IJCSE International Journal of Computer Sciences and Engineering Open Access

Research Paper **Vol.-6, Issue-7, July 2018** **E-ISSN: 2347-2693**

Controlling Distributed Denial-of-Service Attacks through Dynamic Path Identifiers

N. Vijay[1*], M. Sakthivel[2]

[1*] Department of Computer Science and Engineering, Sree Vidyanikethan Engineering College, Tirupati, Andhra Pradesh, India.

[2] Department of Computer Science and Engineering, Sree Vidyanikethan Engineering College, Tirupati, Andhra Pradesh, India

**Corresponding Author: nagoorvijayit48@gmail.com, Tel.: +91-8374698898*

Available online at: www.ijcseonline.org

Accepted: 03/Jul/2018, Published: 31/Jul/2018

Abstract— A Distributed Denial of Service flooding attack in the network performed implicitly and as well as explicitly by the attacker or victim. This attempt is performed to overload the server, generate malicious traffic or interrupting the service. This issue crashes the host and the host's service will be unavailable to the legitimate users. The enterprise, employment, and assessment of D-PID, a basis that uses PIDs transferred between adjacent domains as inter-domain routing objects. In DPID, the PID of an inter-domain path linking two domains is reserved clandestine and changes animatedly. We label in part how neighbouring domains negotiate PIDs, how to uphold constant communications when PIDs change. We shape a 42-node sample comprised by six domains to prove D-PID's possibility and demeanour widespread admirations to gauge its efficacy and charge.

Keywords—Inter-domain routing, security, Distributed Denail-of-Service (DDoS), path identifiers

I. Introduction

In recent years major threat in computer world is DDoS attacks. The DDoS come from the Denial of service (DoS) attacks; Where DoS is early used by the underground attackers to bring down the services of Organizations. It is very easy to performed this attack because the tools (for example Trinoo) required of this attacks are available in internet for free downloadable and easy to use .Distributed denial of service (DDoS) attack, which makes a server suffer in having slow responses to clients or even refusing their accesses, is one of the major threats which will continue in the future. DDoS attack is targets the important resources like banks and other similar organizations to make the important information to access or publish to everyone or some persons. It works distribute over a wide range of systems to launch this attacks to bring down the website by continuously sending requests to access the resources and make it to denial other requests. Distributed denial-of-service (DDoS) flooding attacks are very harmful to the Internet. In a DDoS attack, the attacker uses widely distributed zombies to send a large amount of traffic to the target system, thus preventing legitimate users from accessing to network resources [1]. For example, a DDoS attack against BBC sites in Jan. 2016 reached 602 gigabits per second and "took them down for at least three hours" [3]. More recently, the hosting provider OVH suffered a large scale DDoS attack in Sep. 2016, launched by a botnet composed at least of 150,000 Internet-of-things (IoT) devices. This attack peaked at nearly one terabit per second (Tbps) and even forced Akamai to stop offering DDoS protection to OVH [2]. Therefore, many approaches [4] have been proposed in order to prevent DDoS flooding attacks, including network ingress filtering [5] - [9], IP trace back [10] - [14], capability-based designs [15] - [18], and shut-up messages [19] - [20]. At the same time, in recent years there are increasing interests in using path identifiers PIDs that identify paths between network entities as inter-domain routing objects, since doing this not only helps addressing the routing scalability and multi-path routing issues [21], but also can facilitate the innovation and adoption of different routing architectures [22]. For instance, Godfrey et al. proposed path let routing [21], in which networks advertise the PIDs of path lets throughout the Internet and a sender in the network constructs its selected path lets into an end-to-end source route. There are two different use cases of PIDs in the aforementioned approaches. In the first case, the PIDs are globally advertised (as in path let routing [21] and [22]). As a result, an end user knows the PID(s) toward any node in the network. Accordingly, attackers can launch DDoS flooding attacks as they do in the current Internet. In the second case, conversely, PIDs are only known by the network and are secret to end users (as in LIPSIN [23]. In the latter case, the network adopts an information-centric approach [25] - [27] where an end user (i.e., a content provider) knows the PID(s) toward a destination (i.e., a content consumer) only when the destination sends a content request message to the end user. After knowing the PID(s), the end user sends packets of the content to the destination by encapsulating the PID(s) into the packet headers. Routers in the network then forward the packets to the destination based on the PIDs. It seems that keeping PIDs secret to end users (as in [23], [24]) makes it difficult for attackers to launch DDoS flooding attacks since they do not know the PIDs in the network. Due to the static nature of the PIDs it is not enough to keeping the PIDs secret form the end users as a preventing step for DDoS flooding attacks. For example, Antikainen et al. argued that an adversary can construct novel filters (i.e., PIDs) based on existing ones and even obtain the link identifiers through reverse-engineering, thus launching DDoS flooding attacks [28]. Moreover, attackers can launch DDoS flooding attacks by learning PIDs if they are static. To overcome this issue the dynamic PID (D-PID) mechanism is design, implementation and evaluation is presented in this research. In D-PID, two adjacent domains periodically update the PIDs between them and install the new PIDs into the data plane for packet forwarding. Even if the attacker obtains the PIDs to its target and sends the malicious packets successfully, these PIDs will become invalid after a certain period and the subsequent attacking packets will be discarded by the network. Moreover, if the attacker tries to obtain the new PIDs and keep a DDoS flooding attack going, it not only significantly increases the attacking cost, but also makes it easy to detect the attack. In particular, our main contributions are twofold.

