

PISA*: A System for Control of DDoS Attacks

Parminder Chhabra
Boston University
Boston, MA, USA
pchhabra@cs.bu.edu

Sajal K. Das
The University of Texas
Arlington, TX, USA
das@cse.uta.edu

Ajita John
Avaya Labs
New Jersey, USA
ajita@avaya.com

Wei Zhang
The University of Texas
Arlington, TX, USA
wzhang@cse.uta.edu

Abstract— DDoS attacks can cause extreme performance degradation at network elements when a large number of malicious flows collaborate to cause congestion, resulting in a denial of service to legitimate users. The flows form a logical aggregate which is typically characterized by similar values in several fields in their packets. The fields and their similar values form a *signature* for the attack. The focus of this paper is on providing protection to legitimate users against such attacks by detecting significant signatures in network traffic and controlling aggregates of flows carrying these signatures. The paper proposes a system PISA* for deployment at a network element. The detection algorithm in PISA* is based on an improved version of our earlier randomized algorithm. A new control mechanism based on a drop probability function over an attribute named RED Drop Aggregate (RDA) is incorporated in PISA*. RDA is the normalized count of the number of RED (Random Early Detection) drops suffered by the flows carrying a signature. This paper discusses results from an implementation of PISA* that demonstrates that it is effective in detecting, isolating, and controlling offending traffic and providing protection to non-offending traffic.

Keywords: security attacks; traffic signatures; aggregation; RED; malicious sources

I. INTRODUCTION

As Distributed Denial of Service (DDoS) attacks increase in sophistication, frequency, and intensity, the time lag between the onset of the attacks and the manual control of attack traffic becomes very costly, as demonstrated by all recent attacks[2][3]. In a typical DDoS attack, intermediate links, routers and end-systems have to deal with an enormous amount of traffic from infected machines. It is not necessary for each flow to carry a large volume of traffic. The aggregated effect of many flows each carrying a small amount of traffic may be enough to cause an attack. The set of flows participating in an attack forms a logical aggregate characterized by high bandwidth consumption [5][6]. The sources of these flows are typically compromised in a similar fashion by exploiting vulnerabilities in protocols or implementations. Due to the similar nature of the compromise, flows in an attack share common characteristics as manifested in the train of packets in the flows. For example, in a SYN attack, the values for packet size, protocol, the SYN flag in the TCP header, destination address (if a particular server is being targeted), and destination port (if a particular kind of server is being targeted) may be the same for all flows participating in the attack. The common values in the Code Red worm attack were in the following fields: $\{packet\ size, protocol, destination\ port\}$ [3]. A recent attack Novarg / MyDoom [2][3] that spread by email can be represented at a gateway

by $\{(protocol=TCP)(destination\ port=80)\ (destination\ ip=www.sco.com)\ (packet\ size=48)\}$. Typically, unsolicited response traffic has similar values for packet size and error type / code in ICMP packets and TCP flags in TCP packets. Some attacks launched by commonly used attack tools [4] have the same header checksum and associated IP options. Additionally, similarity in application-level fields such as Subject, Body, and Attachment filenames can be found in SMTP-based email viruses. The similarity in values in some fields of a set of flows is a pattern that is referred to as a *signature* [5]. Detection of signatures in network traffic has received a lot of recent attention [5][6][14]. While all signatures may not be indicative of attacks, persistent signatures that are consistently seen across several samples of traffic may be candidates for automated control.

A Our Work and Contribution

This paper describes an architecture for a complete system PISA* for the *detection and control* of DDoS attacks and may be deployed at a network element such as a router. The detection scheme is a randomized algorithm that provides a significant improvement in complexity over earlier work by the authors [5]. The improvement enables the processing of packets in real-time. It generates signatures from any subset of fields in packets and detects significant signatures based on defined properties on signatures. No pre-computed signatures are required. The paper proposes a new control mechanism that rate-controls flows carrying significant signatures based on a bandwidth-based attribute named RED Drop Aggregate (RDA) for a signature. RDA is defined as the normalized sum of RED drops [7] for all flows carrying a signature. RDA represents the combined bandwidth consumption of a group of flows and, thereby provides a measure for the “malicious behavior” of aggregates of flows in flooding attacks. Aggregates of flows are controlled according to a drop probability function of their RDA. The goal of the work is not to block flows that belong to aggregates. Instead, it attempts to proportionately reduce the bandwidth consumption of all aggregates to a percentage of the capacity of the shared link at which the system is deployed. Per-flow state is not maintained and the per-packet computation in the fast path is small.

