STONE: A Stream-based DDoS Defense Framework

Mar Callau-Zori
Universidad Politécnica de Madrid, Spain
mcallau@fi.upm.es

Vincenzo Gulisano
Universidad Politécnica de Madrid, Spain
vgulisano@fi.upm.es

Zhang Fu
Chalmers University of Technology, Gothenburg, Sweden
zhafu@chalmers.se

Ricardo Jiménez-Peris
Universidad Politécnica de Madrid, Spain
rjimenez@fi.upm.es

Marina Papatriantafilou
Chalmers University of Technology, Gothenburg, Sweden
ptrianta@chalmers.se

Marta Patiño-Martínez
Universidad Politécnica de Madrid, Spain
mpatino@fi.upm.es

ABSTRACT

An effective Distributed Denial of Service (DDoS) defense mechanism must guarantee legitimate users access to an Internet service masking the effects of possible attacks. That is, it must be able to detect threats and discard malicious packets in a online fashion. Given that emerging data streaming technology can enable such mitigation in an effective manner, in this paper we present *STONE*, a stream-based DDoS defense framework, which integrates anomaly-based DDoS detection and mitigation with scalable data streaming technology.

With *STONE*, the traffic of potential targets is analyzed via continuous data streaming queries maintaining information used for both attack detection and mitigation. *STONE* provides minimal degradation of legitimate users traffic during DDoS attacks and it also faces effectively flash crowds. Our preliminary evaluation based on an implemented prototype and conducted with real legitimate and malicious traffic traces shows that *STONE* is able to provide fast detection and precise mitigation of DDoS attacks leveraging scalable data streaming technology.

Categories and Subject Descriptors

H.2 [**Information Systems**]: Database Management

General Terms

Algorithms, Experimentation, Performance, Security

Keywords

DDoS Detection and Mitigation, Data Streaming

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SAC'13 March 18-22, 2013, Coimbra, Portugal.
Copyright 2013 ACM 978-1-4503-1656-9/13/03 ...$10.00.

1. INTRODUCTION

A *Distributed Denial of Service* (DDoS) attack in communication networks attempts to exhaust the computational resources of a server making its service unavailable to legitimate users. A typical attacking method is packet flooding, conducted sending thousands of packets per second to a target host to congest its network links. The high volume of illegitimate traffic is usually generated by controlling a collection of compromised machines connected to the Internet (botnets). DDoS detection and mitigation techniques are challenging due to the vast and evolving range of big scale attacks. A successful defense mechanism must filter out malicious traffic effectively, minimizing legitimate users service degradation. Furthermore, the filtering mechanism should stand constantly to react immediately to threats and should incur in a negligible latency overhead (especially when there are no attacks being perpetrated). We foresee that data streaming is the right paradigm to address the requirements of mitigation of DDoS attacks, especially, due to the recent advances in the field that have yielded elastic and scalable data streaming engines [6]. In this paper we leverage such elastic and scalable data streaming technology that enables the creation of *continuous queries* that constantly process incoming data and produce results in a real-time fashion, scaling as needed to deal with DDoS attacks.

Prior work.

In recent years the need for effective solutions against DDoS attacks has made it an active and important research topic. Two main types of approaches exist for network intrusion detection [9]: 1) signature-based and 2) anomaly-based solutions. Signature-based solutions [13] check each incoming packet to verify its signature and decide whether to forward or discard it. They have several limitations: not all the protocols can be signed and, for each new type of attack, a new type of signature may be necessary [11]. Contrary to signature-based ones, anomaly-based solutions, adopted in the context of threats where every malicious packet may seem legitimate if analyzed individually, attempt to cover a wider spectrum of attacks by spotting deviations between current and reference traffic behavior [9]. Plenty of anomaly-based solutions provide detection or mitigation of multiple types of attacks relying on complex analysis of the traffic features [3, 11, 15]. The challenge lies in defining a traffic

analysis that is amenable for on-line processing (solutions based on mining tools as [11] are best suited to study rather than detect threats) but accurate enough to properly mitigate attacks (solutions that simply look at the overall traffic volume [1, 10, 14] provide no insights about what to discard during an attack).

Our contribution.

