifier. We also have access to non-DNS UDP packet captures from twelve other distinct networks whose names cannot be revealed. These data contain traffic from large corporations (edge and core), ISPs, universities, financial institutions, *etc*, and were captured between 2002 and 2011. The data from the ISP network analyzed in this paper is from the last week in January 2012.

5.2 Experimental Results

5.2.1 Validating the proportional-packet rate assumption

In order to validate the proportional-packet rate assumption stated in Section 4, we used various sampling time intervals (BTTL) to determine the maximum ratio value for the analyzed datasets. The first dataset we used is that generated on our testbed by various UDP applications: DNS, NFS, DHCP, NTP, QQ IM, MSN IM, PPlive, PPstream, Sopcast, QQlive, Counter-Strike 1.6, Quake 3 Arena, and a few others. We also mixed this traffic with other network traffic. Next we used datasets generated by the benign traffic generators on our DeterLab setup. Last but probably the most important part was to analyze the data from the various production networks (Table 1 - rough estimation of the # of flows and the traffic duration). The experimental results are summarized in Table 2. The maximum ratio is determined by finding the lowest threshold value for our classifier that would not generate false alarms for the specific dataset. The datasets labeled with "*" (C, F, M, U, UN) are influenced by asymmetric routes and thus the measurement cannot be fully trusted. For instance dataset F contains NFS v3 traffic over UDP; there are a large number of requests but no single response. Another example is dataset U where we could see SNMP v1 protocol responses but no requests. Therefore the results registered on these datasets are likely to be inaccurate. The datasets labeled with "+" (D and O) contain UDP protocols where by protocol design no response message is necessary: Syslog over UDP protocol and older versions of Netflow. These protocols do not implement any delivery confirmation or retransmission mechanism. Some interesting observations from these results: 1) the ratio function generally increases with sampling time (BTTL), but not always; 2) for benign traffic with small sampling time (5 sec or shorter), the maximum ratio value is a few hundreds (excluding the special cases marked by * and +).

5.2.2 Ratio function for UDP attack traffic

We measured the rate (packets per second) for a few UDP flooding tools (LOIC and modified LOIC) and also the ratio functions at the victim's side. These measurements were performed on the ArgusLab 20-host testbed, the DeterLab setup, and on a suspected DDoS attempt (or an application anomaly) in one of the analyzed datasets from production networks. On our testbed set up one machine running LOIC at the highest speed sends approximately 220,000 packets per second (pkts/sec) to the victim and approximately 250 pkts/sec at the lowest speed. The victim machine (Ubuntu 11.10) will at most send six ICMP packets back to the attacking IP address in the first second and then it will not send anything back. This means that the ratio function will grow significantly as more incoming packets are received, especially after the first data sample (bucket) expires. On DeterLab the flooding tool sending rate is approximately 75,000 pkts/sec at the highest speed and 200 pkts/sec at the lowest. The two attack attempts discovered in the ISP dataset contained 12,000 packets that were sent in around two minutes. Table 3 summarizes the test results. However for a flooding attack to be effective the aggregate attack packet rate arriving at the victim needs to be high enough, depending on the network bandwidth and the computation capabilities of the victim. We used the DoS metric tool by Mirkovic, *et al.* [20] to measure the QoS of a service with and without attack traffic. We tried to cripple a web server by sending a flood of UDP packets from multiple sources. In this way we measured the ratio value needed for a DoS/DDoS attack to be effective on our testbed (Table 4). The table shows that to achieve 100% successful DoS attack on our victim, a ratio value of about 900,000 is needed. This is significantly higher than the maximum ratio value observed in benign traffic (Table 2). Table 3 also shows that when the attack traffic is slowly sent there might be more time needed to detect it (*e.g.*, the ISP attack traffic).

5.2.3 Performance, accuracy, and calibration

We established various threshold ratios and measured the false-positive rates (the proportion of source addresses misclassified as attack addresses) on the various datasets (see Table 5). Even though there are two thresholds (BTTL = 1 sec with RatioMax = 10,000 and BTTL = 100 sec with RatioMax = 15,000) that obtained 0% false positive for all datasets, we cannot conclude that these shall be the recommended parameters. For example both of them miss the attack attempt in the ISP dataset due to the high ratio cutoff. There are certain trade-offs that have to be considered. A high ratio cutoff may result in a high number of false negatives and a low BTTL enables the attackers to use less resources when performing a DDoS attack to evade the detection. On the other hand a high BTTL may consume more resources on the detector side. Therefore these parameters need to be established based on the type of service a potential victim hosts, the type of UDP protocol applications used on a network and the objective of the owner. The BTTL and the predefined maximum ratio are tunable parameters that should be calibrated on the deployed environment. For example one can run the classifier under normal operating conditions to measure the traffic characteristics similar to those shown Table 2 and then set up the cutoff ratio accordingly. A good idea might be to have different instances of the classifier with different parameters and varying actions for monitoring the same potential target and employ a mixture of rate limiting and blocking.