We present experimental results from an implementation of PISA* as a dynamically loadable kernel module in the Linux 2.6 kernel [16]. Results show that (1) PISA* quickly identifies signatures of flows that share similar values for several packet fields. (2) RDA serves as an effective measure in controlling high-bandwidth aggregates, which may contain individual low rate flows. (3) Non-offending low rate flows and real-time applications such as voice are not penalized by this scheme. (4) The overhead introduced at the router by the kernel module is small.

The key contribution of this work is the design and realization of a system for the real-time detection of candidates for unknown DDoS attacks and the coupled control mechanism that can quickly control the candidates to provide protection to non-offending flows.

The paper is organized as follows: Section II describes related work. Section III provides background on signatures and their properties. Sections IV, V and VI present PISA*, experimental results and conclusions, respectively.

II. RELATED WORK

Recent algorithmic work has focused on automatic detection of signatures that represent profiles of aggregates [6][5]. This allowed for extraction of signatures of attacks [14][8][5]. Statistical techniques as in [9] and [15] have been used to extract features of attacks and subject such flows to control as in [15] via pushback [11]. RED-PD (RED [7] with Preferential Dropping) [10] and ACC (Aggregate Congestion Control) [11] are mechanisms to identify and control individual high bandwidth flows and high bandwidth aggregates, respectively. However, both RED-PD and ACC do not address attacks of the type where a number of low bandwidth flows collaborate to effect a state of cumulative congestion at a router.

PISA* addresses DDoS flooding attacks where the signature may be formed by any subset of fields in the packets. Additionally, PISA* addresses attacks where cumulative rather than the individual effect of flows dominate. PISA* uses a cumulative RED drop measure, that is easy to compute, to *control* flows that are part of an aggregate. By using a cumulative drop measure, PISA* can target various types of flooding attacks including the case where a number of low-bandwidth flows collaborate to cause congestion.

III. SIGNATURES IN NETWORK TRAFFIC

This section summarizes key concepts and definitions that were presented in [5] and forms a background for our work.

A signature is defined as k ordered pairs $\{(f_1, v_1), (f_2, v_2), \dots, (f_k, v_k)\}$, where $k \geq 1$ is the dimension of the signature, f_i is a field in a packet and v_i is a value for f_i , $1 \leq i \leq k$. The notion of a flow used in this work is a 5-tuple with the following fields: (source address, source port, destination address, destination port, protocol). Four properties that characterize signatures are as follows: (1) *Dimension* is the number of field-value pairs in a signature. (2) *Intensity* is the average number of packets/bytes that carry the signature. (3) *Persistence* is a signature's activity over time. (4) *Distribution* is the average number of flows (or, prefix matches of IP addresses across flows) carrying the signature and is useful in determining the spread of the signature.

The *significance* of a signature can be defined in terms of its properties. For example, a system may define significant signatures to be those that contain at least 4 fields (dimension), have a bandwidth consumption of at least 2% (intensity), appear for at least 60 seconds (persistence), and be carried by 5% of the flows (distribution).

An m -dimensional cluster C_m is defined as a set of flows in which all flows have similar values for each field in a set of m fields. The m fields form a signature for the flows in the cluster. The flows in a network sample can be grouped into

clusters of flows such that all flows in a cluster carry a "maximal" signature, where the maximal signature for a set of flows is the largest set of similar field-value pairs. Each flow in a cluster has a weight given by the number of packets/bytes in that flow. The weight of a cluster is defined as the sum of the weights of all the flows in it. The weight of a cluster is an indicator of its bandwidth consumption. An *intensive* cluster is one whose weight is above a specified threshold. *Intensive clusters yield intensive signatures.*

IV. PISA*: DEFENSE AGAINST DDoS ATTACKS

Figure 1 shows the architecture for PISA*. There are two main data structures in the architecture: a flow table and a signature table. There are two main routines: the signature detection routine and the control routine.

The flow table contains samples or *segments* of incoming traffic. The packets in a segment are grouped according to the flow that they belong to. Each entry in a segment of the flow table corresponds to a flow. It contains the fields in the flow that will be considered for signature detection and the number of packets that have been sampled for that flow. The signature detection routine operates on the segments. Each entry in the signature table contains a signature that has been generated by the signature detection routine, its RDA, and a Time-To-Live field that allows the removal of signatures that have not been seen for a long time. The signature detection routine executes outside the fast path of the incoming packet.