On one hand, we propose the D-PID design by addressing the following challenges. First, how and how often should PIDs change while respecting local policies of autonomous systems (ASes)? To address this challenge, D-PID let's neighboring domains negotiates the PIDs for their inter-domain paths based on their local policies. It is mandatory for legitimate communications to maintain its communications to prevent illegal communications while changing the PIDs because inter-domain packet forwarding uses PIDs which changes. Overcome this problem, D-PID lets every domain distribute its PIDs to the routers in the domain. Third, the overheads incurred by changing PIDs should be kept as small as possible.(It has Passover overhead for neighboring domains to pass PIDs and also have distributed overhead of updated PIDs for domains. This is experimented with 42-node prototype over six domains to verify and compare D-PIDs feasibility.

As a last paragraph of the introduction should provide organization of the paper/article (Rest of the paper is organized as follows, Section I contains the introduction of DDoS attacks, Section II contain the related work of DDoS flooding attacks, Section III contain the problem statement of Dynamic path identifier, Section IV contain the background work and motivation, section V explain the prototype methodology, Section VI describes results and discussion about D-PID over the DDoS attacks , Section VII explains the conclusion.

II. RELATED WORK

Because of the complexity and difficulty in defending against DDoS flooding attacks, many approaches have been proposed in past two decades. For instance, filtering-based approaches aim at mitigating DDoS flooding attacks by deploying source address filtering at routers [5] - [9]. Similarly, IP trace back-based methods trace attacks back through the network toward the attacking sources [10] - [14]. In addition, the approaches proposed in [19] - [20] aim at mitigating DDoS attacks by sending shut-up messages to the attacking sources, assuming that they will cooperate and stop flooding. While there are too many literatures, we refer interested readers to [4] for a survey on existing approaches in defending again DDoS flooding attacks. Instead, we outline prior work closely related to this work and compare D-PID with them. A main reason that DDoS flooding attacks proliferate is a node can send any amount of data packets to any destination, regardless whether or not the destination wants the packets. To address this issue, several approaches have been proposed. In the "off by default" approach [15], two hosts are not permitted to communicate by default. Instead, an end host explicitly signals, and routers exchange, the IP-prefixes that the end host wants to receive data packets from them by using an IP-level control protocol. The D-PID design is similar in spirit, since D-PID dynamically changes PIDs and a content provider can send data packets to a destination only when the destination explicitly sends out a GET message that is routed (by name) to the content provider. However, there are two important differences. First, the "off by default" approach works at the IP-prefix granularity, but D-PID is based on an information-centric network architecture and works at the content granularity. Second, the IP-prefixes that an end host wants to receive packets from are propagated throughout the Internet in the "off by default" approach, which may cause significant routing dynamics if the allowed IP-prefixes of end hosts change frequently. On the other hand, the PIDs are kept secret and change dynamically in D-PID. While this incurs cost since destinations need to re-send GET messages, the results presented in Sec. V show that the cost is fairly small. The capability-based designs [16] - [17] also share the same spirit with "off by default" and D-PID. In these approaches, a sender first obtains the permission from the destination