We present *STONE*, a stream-based DDoS defense framework implemented on top of StreamCloud [5, 6], an elastic parallel-distributed Stream Processing Engine (SPE). With *STONE*, network traffic features are analyzed to both detect and mitigate threats. One of the novel features of *STONE* is its smooth shaping of a host's traffic volume even in the presence of legitimate abrupt changes (i.e., during a *flash crowd*, *STONE* avoids the saturation of the protected host while prioritizing sources communicating frequently with it). Filtering decisions are taken for groups rather than individual machines applying prefix level traffic aggregation. Thus, all the machines that belong to a small network whose traffic is legitimate will have equal chances of communicating with the protected host in the presence of attacks. We provide an evaluation of *STONE* conducted using data traces based on the network traffic data from CAIDA [7] and SUNET [16, 12]. The results achieved by *STONE* prototype show quick detection and effective mitigation capabilities, thus making *STONE* a promising solution for mitigating DDoS attacks. Our contributions can be summarized as follows:

1. STONE: an anomaly-based defense solution that provides both detection and mitigation of DDoS attacks.
2. A novel traffic analysis approach based on data streaming and implemented on top of StreamCloud, an elastic parallel-distributed SPE.
3. An evaluation based on a real prototype and conducted using real legitimate and malicious traffic traces.

The rest of the paper is organized as follows: the system model is introduced in Section 2. In Section 3, we discuss the architecture of *STONE*. The evaluation is presented in Section 4. Finally, conclusions are presented in Section 5.

2. SYSTEM MODEL

In this section, we introduce the network and stream model, we define the adversary model and state the problem *STONE* aims to solve.

Network and Stream model. The network comprises four kinds of entities: (1) *protected entities*, network hosts that can be attacked; (2) *legitimate hosts*, end-hosts who consume protected entities services; (3) *STONE* machines, used to run *STONE*; and (4) *bots*, network hosts controlled by the attacker. The protected entities, legitimate hosts and the bots are connected via network links and routers, while *STONE* machines form a separated private network that cannot be reached by the attacker. Note that *STONE* can be used for protection of both single host and multiple hosts. It can also be extended for deployment in the framework for traffic control and isolation against DDoS problem and protecting network resources [4]. Due to the space constraint, we do not address such issue in this paper. In the following, we study how *STONE* behaves considering only the traffic being sent to a specific protected entity.

STONE defines two input data streams: *network stream* S and *aggregated network stream* S_a. S represents the flow of packets sent to a protected entity; each packet can be seen as a tuple $\langle srcIP, bytes\rangle$. Attributes $srcIP$ and *bytes* represent the source IP address and the size of the packet, respectively. S_a tuples contain aggregated information of S packets on a per-$srcIP$ basis over periods of time and are composed by attributes $\langle srcIP, ts_A, ts_B, packets, bytes\rangle$ (e.g., given period 8:00:00-8:00:30, tuple $\langle A, 8{:}00{:}12, 8{:}00{:}25, 5, 250\rangle$ states that A sent 5 packets -250 bytes- starting from 8:00:12 to 8:00:25). Stream S_a can be created from S using monitoring applications such as Cisco Netflow, which is widely supported by network devices or ISPs.

In data streaming, incoming data (tuples) is processed on-the-fly by *continuous queries*, defined as directed acyclic graphs of operators. Results are computed over the most recent window of tuples (e.g., tuples received in the last 5 minutes). Windows cover overlapping periods of *size* time, *advance* time units far from each other. For instance, a window of size and advance of 30 and 10 minutes, respectively, will cover periods [8:00:00-8:30:00), [8:10:00-8:40:00) and so on.

Adversary model. STONE aims at defending against packet-flooding based DDoS attacks, e.g., SYN flood and UDP flood attacks. The adversary can use different types of packets in the attacks, such as TCP packets, UDP packets, ICMP packets, etc. However, we assume that the adversary has no knowledge about the characteristics of the traffic to the victim, such as distribution of source addresses, the number of flows and the rates of flows. In other words, the adversary can hardly launch an attack without disproportional changes of the victim's traffic features. With respect to the reference information maintained by *STONE*, we assume that the attacker cannot modify nor pollute it. We stress that preventing the pollution of the reference information is orthogonal to the task of using it in order to detect attacks.