An incoming packet may be added to the flow table depending on the sampling and is subjected to a RED drop. The signature table acts as a post-filter to a RED gateway. If the flow for an incoming packet carries a signature that is in the table, the packet is dropped with probability that is computed based on the RDA of the signature.

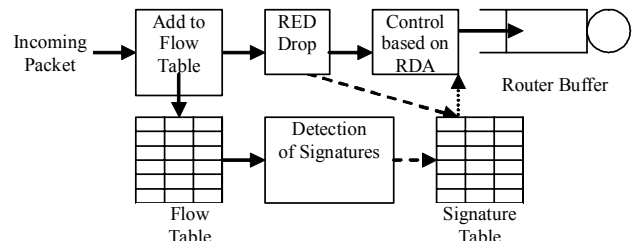


Figure 1: Architecture of PISA*

A Detection of Signatures in PISA*

Given a set of fields that may be defined in packets (well-defined header fields such as IP, TCP, UDP, ICMP etc. or content fields or the name of an attachment, subject heading in an email etc.), the signature detection routine in PISA* finds significant signatures that share similar values for any subset of fields across samples of network traffic. The authors presented a randomized algorithm named PISA in [5] that generates such signatures. In this section, we present an improved algorithm with reduced complexity. Comparisons of complexity between the two versions are presented in IV.C.

Consider a lattice of all possible subsets of F and referred to as the flow lattice in this paper. The empty set is at the bottom of the lattice and F is at the top of the lattice. An edge exists between two subsets S_1 and S_2 if S_1 is a superset of S_2 and $|S_1| = |S_2| + 1$. The lattice consists of $|F|+1$ levels where

each level i has all the subsets of F with cardinality i (See **Figure 2**). The intensive clusters of F are points on the lattice.

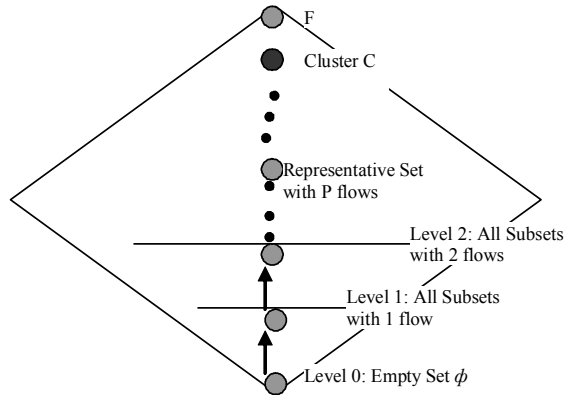


Figure 2: Signature Detection in PISA*

Let h be the number of fields of interest in the packets and k ($1 \leq k \leq h$) be the minimum number of fields desired to be similar in a cluster. The goal is to find intensive signatures of dimension greater than or equal to k . [5] discusses how the dimension of a cluster (and the dimension of its associated signature) monotonically decreases up the lattice. This implies that for any $1 \leq k \leq h$, either F is a k -dimensional cluster or there must exist two subsets of F , S_i, S_{i+1} , such that S_i is an i -dimensional cluster, S_{i+1} is a j -dimensional cluster, there is an edge between S_i and S_{i+1} , and $j < k \leq i$. S_i is called a *representative set*. Along the path from the bottom of the lattice to S_i , there may be several representative sets of dimensions greater than k . The number of paths to a representative set is exponential in the number of flows in the representative set. A representative set is so named because it is representative of a cluster C in the lattice.

The signature extraction phase tries to find a path to a representative set through a *random walk* where random selections of flows are made to move up the levels of the lattice. Starting at the bottom of the lattice by initializing a set S as the empty set, it traverses up in a sequence of steps. At each step, it makes a random selection of a flow from the flow table to add to the set S , each time checking if the resulting set is a representative set of dimension k or greater. It collects all representative sets and stops the traversal when the resultant set is a cluster of dimension less than k .