6 Discussion

6.1 Limitations

Malicious users can optimize bandwidth at their disposal by spoofing source IP addresses, which presents a challenge for our approach. Recent events have shown however that a large number of ISPs perform egress filtering to combat blatant address spoofing. Attackers had to use flooding tools (e.g LOIC) that do not spoof the source

IP address. Similarly, historical challenges around the area of NAT (Network Address Translation) which ail the majority of network-based defense systems, could cause problems in the case where malicious and legitimate users share the same public IP address. As mentioned in the Related Work section we are considering to address these limitations by using the approaches described in [12, 13].

6.2 Future Work

The inline version of the classifier is under development. We are working on integrating it with an existing appliance to be deployed in a number of production networks. The inline version of our classifier is capable of detecting and stopping ongoing DDoS attacks.

7 Acknowledgments

We would like to thank Eugene Vasserman for his feedback and suggestions on various classifier use-cases including the NACK behavior. Mirko Montanari provided substantial materials that helped us gain a comprehensive understanding of past DDoS work. This material is based upon work supported by U.S. National Science Foundation under award no. 1038366 and 1018703, and HP Labs Innovation Research Program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, or Hewlett-Packard Development Company, L.P.

8 Conclusion

We designed a UDP traffic classification algorithm based on the proportional-packet rate assumption, for the purpose of detecting UDP flooding attacks. We thoroughly examined the assumption through experimentation on a large number of data sets from production networks and various testbeds. The experimentation results provide critical clues on how this classification algorithm can be used for DDoS detection and reveal some key observations on its effectiveness.

References

[1] Botnets - Herds of Internet Creatures Running Amuck. http://www.securityweek.com, 2010.

[2] DDoS Top 2011. http://www.net-security.org, 2011.

[3] Deter SEER Wiki. http://seer.deterlab.net/trac, 2011.

[4] DoS and DDoS Yesterday and Today. http://dvlabs.tippingpoint.com, 2011.

[5] Hackers Step Up Attacks After Megaupload Shutdown. http://bits.blogs.nytimes.com/2012/01/24/, 2012.

[6] National Cyber Alert System - Anonymous DDoS Activity. http://www.us-cert.gov/cas/techalerts/TA12-024A.html, 2012.

[7] R. Braga, E. Mota, and A. Passito. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In *35th IEEE Conference on Local Computer Networks (LCN)*, 2010.

[8] Y. Chen, K. Hwang, and W. Ku. Collaborative Detection of DDoS Attacks over Multiple Network Domains. In *IEEE Transactions on Parallel and Distributed Systems Journal*, 2007.

[9] J. Chou, B. Lin, S. Sen, and O. Spatscheck. Proactive Surge Protection: A Defense Mechanism for Bandwidth-Based Attacks. In *IEEE/ACM Transactions on Networking Journal*, 2009.

[10] C.E. Gates. A Case Study in Testing a Network Security Algorithm. In *Proceedings of the 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities*, 2008.

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

[12] A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2003.

[13] C. Jin, H. Wang, and K.G. Shin. Hop-Count Filtering: An Effective Defense Against Spoofed DDoS Traffic. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, 2003.

[14] A.D. Keromytis, V. Misra, and D. Rubenstein. SOS: An Architecture For Mitigating DDoS Attacks. In *IEEE Journal on Selected Areas in Communications*, 2004.

[15] A. Keshariya and N. Foukia. DDoS Defense Mechanisms: A New Taxonomy. *Defense*, 5939 LNCS, 2010.

[16] F. Lau, S.H. Rubin, M.H. Smith, and L. Trajkovic. Distributed Denial of Service Attacks. In *In IEEE International Conference on Systems, Man, and Cybernetics*, 2000.

[17] L. Li and G. Lee. DDoS Attack Detection and Wavelets. In *Proceedings of the 12th International Conference on Computer Communications and Networks (ICCCN)*, 2003.

[18] X. Liu, X. Yang, and Y. Lu. To Filter or to Authorize: Network-Layer DoS Defense Against Multimillion-node Botnets. In *ACM SIGCOMM*, 2008.

[19] J. Mirkovic, E. Arikan, W. Songjie, S. Fahmy, R. Thomas, and P. Reiher. Benchmarks for DDoS Defense Evaluation. In *Military Communications IEEE Conference (MILCOM)*, 2006.

[20] J. Mirkovic, S. Fahmy, P. Reiher, and R.K. Thomas. How to Test DoS Defenses. In *Conference on Cybersecurity Applications Technology for Homeland Security (CATCH)*, 2009.

[21] J. Mirkovic, G. Prier, and P.L. Reiher. Attacking DDoS at the Source. In *Proceedings of the 10th IEEE International Conference on Network Protocols*, 2002.

[22] J. Mirkovic and P. Reiher. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34, 2004.