Problem formulation. Given a protected entity and its maximum load L, the goal of *STONE* is to monitor its traffic to (a) detect possible threats and (b) to shape its volume when it exceeds a threshold load αL. Whenever filtering is applied, *STONE* must maximize the percentage of legitimate traffic forwarded to the protected entity. As discussed in the introduction, *STONE* can be used to protect the host from real threats as well as legitimate peak loads (e.g., flash-crowds). For this reason, we use the term legitimate traffic in a global way to refer to sources that are either frequently communicating with the victim host or that were communicating with it before the attack or peak load started. That is, during a peak load, the legitimate traffic is the one generated by the usual clients of the host. The defense mechanism must ensure that the malicious IPs that reach the protected entity before detecting the attack have no way to exceed its maximum load. The challenge lies in which criteria to use to discard packets in S guaranteeing that the overall traffic volume does not exceed L while forwarding as much legitimate traffic as possible.

The reason why *STONE* filters traffic only if it exceeds αL is two-fold: on one hand, our solution is not intended to analyze the cause of the anomaly and does not distinguish between legitimate or illegitimate traffic spikes; on the other hand, forwarding potentially malicious traffic when αL is not

exceeded makes harder for the attacker to adapt the attack depending on how the system is reacting to it.

3. STONE ARCHITECTURE

In this section we present the components in charge of analyzing the protected host traffic in order to the detect possible threats and in charge of shaping its traffic volume when it exceeds αL.

Figure 1 shows *STONE* architecture. The *Detection Control Center* (DCC) is the subsystem in charge of detecting attacks. It consumes S_a and compares its current features with the reference features maintained at the *Historic Dataset* (HD). The *Mitigation Center* (MC) is placed between the incoming stream S and the protected entity and takes care of filtering the traffic if it exceeds αL. Its output stream S_m is equal to S whenever the incoming load is lower than αL or a subset of S when filtering is applied. The filtering criterion is determined by the DCC. Whenever the traffic of the protected entity is below αL, the MC is not active and simply forwards S packets, resulting therefore in a negligible overhead for the protected entity.

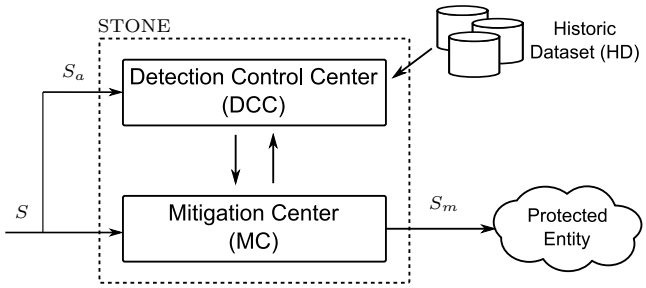

Figure 1: *STONE* architecture

STONE computes traffic features aggregating together information of multiple IPs. More precisely, information is maintained for groups of source IPs, referred as *source clusters* ($srcCL$), sharing the same prefix of b bits of the IP addresses. The reason why features are not maintained on a per-$srcIP$ basis is threefold: (a) the huge number of IPs connecting to an entity might render the protocol impractical, (b) the traffic exchanged by individual IPs might be negligible with respect to the overall traffic, making thus comparison between current and reference features not reliable and (c) decisions taken for source clusters (i.e., physically close machines for small source clusters) allow for smoother mitigation (e.g., forwarding traffic from all the nodes of a legitimate network rather than only a subset of them).

3.1 Detection Control Center (DCC)

The DCC is fed with the tuples of stream S_a. The features of sources belonging to the same source cluster are aggregated together so that each source cluster i ($srcCL_i$) is represented by features $f_i = (\phi_i, \omega_i, \tau_i)$, where ϕ_i represents the average number of packets per flow, ω_i represents the average amount of bytes per flow and τ_i represents the average elapsed time per flow. We remark that, being based on temporal windows, this information refers to the most recent fraction of data. For instance, ϕ_i could represent the average number of packets per flow sent to the protected entity from $srcCL_i$ during the last hour.