In the algorithm proposed in [5], each representative set was expanded against all the flows in the set F to construct the corresponding cluster C . If C contained a certain threshold of packets/bytes, the cluster was marked as an intensive cluster. In PISA*, we eliminate the expansion phase and test if the representative set contains a (scaled) threshold of packets/bytes. Instead of finding the intensive clusters, we find *intensive representative sets*. This provides a significant improvement to the complexity (and hence, execution time) of the algorithm and results demonstrate that this method is very effective in detecting significant signatures in real-time.

The signature extraction phase proceeds through a number of random walks where intensive representative sets are detected. The number of random walks can be controlled by making the algorithm abort when no (or few) new intensive

representative sets are detected in a number of consecutive random walks. The signature extraction phase updates the signature table with the signatures of the intensive representative sets.

B RDA-based Control of Flows Carrying Signatures

This subsection discusses the control of flows carrying signatures based on their bandwidth consumption. We estimate the bandwidth consumption of a signature by the RED drop count suffered by the flows carrying the signature. We define an attribute named RED Drop Aggregate (RDA) for a signature across a time interval as the normalized sum of the RED drops suffered in that interval by the flows carrying the signature. For any signature A ,

$$RDA(A) = K * \frac{\sum_{f_i \in A} \text{Red Drops for } f_i}{\sum_{A_i} \text{Red Drops } (A_i)} \quad (a)$$

where K is a parameter to be configured, f_i is a flow that carries signature A and $Red\ drops(A_i)$ is the sum of the RED drops suffered by all the flows carrying A_i . Thus the RDA of a signature is the sum of the RED drops of all the flows carrying that signature divided by the sum of the red drops across all signatures times a constant K . Hence, $0 \leq RDA(A) \leq K$. K helps to fix the amount of bandwidth that is allowed to all signatures at a network element. This bandwidth is shared proportionately across all signatures. The sum of the RED drops across all flows is chosen for computing the RDA because it is the cumulative effect of the flows that causes congestion. Using an average value would not provide a good measure for an attack where a large number of low bandwidth flows collaborate to increase congestion.

Each entry in the Signature Table corresponds to a unique signature and stores the red drop count suffered by the flows carrying the signature. This count is normalized using the formula specified in (a) to compute RDA. It is computed across intervals of time to soften the effect of bursts in traffic.

Each signature in the signature table is associated with a TTL. Short TTL values will favor misidentified signatures being swapped out of the table faster while long TTL values will favor high values of red drop counts to build up. Each RED drop for a flow in a signature adds a TTL value to the remaining TTL value in the entry corresponding to the signature. This implies that the “effect” of a RED drop lasts for one TTL. Signatures whose TTL have expired are flushed from the table after they are logged. The TTL allows only persistent signatures to affect the control strategy.

For any incoming packet, if the average queue size is between the minimum and maximum thresholds defined for the router buffer, the arriving packet is marked for drop with a probability computed by RED (RED drop). It is then checked against the signature table. For an efficient lookup, the table may be implemented as an h -level hashtable where h is the maximum number of fields of interest in a signature and the i^{th} level in the hashtable corresponds to the lookup for the i^{th} field in the signature. If the flow corresponding to the packet does not carry any signature in the table, then the packet is treated just as in the RED algorithm [7] – it will be dropped if it was marked for a RED drop, otherwise it enters the buffer queue. If the flow corresponding to the packet carries one or more signatures in the table (table hits), it has to be controlled depending on the RDAs of the signatures and the red drop count of the matching signatures may have to be updated.

There are two possibilities for such a packet: (1) The packet was marked for drop by RED; (2) The packet was not marked for drop by RED. In (1), the red drop counts of all matching signatures are incremented by 1 (and the packet is dropped). In (2), the packet is marked for drop (table drop) with a probability based on the maximum red drop count across all matching signatures. To soften the effect on flows carrying misidentified signatures, an initially slow growing function for the table drop probability p is chosen as follows:

$$p = \min\{1, P_{RED} * 2^a / (1 + 2^a)\}$$

where P_{RED} is the RED drop probability at the RED gateway and “ a ” is the RDA given by formula (a).

The parameter K in (a) determines the percentage of the link capacity allocated for the signatures, which is proportionately shared among all the signatures. High values of K will effect greater control on the signatures and provide more bandwidth for legitimate users and vice versa. K can be computed to allocate a fraction x (say 0.1) of the link capacity to the signatures by solving the following equation for K :

$$x = P_{RED} * 2^K / (1 + 2^K)$$

P_{RED} can be fixed by fixing the average queue size.