in order to send data packets to it. The destination provides the capabilities to the sender if it wants to receive packets from the sender. The sender then embeds the obtained capabilities into packets. Routers along the path from the sender to the destination verify the capabilities in order to check whether or not the destination wants to receive the packets. If not, the routers simply discard the packets. D-PID differentiates from the capability-based approaches in two aspects. One hand, communications are initiated by receivers in D-PID but by senders in capability based approaches. On the other hand, the capability-based approaches are vulnerable to "denial-of capability" attacks, where compromised computer(s) sends plenty of capability requests to a victim, thus preventing normal users to obtain the capability from the victim. By contrast, D-PID effectively mitigates such attacks because of three reasons. First, the GET messages carry the PIDs along the paths from the compromised computers to the victim. Second, the PIDs are negotiated by neighbouring domains that can verify the authenticity of PIDs when they forward GET messages.

III.PROBLEM STATEMENT

The impact off DDoS Flooding attack ranges from the simple inconvenience to use a particular service to causing major failures at the targeted server. The existing methods used static path identifiers(S-PIDs) for communication between the routers or resource manager or systems. Due to the static nature it is easy to launch an attack by an attacker like DDoS flooding attacks. To overcome this issue, Dynamic path identifiers (D-PIDs) are one of the proposed methods to detect and defend the DDoS flooding attacks.

IV. METHODOLOGY

BACKGROUND AND MOTIVATION

A. Brief Introduction to CoLoR

CoLoR is a receiver-driven information centric network architecture that assigns unique and persistent content names (or service identifiers, SIDs) to content chunks. As in [20] and [27], CoLoR assigns intrinsic secure self-certifying node identifiers (NIDs) to network nodes and ASes so that authenticating a node/AS does not require an external authority such as ICANN, thus improving security and privacy. In addition, two neighboring domains negotiate a PID for every inter-domain path between them and the PID is only known by them. The two domains then use the PIDs assigned to their interdomain paths to forward packets from one domain to the other. For this purpose, the routers in a domain maintain an interdomain routing table, which records the PID of each interdomain path and the border router that the PID originates. Furthermore, every domain in the Internet maintains a logically centralized (but may be physically distributed) resource manager (RM) used to propagate the reach ability information of SIDs. Particularly, when a content provider wants to provide a content chunk to consumers, he registers the SID of the content chunk to its local RM. The local RM then registers the SID to its providers or peers, by using an approach similar to the one used in [26]. When a content consumer wants to obtain a piece of content, it sends out a GET message to its local RM. RM forwards the GET message to that node. Otherwise, the RM forwards the GET message to the RM in a neighbouring domain (toward the content provider) over a secure channel between the two RMs (because of the use of intrinsic secure identifiers). During this process, the PIDs of inter-domain paths from the content provider to the content consumer are determined. The content provider then sends the desired content to the content consumer by embedding the collected PIDs into headers of packets for the desired content.

B. Why Dynamically Changing PIDs

It is explained that the necessity of change color to dynamically changing PIDs when PIDs are static. We then present an example to show that an attacker can launch Dodos attacks when he has learnt some PIDs in the network.