STONE analyzes how source clusters behave by partitioning them into groups and studying how the frequency of each group evolves over time. Concretely, *STONE* splits the space into eight different groups $\{G_0, \ldots, G_7\}$ and maintains the number of source clusters belonging to each group. Source clusters are partitioned into groups comparing their features with the reference point $O = (O_\phi, O_\omega, O_\tau)$ (Figure 2). The reference point cannot be fixed as an absolute value but must be instead computed depending on the traffic features of the entity being protected. For this reason, the reference point O is computed over past features (maintained at the HD) as the 0.95-quantile value of each feature. The 0.95 value is motivated by previous studies [8] stating that more than 90% of the traffic flows are *mice* flows with small number of packets. In our approach, source clusters sending *mice* flows to the protected entity belong to group G_0 (low number of packets and bytes sent during short periods of time), which includes most of the source clusters ($95\% \times 95\% \times 95\% \approx 85\%$ of them if the three features are independent).

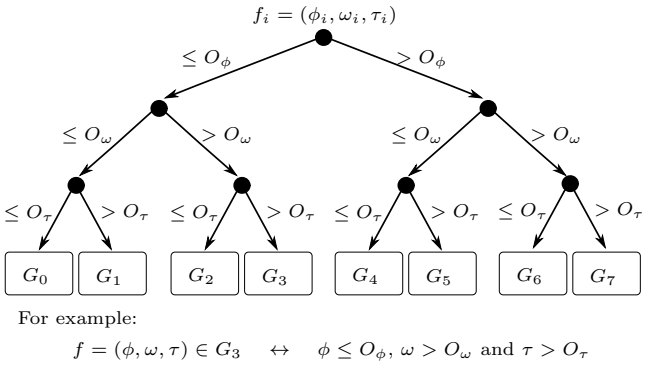

Figure 2: Group description

STONE detects an anomaly if the difference between the reference frequency of each group and the current one is higher than a threshold parameter. The number of source clusters belonging to each group in the current traffic are referred as $\{\widehat{n}_0, \ldots, \widehat{n}_7\}$ while the reference number of source clusters are referred as $\{n_0, \ldots, n_7\}$. *STONE* uses these counters to compute the current ratio per group $\widehat{r}_i = \widehat{n}_i / \sum_j \widehat{n}_j$ and the reference ratio $r_i = n_i / \sum_j n_j$. A traffic anomaly is detected if the distance between reference and current values exceeds a threshold tol, i.e., $\max_i |r_i - \widehat{r}_i| \geq$ tol.

The information maintained at the HD is used to compute the reference point O and the reference ratios $r_0, \ldots, r_7$. These values are based on the traffic features observed over D intervals (e.g., Wednesday, 10:00:00-11:00:00) in the past. The reference ratio r_i is computed as $\sum_d n_i^{(d)} / \sum_{j,d} n_j^{(d)}$, where $n_i^{(d)}$ is the counter of group i at interval d. The reference point O is computed as the weighted 0.95-quantile of each feature of the source clusters communicating with the protected entity during the previous D intervals. Source clusters contribution to O is proportional to the number of times they appear during the reference period. For this purpose, we weight the contribution of each $srcCL_i$ in O defining w_i as the number of intervals in which it appears divided by the total number of intervals (which is D). The

features of $srcCL_i$, referred as $f_i = (\phi_i, \omega_i, \tau_i)$, are computed as the average of the features observed over the last D intervals. For all $srcCL_i$ with weight w_i and average features f_i, the weighted 0.95-quantile of feature ϕ (resp. ω and τ) is computed by ordering all the source clusters average feature $\{\phi_i\}_i$ (resp. $\{\omega_i\}_i$ and $\{\tau_i\}_i$) and selecting the one with weighted position $0.95 \sum_i w_i$. While monitoring the traffic of a protected entity, an anomaly is detected each time the distribution of the source clusters to groups changes abruptly; for instance, when a flooding-packet attack introduces a large number of new source clusters.