Any controlling scheme that uses the RDA attribute will penalize a flow based on its membership in an aggregate rather than its individual behavior. The cost of such a scheme is that a heterogeneous aggregate with a mix of low and high bandwidth flows that is misidentified as being malicious may overly control the low bandwidth flows. However, given that that flows are not blocked out completely, this may be a small price to pay to protect most legitimate traffic.

C Complexity of PISA*

The space complexity of PISA* is determined by the sizes of the flow and signature tables. As long as the signature detection routine consumes the segments in the flow table at least as fast as the sampling routine creates the segments, the flow table size can be bounded. In our experiments with the implementation of PISA*, this goal was achieved because of the improved detection algorithm. The TTL field for the signatures helps to control the size of the signature table.

The fast path computation in PISA* is limited to (1) an h -level hash function lookup in the signature table for each packet. Since h is a constant and typically small, this step takes a constant amount of time. (2) Finding the maximum of the RDA values of the signatures that result from the lookup: The complexity of this step is proportional to the number of signatures that match the packet. (3) The computation of p if there is a table hit for that packet.

The non-fast path computation of the signature detection routine is dominated by the random walks. A key observation is that the more the number flows that carry a signature, the greater the chance of it being detected. Additionally, most random walks will end unsuccessfully at lower levels of the lattice and longer random walks are indicative of signatures that are distributed across many flows and will be detected early. While the maximum length of a random walk is the number of flows, on an average it will be much smaller.

In the algorithm proposed in [5], the complexity was dominated by the expansion phase of the algorithm where each representative set was expanded against all the flows and the complexity of each random walk was $O(f)$ where f is the

number of flows in the lattice. Hence, PISA* provides a significant improvement in performance.

V. RESULTS

The results presented in this section focus on (1) the effectiveness of our control strategy based on RDA and (2) the effectiveness of a PISA* implementation in a real network deployment to quickly detect and control attacks in a mix of attack and non-attack traffic.

A Experiment Results using a Simulation Environment

This section demonstrates that our control mechanism can:

(a) Protect TCP flows as the arrival rate of attack flows varies as a percentage of bottleneck link bandwidth. (b) Provide a configurable level of bandwidth to control rather than completely block them out. (c) Protect adaptive TCP flows under a mix of aggregates containing both high and low bandwidth flows.

In this experimental setup, a number of flows (UDP/TCP) are grouped together on the basis of a signature (using a predefined set of field-value pairs over: protocol type, destination port and NS-based flow-id). Each aggregate has varying UDP rates to study the intensity of flooding attacks

All simulations were run using Network Simulator (NS2) [12]. As in [7], we used a dumbbell network topology. A fixed number of TCP and UDP sources send packets to sinks through two routers connected through a 45 Mbps bottleneck link with a 1 ms delay. The bandwidth of the links from the TCP sources was 100 Mbps.

The maximum window sizes of the 576 byte TCP NewReno sources were large enough so as not to be constrained during TCP data transfer. The total simulation time was 200 seconds of which, the initial and final 10 seconds were ignored. Results were averaged over several runs. The setup was as follows: 200 TCP sources with RTT = 42ms. Sending rate of 70 UDP sources with an end-to-end delay of 21 ms varied between a low-rate of 0.7% and a high-rate of 3% of the bottleneck link. We model three different kinds of aggregates in this setup: (a) Aggregate 1: 35 low-rate UDP flows. (b) Aggregate 2: 20 UDP flows with 18 low-rate and 2 high-rate. (c) Aggregate 3: 5 high-rate UDP flows.

One router serves as the RED gateway with parameters as follows: $\min(\text{threshold}) = 200$, $\max(\text{threshold}) = 600$, maximum buffer size = 900, and the maximum RED packet drop probability = 1/10. The signature table was allowed to grow as needed and a TTL value of 100ms was used.

We compare the control offered by the RED algorithm to the RDA mechanism. Figure 3 shows the throughputs of TCP and aggregates of UDP sources with RED. The three aggregates send at 24.5%, 18.6 and 15% of the bottleneck link and are allowed (with little control) throughputs of 23%, 18.2% and 14.4%, respectively. TCP receives a throughput of 42%. RED is ineffective in exercising control on aggregates.