1. Two approaches to learning PIDs:

The first approach is to learning PIDs is GET luring, where an attacker uses an end host to register normal content names into the network, thus luring GET messages from content consumers. The attacker can get information about the PIDs in the networks because PIDs are carried by GET messages in the networks. We call such a process as the PID learning stage in the rest of this paper. Fig.1 illustrates the process of GET luring. For ease of presentation, we call the AS where the attacker locates as a luring AS and the Saes that send GET messages to the luring AS tempted Saes. Each node in Fig.1 represents an AS in the Internet, AS J is the luring AS, and ASes A, F, M, R, and S are the tempted ASes. At the beginning, ASJ registers content names into the network. Then, ASes A, F, M, R, and S are lured to send GET messages to AS J. The GET messages received by AS J are shown at the bottom of Fig.1. The attacker then learns the corresponding PIDs in the network.

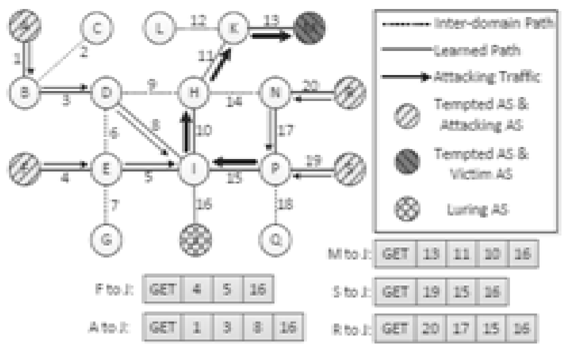


Fig 1: Illustration for GET luring

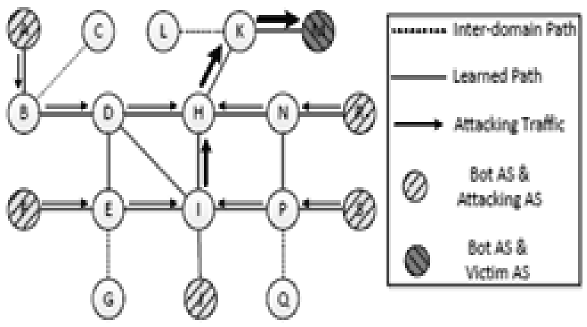


Fig 2: Illustration for the botnet cooperation.

2. Launching DDoS Attacks:

Servers are most powerful in serving the requests because it has several 100Mbps of Internet. To make it busy with one single system with limited Internet speed up-to 1Mbps is not possible. DOS is based on single system, So it required large number of computers. This is the reason that most DOS attacks are actually that is Distributed-Denial of Service. DDOS attacks are used to bring down the websites by using botnet, the Botnet are uploaded in the compromised computer by means of any other methods like virus or Trojans. Virus and Trojans are placed in the Public domain to download, when the system or user is downloads this file the Trojans will settled down in the system and works for the creator of the Trojans. DDOS attack is launched when the attacker have enough systems to launch. The attacker sends messages from the botnet to the targeted system like websites .Trojans on several systems around the world, which may then individually begin attacking a server. This can be very harmful to the website, which due to lack of resources, may shut down for a long time and even get corrupted due to overloading. Note that this is a pessimistic assumption since the inerrability of content in information-centric networking is usually easy to verify [25] - [27]. Then the attacker can order the zombies to flood a victim that should also be along the learned paths. We call such a process as the attacking stage.We call such a process as the attacking stage. Fig. 1 and Fig. 2 illustrate the attacking stage. We call the ASes where the compromised computers (that flood the victim) locate as attacking ASes and the AS where the victim locates as the victim AS. Note that an AS may play multiple roles, e.g., a tempted AS at the PID learning stage may be an attacking AS at the attacking stage. In Fig. 1, AS M is the victim AS, and ASes A, F, R, and S are the attacking ASes that are compromised by the attacker and can flood the victim by using the learned PIDs, as illustrated by the arrowed lines in Fig. 1. In Fig. 2, AS M is the victim AS, ASes A, F, J , R, and S are the attacking ASes, and the attacking traffic is represented by the arrowed lines. From the above descriptions, one can see that it is possible for an attacker to launch DDoS attacks if PIDs are kept secret but static. In addition, since the PIDs carried by data packets are popped out domain-by-domain, the victim does not know the PIDs to the attackers. Accordingly, it cannot trace back them. One may argue that we should not pop out the PIDs when data packets pass through domains. In that case, however, an attacker can try to hide himself by prepending some invalid PIDs at data packets. For instance, the actual PIDs from the content provider S to the content consumer C are PID2 and PID1. In order to hide himself, S can prep an invalid PID before PID2 and PID1. This way, the content consumer C cannot easily find S even if we do not pop out PIDs during the network, packet forwarding process. Therefore, we propose to defend against DDoS attacks by dynamically changing PIDs.