3.2 Mitigation Center (MC)

MC is responsible for mitigating an attack to the protected entity filtering malicious traffic while minimizing the degradation of the legitimate one. Figure 3 shows the sequence of steps taken to decide which packets to forward to the protected entity and which to discard. If the maximum load of the protected entity has not been exceeded and mitigation is not activated, packets are forwarded. If the traffic is being mitigated, the MC must ensure the maximum load L is not exceeded while prioritizing legitimate traffic. Filtering of traffic belonging to group G_0 (i.e., mice flows) prioritizes source clusters that were communicating with the protected entity just before detecting the threat. Due to the large number of source clusters that might belong to G_0, its filtering relies on a Bloom Filter [2]. On the other hand, filtering of traffic belonging to groups $G_1, \ldots, G_7$ (i.e., elephant flows) prioritizes source clusters that communicate frequently with the protected entity. In this case, filtering is applied on a per-source cluster basis, depending on the information maintained in the Acquaintance-List (AL). Besides, packets forwarding decisions are taken partitioning the load into channels (one for each group), so that the proportion of load consumed by each group resembles the one observed processing only legitimate traffic. Packets that refer to unknown source clusters (i.e., traffic that could be either from the attack or a flash crowd) are forwarded using a probabilistic function that depends on the available capacity.

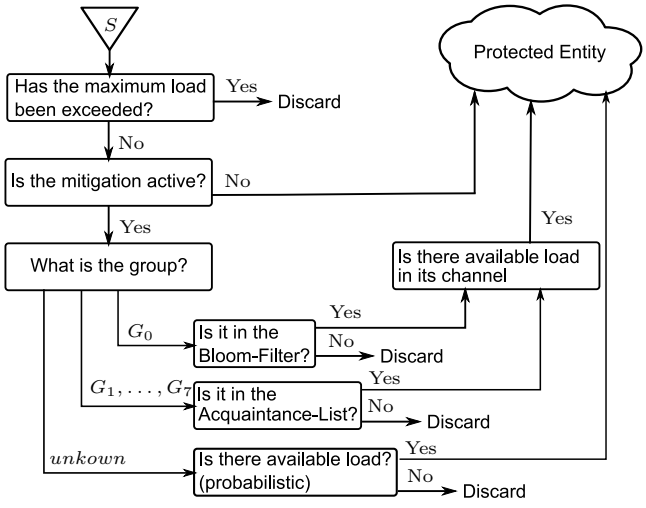

Figure 3: Mitigation Center

The Bloom-Filter (BF), used to filter packets sent by sources cluster belonging to group G_0, is a space-efficient probabilistic data structure used to check the membership of elements in a set [2]. While BFs allow for new elements to be added to the set, removal of elements is not trivial. In *STONE* we improve the base BF defining a *time-based BF* where elements belonging to the set are associated to timestamps that specify when they have been added (*time-based BF* have the same time and space complexity of the base one [2]). Doing this, *STONE* is able to maintain the source clusters belonging to group G_0 over periods of time (over the last 5 minutes in our mitigation mechanism).

The Acquaintance-List AL is used to maintain the source clusters that belong to groups $G_1, \ldots, G_7$. Each source cluster is paired with a probability that specifies the proportion of the packets coming from each source cluster should be forwarded while mitigating an attack. Packets belonging to source clusters appearing to be frequently communicating with the protected entity will have a higher probability to be forwarded than the ones sent by source clusters that connect sporadically.

It should be noticed that, either using the BF or the AL, we are still not guaranteeing that a single source cluster (or a group of them) whose traffic is being forwarded to the protected entity is unable to exceed its maximum load L. Although the attacker has no knowledge about the traffic of the protected entity, if a set of the source addresses overlaps with the ones maintained by the BF and the AL, we need a mechanism to ensure that the traffic of these sources will not saturate the capacity of the protected entity. For this reason, we assign to each group a fraction of the maximum load proportional with the one that each group usually injects.

3.3 Parallel Data Stream Computation

In this section, we present how *STONE* has been implemented by means of a data streaming *continuous query* and how its execution is parallelized by the StreamCloud SPE.