Figure 4 shows that RDA forms a more effective parameter to control the aggregates. Since the RDA of each aggregate is normalized, each aggregate suffers a drop in throughput proportional to its bandwidth consumption. By varying the parameter K , we can affect varying levels of control on the throughputs of the aggregates. For example, for $K=6$, the throughput of the three aggregates falls to 17.6%, 14.5% and 12% respectively and TCP throughput rises to 50%. For $K=50$, the throughput of the three aggregates receive

14.5%, 11.1% and 8.9%, respectively and TCP throughput rises to 65.5%.

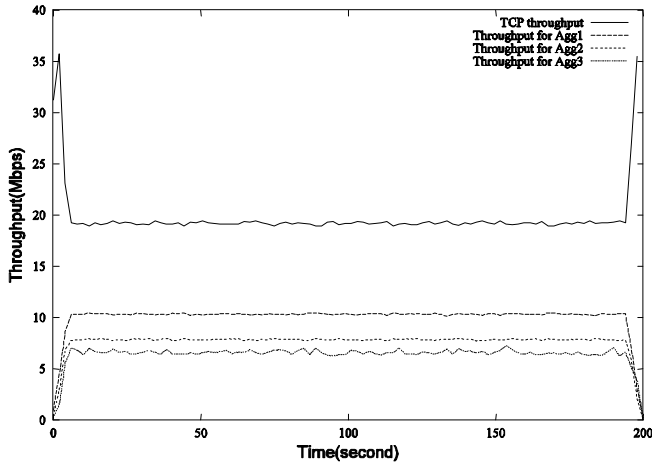


Figure 3: Throughput of TCP and aggregates under RED.

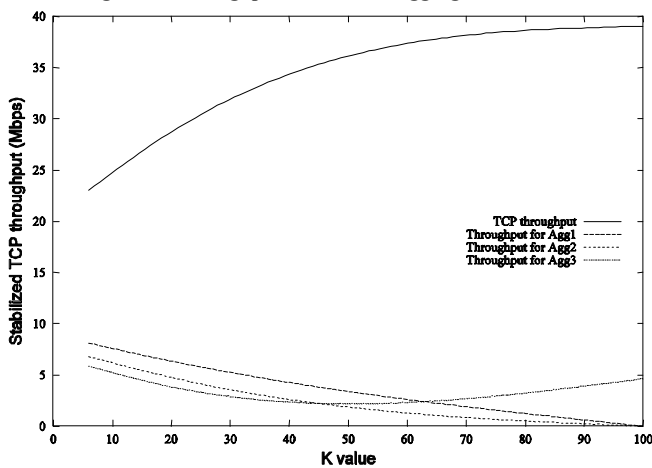


Figure 4: Throughput of TCP and aggregates with varying K

Even though the Aggregate 1 consists of only low bandwidth flows, its combined bandwidth consumption is more than Aggregates 2 and 3. Hence, it suffers a larger proportion of the RED drops and so, is penalized more heavily than Aggregate 3. Once Aggregates 1 and 2 are controlled, the available bandwidth is taken by Aggregate 3 as seen in Figure 4. Aggregate 3 may be controlled for larger values of K, not shown in figure for clarity.

B Experiments with a System implementing PISA*

Having understood how RDA can serve as an effective measure to control aggregates in a simulation environment, we now study its behavior in an actual network deployment of our system implementation. A full system realization of the architecture shown in Figure 1 was done as a loadable module in the Linux Kernel. The module was registered with the Netfilter [16] hook to receive all packets. The fields of interest in every arriving packet were stored in the flow table. This was done on the fast path of every inbound packet. The signature detection routine ran as a kernel thread to generate signatures which were stored in a signature table. This allowed us to match fields of packets against entries in the signature table on the fast path and before any routing decision on the packet was made. We used a dumbbell topology similar to the one shown in [7]. All machines were

connected via 100 Mbps ethernet cards. The physical interfaces on the source and sink machines were configured to have several logical interfaces. Our goals were to determine

- How quickly was the attack signature identified?
- How persistent was the attack signature?
- How effective is RDA for control of flows with signatures?
- The effect on the good sources (TCP / UDP/ voice).

V.B.i Experimental Setup and Results

The following values were used for the parameters in the PISA* system: Segment Size = 1000 packets, Time to Live (TTL) = 6s, Minimum fields in a signature = 4, Threshold intensity = 8% of segment size. Signatures are persistent if were detected in at least 10 out of 100 consecutive segments.