[23] J. Mirkovic, B. Wilson, A. Hussain, S. Fahmy, P. Reiher, R. Thomas, and S. Schwab. Automating DDoS Experimentation. In *Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test*, 2007.

[24] G. Oikonomou, J. Mirkovic, P. Reiher, and M. Robins. A Framework for a Collaborative DDoS Defense. In *22nd Annual Computer Security Applications Conference (ACSAC)*, 2006.

[25] T. Peng, C. Leckie, and K. Ramamohanarao. Survey of Network-based Defense Mechanisms Countering the DoS and DDoS Problems. *ACM Comput. Surv.*, 39, 2007.

[26] H. Sengar, X. Wang, H. Wang, D. Wijesekera, and S. Jajodia. Online Detection of Network Traffic Anomalies Using Behavioral Distance. In *17th International Workshop on Quality of Service (IWQoS)*, 2009.

[27] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker. DDoS Defense by Offense. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2006.

[28] A. Yaar, A. Perrig, and D. Song. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In *IEEE Symposium on Security and Privacy*, 2004.

[29] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting Network Architecture. In *Proceedings of conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2005.

Table 1: Datasets Information

	C	CO	D	F	G	ISP	M	O	S	T	UN	U
≈ Duration	21 *min*	5 *sec*	1 *sec*	11 *min*	23 *sec*	6 *days*	10.5 *hours*	3 *min*	6 *min*	9 *min*	5 *min*	12 *min*
≈ # of flows	67098	1268	658	2216	203	45710	848	1375	25773	23252	8396	258325

Table 2: Legitimate traffic measurements; Legend: * - asymmetric routes; + - one-way protocols; # - without attack traffic

BTTL in sec	Max. Ratio for Synthetic Benign Data		Max Ratio for Benign Data form Production Networks												
	ArgusTestbed	DeterLab	C^*	CO	D^+	F^*	G	$ISP^\#$	M^*	O^+	S	T	UN^*	U^*	K-State CIS Dept.
1	51	281	280	85	1090	9990	9	85	5300	330	80	41	268	2610	200
2	41	411	550	135	1305	10500	18	85	5300	560	95	28	471	2600	300
3	54	560	650	141	1305	10400	21	85	5300	580	115	28	471	2600	450
4	43	728	935	141	1305	10200	21	85	5300	790	121	30	505	2600	600
5	42	778	1200	141	1305	10300	21	85	5300	915	129	41	530	2600	700
100	42	1230	7300	820	1305	13400	30	85	5300	11600	1680	122	2470	3150	2500
600	12	1230	17000	1070	1305	26700	30	285	5300	20300	2600	512	2470	8070	2500

Table 3: Attack traffic is present and BTTL = 3 sec

Attack duration in sec	ArgusTestBed (with LOIC traffic)		DeterLab (with LOIC traffic)		ISP Dataset with attack traffic
	Ratio at Highest Speed	Ratio at Lowest Speed	Highest Speed	Lowest Speed	
1	≈ 44,000	≈ 250	≈ 75,000	≈ 97	≈ 21
2	≈ 88,000	≈ 500	≈ 150,000	≈ 200	≈ 39
3	≈ 132,000	≈ 750	≈ 225,000	≈ 300	≈ 45
4	≈ 220,000	≈ 250	≈ 75,000	≈ 100	≈ 64
5	≈ 430,000	≈ 500	≈ 150,000	≈ 200	≈ 84
6	≈ 660,000	≈ 750	≈ 225,000	≈ 300	≈ 99
34	≈ 220,000	≈ 250	≈ 75,000	≈ 100	≈ 100
35	≈ 430,000	≈ 500	≈ 150,000	≈ 200	≈ 218
36	≈ 660,000	≈ 750	≈ 225,000	≈ 300	≈ 349

Table 4: DoS Metrics [20] for a Web Server

Percentage of Failed Transactions (PFT)	Successes	Failures	QoS	DoS	# of UDP attack packets
0%	14	0	98.5526%	0%	0
66.6667%	3	6	89.1839%	70.547191%	≈ 220,000
100%	0	6	0%	83.562626%	≈ 660,000
100%	0	3	0%	100%	≈ 880,000

Table 5: False-Positives percentages for different cutoff examples; Legend: * - asymmetric routes; + - one-way protocols

Cutoff		Synthetic Data		Data form Production Networks												
RatioMax	BTTL in sec	ArgusTestbed	DeterLab	C^*	CO	D^+	F^*	G	ISP	M^*	O^+	S	T	UN^*	U^*	K-State CIS Dept.
100	1	0%	37.5%	4.3%	0%	2%	76.6%	0%	0%	28.75%	3.3%	0%	0%	37.7%	1%	1%
10000	1	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
500	3	0%	6.25%	0.3%	0%	2%	63.3%	0%	0%	12.5%	3.3%	0%	0%	0%	0%	0%
15000	100	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
20000	600	0%	0%	0%	0%	0%	3.3%	0%	0%	0%	3.3%	0%	0%	0%	0%	0.5%