As shown in Figure 4, *STONE*'s query is defined by 7 operators: OP_1, ..., OP_6 compose the DCC while OP_7 composes the MC. For each operator input stream, Figure 4 shows its tuples schema. In the following, we provide a short description of each operator. Operator OP_1 is used to compute the source cluster to which each tuple of S_a refers to. For each tuple t forwarded by OP_1, OP_2 outputs a tuple containing the features of the source cluster maintained by OP_2 before processing t (f_{old}) and after updating them (f_{new}). Group G_{old}, representing the group to which the source cluster belongs based on f_{old}, and group G_{new}, representing the one to which the source cluster belongs based on f_{new}, are computed by OP_3 and, if $G_{old} \neq G_{new}$, a tuple is sent to OP_4 to update groups counters n_{old} and n_{new}. If G_{old} = NULL (i.e., the source cluster appears in G_{new}), only counter n_{new} is updated. Similarly, If G_{new} = NULL (i.e., the source cluster disappears from G_{old}), only counter n_{old} is updated. Group counters are forwarded from OP_4 to OP_5 in order to compute observed ratios $\{\widehat{r}_0, \ldots, \widehat{r}_7\}$, which are subsequently forwarded to OP_6 in order to be compared with the reference ones. Mitigation is invoked by OP_6 each time $\max_i |r_i - \widehat{r}_i| >$ tol. When activated, OP_7 is used to shape the protected entity traffic maximizing the forwarding of legitimate packets.

We discuss now how the query execution is parallelized by StreamCloud, where each operator can be executed at an arbitrary number of *STONE* machines. On one hand, stateful operator OP_2 (maintaining a *window* of tuples) is

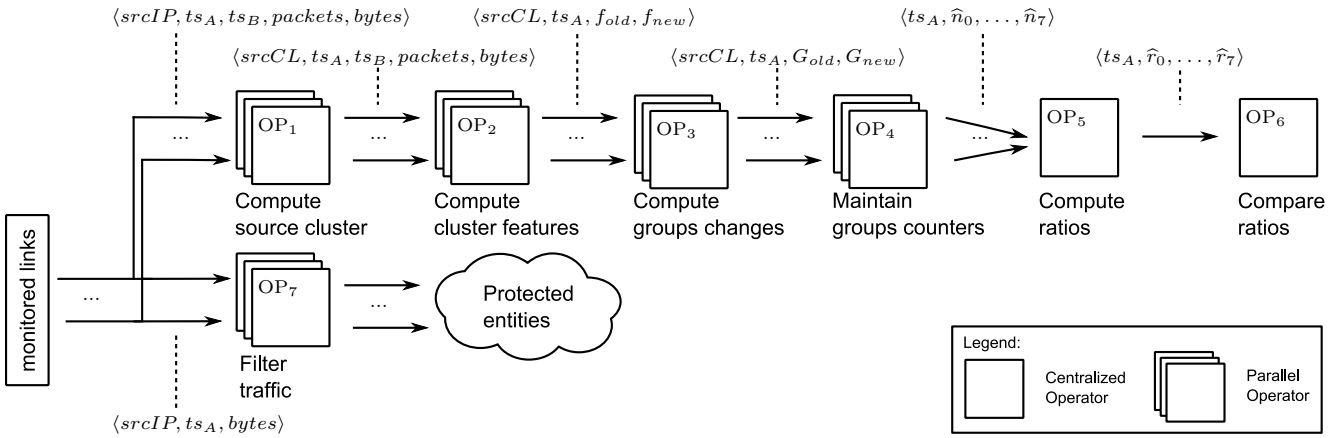

Figure 4: Implementation as a continuous query

parallelized using semantic-aware routing (i.e., tuples referring to the same source cluster are forwarded to the same machine). On the other hand, stateless operators OP_1, OP_3, OP_4 and OP_7 that do not maintain windows are parallelized using round-robin routing. In order to compute the current ratios, all the information maintained by OP_4 nodes must be processed together; for this reason, operator OP_5, and subsequently OP_6, are centralized.

4. EVALUATION

In this section we provide an evaluation of *STONE* detection and mitigation capabilities. As aforementioned, an effective DDoS defense mechanism must detect attacks quickly and mitigate them minimizing the filtering impact on the legitimate users. For this reason, we evaluate three metrics, namely, (1) detection time, the time elapsed between the attack start and the attack detection, (2) mitigation precision, the quantification of the degradation of the legitimate user traffic and (3) traffic volume shaping, the quantification of how much traffic is discarded during the attack.