The experimental setup consisted of (i) 30 TCP sources, (ii) 20 low-rate UDP sources (set to consume about 9% of the link) and (iii) a few ICMP sources (iv) voice traffic using Avaya VoIP [1] phones. The attack traffic was modeled to be 50 UDP flows forming an *attack signature* with the following fields - <Protocol, Packet Size, Source Port, Destination Port>.

Netperf 2.4 [13] was used as the generator to generate the above traffic load. We modeled a slow growing attack as shown in Figure 5. TCP and good UDP sources were made active within the first 20 seconds of the start of the experiment. The malicious UDP traffic had a staggered growth, with a new flow becoming active every 3 seconds, starting at the 26th second. Each malicious flow contributes 200 packets per second. The observed incoming packet rate at the router was about 10,000 packets per second.

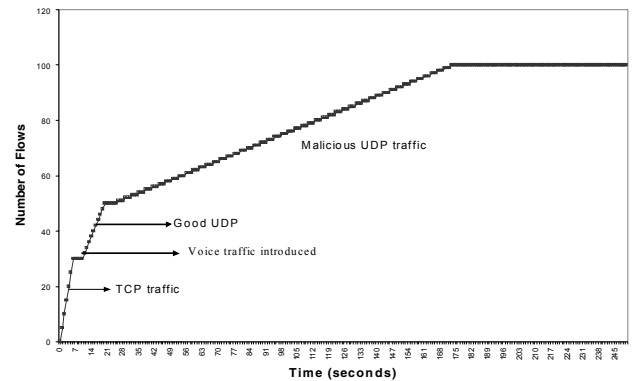


Figure 5: Time duration during which all sources are active. The malicious UDP traffic is slow growing.

Identification of Signatures: The *attack signature* was first identified after 29 malicious flows were active. This signature appeared as a persistent signature when about 40 flows were active.

Persistence of Signatures: Figure 6 shows snapshots of the signatures taken every 20s. We observe that (1) initially, all signatures were TCP signatures. Signatures involving TCP traffic were found to be different and transient. (2) One UDP attack signature is seen as a persistent signature during the course of the attack. (3) As the intensity of the attack increases, the total number of signatures decreases since flows corresponding to transient TCP signatures now contribute to a smaller percentage of the overall traffic.

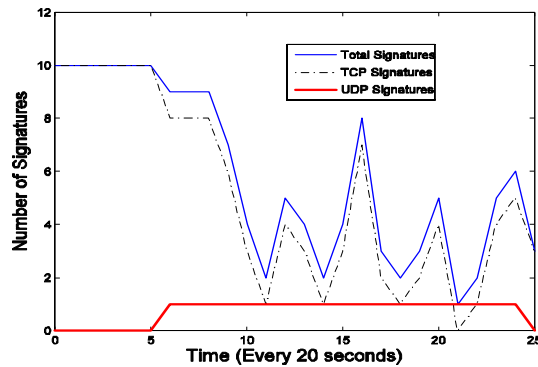


Figure 6: A breakdown of the number of signatures observed into TCP and UDP signatures.

Effectiveness of RDA in control: Figure 7 compares the router's inbound attack traffic against the inbound attack traffic observed at the sink. The traffic arriving at the sink has already been subject to control based on RDA of the signature.

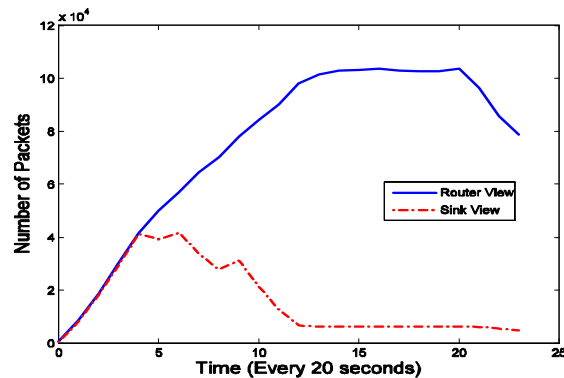


Figure 7: A comparison of router's view of inbound malicious UDP traffic and the sink's view of the same traffic.

Initially, the flows corresponding to the UDP attack are not penalized too severely since they suffer few RED drops. However, as the intensity of the attack increases, the attack signature becomes persistent and its RDA builds up. Since the other observed signatures (TCP) are infrequent, flows that correspond to such signatures are penalized insignificantly.