Evaluation setup. The legitimate traffic is derived from real anonymized data traces from an OC-192 (10Gbits/s) backbone link of OptoSUNET [16]. The data traces are excerpts of the traffic happening on Thursdays (during 9 weeks in 2010) during the period 11:00-12:00. Tuples belonging to S_a are generated for intervals of 5 minutes. Among the destination hosts that appeared in the data traces, we selected the one with higher traffic as the protected entity. The `HD` is populated using the features of the traffic in the first 8 days, while the last day trace is used as *legitimate data trace*. The *attack data trace* is taken from CAIDA [7] and contains approximately one hour of anonymized packets from a DDoS attack. The *attack data trace* has been mixed with the *legitimate data trace* simulating an attack starting after 20 minutes (1200 sec). In our evaluation we use a set of nodes equipped with a quad-core Xeon X3220@2.40GHz, 8GB of RAM and 1Gbit Ethernet.

Detection time. A fast detection of an incoming DDoS attack is crucial for the mitigation effectiveness. Figure 5 shows the maximum difference between the reference and current ratio ($\max_i |r_i - \widehat{r}_i|$) for any group i during the period of time when the attack is taking place. This difference increases rapidly when the attack starts. An anomaly is detected if this maximum exceeds the tolerance. For instance, given a tolerance `tol` $= 0.05$, *STONE* detects the attack after 18 seconds.

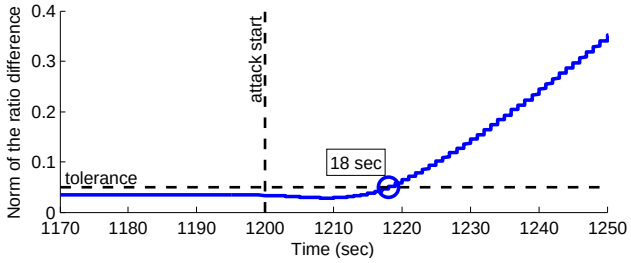

Figure 5: Detection time

Mitigation precision. This experiment studies how effective the BF and the AL are in preventing the forwarding of illegitimate traffic to the protected entity. In order to do this, upon detection of the injected attack (happening at second 1218 in the experiment), the system discards all the packets that do not belong to the BF or the AL. We first measure the precision with which the `MC` component forwards legitimate traffic and discards illegitimate one. Figure 6 presents the percentage of legitimate traffic with respect to the overall legitimate traffic and the percentage of illegitimate traffic with respect to the overall illegitimate traffic forwarded to the protected entity. It can be noticed that, during the attack, the percentage of legitimate traffic that is forwarded is around 90%. The illegitimate traffic is forwarded entirely to the protected entity before the attack detection, but once the mitigation is activated more than 99% is discarded.

Traffic volume shaping. Together with the mitigation precision evaluation, we now evaluate how effective the BF and the AL are when shaping the victim traffic volume (Figure 7). The load is expressed in KBit/second using a logarithmic scale for the y axis. The solid line represents the overall traffic load injected during the attack while the dashed line represents the output traffic forwarded by the

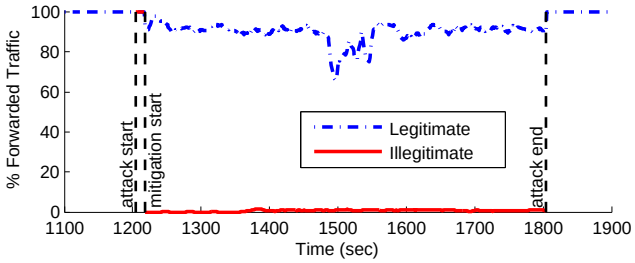

Figure 6: Mitigation precision

MC to the protected entity. While the mitigation is not active, all the input traffic is forwarded to the protected entity. Once the attack is detected, most of the traffic ($\approx 97\%$) is discarded by the BF and the AL. That is, the BF and the AL together are able to reduce drastically the amount of suspicious traffic, leading to an effective mitigation in the presence of legitimate load peaks (e.g., flash crowds).

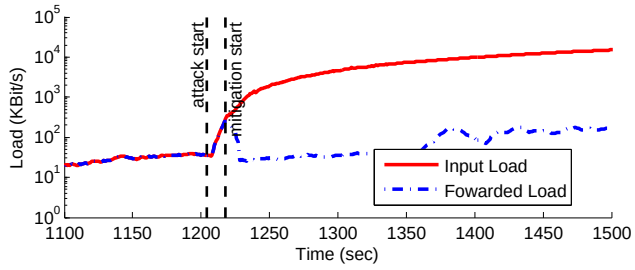

Figure 7: Forwarded traffic load

5. CONCLUSIONS

In this paper we have proposed *STONE*, an anomaly detection DDoS defense framework that leverages data streaming to attain the real-time requirements of this kind of application. *STONE* detects packet-flooding attacks by maintaining traffic features extracted by means of continuous data streaming queries that are also used to mitigate the attack masking it to the legitimate users. Our mitigation mechanism is also beneficial in the presence of flash crowds as it can be used to prioritize usual clients while trying to distribute the remaining capacity among all the clients. The system has been implemented on top of StreamCloud, an elastic parallel-distributed SPE. Our evaluation shows *STONE*'s effectiveness both in terms of detection and mitigation.

6. REFERENCES

[1] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *ACM IMW'02*, pages 71–82, 2002.

[2] A. Z. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2003.

[3] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *CSUR*, 41(3), 2009.

[4] Z. Fu, M. Papatriantafilou, and P. Tsigas. Club: a cluster based framework for mitigating distributed denial of service attacks. In *ACM SAC'11*, pages 520–527, 2011.

[5] V. Gulisano, R. Jiménez-Peris, M. Patiño-Martínez, C. Soriente, and P. Valduriez. Streamcloud: An elastic and scalable data streaming system. *IEEE TPDS'12*, 2012.

[6] V. Gulisano, R. Jiménez-Peris, M. Patiño-Martínez, and P. Valduriez. Streamcloud: A large scale data streaming system. In *IEEE ICDCS '10*, pages 126–137, 2010.

[7] P. Hick, E. Aben, and K. Claffy. The CAIDA "DDoS Attack 2007" Dataset. http://www.caida.org, 2012.

[8] M.-S. Kim, Y. J. Won, and J. W. Hong. Characteristic analysis of internet traffic from the perspective of flows. *Comp. Comm.*, 29(10):1639–1652, 2006.

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

[10] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: methods, evaluation, and applications. In *ACM IMC'03*, pages 234–247, 2003.

[11] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *ACM SIGCOMM'05*, pages 217–228, 2005.

[12] F. Moradi, M. Almgren, W. John, T. Olovsson, and T. Philippas. On collection of large-scale multi-purpose datasets on internet backbone links. In *BADGERS '11*, pages 62–69, 2011.

[13] M. Roesch. Snort, http://www.snort.org/, 2012.

[14] M. Roughan, T. Griffin, Z. M. Mao, A. G. Greenberg, and B. Freeman. Combining routing and traffic data for detection of IP forwarding anomalies. In *ACM SIGMETRICS '04*, pages 416–417, 2004.

[15] F. Silveira, C. Diot, N. Taft, and R. Govindan. Astute: detecting a different class of traffic anomalies. In *ACM SIGCOMM'10*, pages 267–278, 2010.

[16] The Swedish University Computer Network OptoSUNET. SUNET. http://basun.sunet.se/engelska.html, 2012.

7. ACKNOWLEDGEMENTS

This research has been partially funded by the Madrid Regional Council (CAM), FSE and FEDER under project CLOUDS (S2009TIC-1692), the Spanish Research Agency MICINN under project CloudStorm (TIN2010-19077), the European Commission under project MASSIF (FP7-257475), Swedish Civil Contingencies Agency (MSB) and European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No.257007 SysSec. Ricardo Jiménez-Peris and Marta Patiño-Martínez filed a patent at USPTO with number US13/112,628 related to this paper. We thank Farnaz Moradi and Wolfgang John for their support with the data traces collection.