The strength of the RDA as a control mechanism in the system implementation can be gauged from the fact that over the duration of the experiment (about 300s), only 700 TCP packets were dropped whereas 1.6 million malicious UDP packets were dropped.

Effect on low-rate non-attack UDP and voice traffic: We observed that low-rate sources (UDP and voice) are not affected by the system. As a result, there is very little difference between traffic seen at the inbound interfaces of router and sink.

The signature generation routine of system consumes the segments as fast as the sampling routine creates them. This allows for the use of fixed size flow table. The implementation added a CPU overhead of about 7% at the router with 800 MHz Intel processor running Linux 2.6.

VI. CONCLUSIONS

Recent work on identifying flows participating in DDoS attacks has grouped flows into aggregates based on

commonality in values of packet fields. A signature consists of the set of common field-value pairs in these flows. This paper describes an overall architecture for a system named PISA* that may be deployed at a network element such as a router for detection and control of unknown DDoS attacks. The architecture is based on an efficient randomized algorithm that detects significant signatures based on any subset of fields in packets and subjects them to control based on a composite attribute named RDA. RDA is the normalized RED drop count suffered by the flows carrying a signature. RDA can be easily computed from existing information (RED drops) at a network element. PISA* places no restriction on the number of aggregates that can be controlled. The state information required is small. PISA* allows for the proportional sharing of a percentage of the capacity of a link among signatures that have been identified for control. It enables the network element to perform self-healing in the face of performance degradation under DDoS attacks.

The paper describes simulation results that compare the effect of RED and RDA in controlling multiple aggregates. It shows that RDA provides an enhanced and configurable level of protection to legitimate flows under varying degrees of DDoS attacks. In addition, experiments on a system implementation of PISA* demonstrate the effectiveness of PISA* to quickly isolate offending traffic and provide protection to non-offending traffic.

VII. REFERENCES

- [1] Avaya, At <http://www.avaya.com>
- [2] CERT, "W32/Mydoom." CERT/CC Current Activity, Incident Note IN-2004-01, January 2004.
- [3] CERT: Vulnerabilities, Incidents and Fixes. At http://www.cert.org/nav/index_red.html
- [4] Distributed Denial of Service Attacks/tools: Available at <http://staff.washington.edu/dittrich/misc/ddos/>
- [5] P. Chhabra, A. John, H. Saran, "PISA: Automatic Extraction of Traffic Signatures". *IFIP Networking 2005*, Waterloo, May 2005.
- [6] C. Estan, S. Savage and G. Varghese, Automatically Inferring Patterns of Resource Consumption in Network Traffic, Proceedings of the ACM SIGCOMM Conference, Karlsruhe, Germany, August 2003.
- [7] S. Floyd, and V. Jacobson, "Random Early Detection gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, V.1 N.4, August 1993, pp. 397-413.
- [8] H.-A. Kim, and B. Karp, "Autograph: Toward Automated, Distributed Worm Signature Detection." In *Proceedings of the 13th Usenix Security Symposium*, San Diego, CA, August, 2004.
- [9] A. Lakhina, M. Crovella and C. Diot, "Mining Anomalies Using Traffic Feature Distributions." In *Proceedings of ACM SIGCOMM 2005*, Philadelphia, PA, August 2005.
- [10] R. Mahajan, S. Floyd, D. Wetheral. Controlling High-Bandwidth Flows at Congested Router. Ninth IEEE ICNP, 2001.
- [11] R. Manajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling High Bandwidth Aggregates in the Network." In *CCR*, V.32 N.3, July 2002
- [12] S. McCanne and S.Floyd, "ns - Network Simulator", <http://www.nrg.ee.lbl.gov/ns/>.
- [13] Netperf, Available at <http://www.netperf.org/netperf/NetperfPage.html>
- [14] S. Singh, C. Estan, G. Varghese, and S. Savage, "Automated Worm Fingerprinting." In *Proceedings of the 6th ACM/USENIX OSDI Symposium*, San Francisco, CA, December 2004.
- [15] H. Sun, J. C. S. Lui, D. K. Yau, "Defending Against Low-Rate TCP Attacks: Dynamic Detection and Protection." In *Proceedings of ICNP 2002*, Berlin, Germany, August, 2004.
- [16] The Linux Kernel Archives. Available at <http://www.kernel.org/>