THE D-PID DESIGN

A. Overview of D-PID

One can see that an attacker can learn a part of the PIDs used by domains in the Internet and launch attacks, if the PIDs are static. Thus, the core idea of DPID is to dynamically change the PID of an inter-domain path. In particular, for a given (virtual) path connecting two neighbouring domains A and B, it is assigned a PID and an update period TPID. The update period TPID represents how long the PID of the path should be changed since the PID is assigned. For instance, if path P1 in Fig. 4 is assigned PID1 at time t , the RMs in the two domains should negotiate a new PID (i.e., PID2) for P1 at time t + TPID and a new update period T′PID, by using the negotiation process described in Sec. III-B. At time t + TPID + T′PID, the two RMs will negotiate another new PID (i.e., PID3) for P1. Once the new PID (i.e., PID2) is assigned to the path, the RMs in domains A and B then distribute the new PID (i.e., PID2) to the routers in domains A and B (Sec. III-C). After that, the RMs append the new PID (i.e., PID2) onto GET messages if the path is chosen to carry the corresponding data packets. At the same time, the border routers forward data packets based on the new PID (i.e., PID2). Since some GET packets are forwarded from domain A (or B) to domain B (or A) by using the old PID (i.e., PID1) of the path, the old PID is still valid until t + TPID + T′PID. Without loss of generality, we assume that TPID equals to T′PID in the rest of this paper. That is, the update period of a path is fixed. Note that the new PID of the path is still known only by the two domains. However, it is possible that a communication lasts longer than two update periods. To address this issue, we propose a mechanism similar to the one that the current Internet collects the minimum MTU of networks so that a content consumer knows the minimum update period of PIDs along the path from a content provider to it. Based on this period, the content consumer then re-sends a GET message to the network in order to renew the PIDs along the path. Note also that in D-PID, all domains should dynamically change the PIDs of its inter-domain paths. Depending on its local policy, a domain may simultaneously (or asynchronously) change these PIDs. In the former case, the cost for updating the PIDs is fixed since a domain only needs to distribute the new PIDs to its border routers once every PID update period. In the latter case, every time the PID of an inter-domain path is updated, the domain needs to distribute the new PID to its border routers.

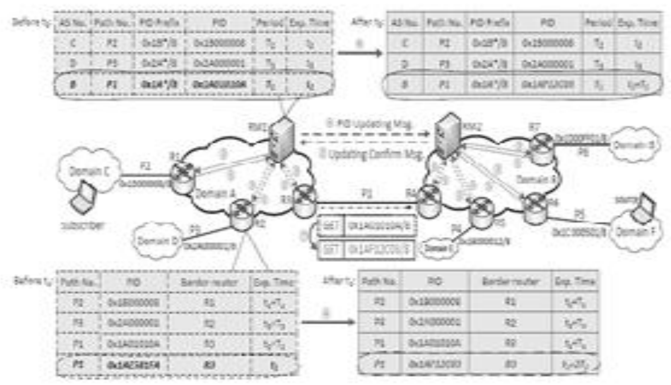


Fig 3: Illustration for PID negotiation & PID update in D-PID.

B. Negotiating PIDs

Since inter-domain packet forwarding is based on PIDs, it is necessary to guarantee that the PIDs used by a domain are different from each other, even if they change dynamically. To achieve this, a direct approach is for domain A to notify its neighbouring domain B the set of PIDs that are used (or conversely, the set of PIDs not used) by domain A, and domain B chooses a PID not used by both domains A and B. However, domains may be reluctant to adopt this approach since it may leak their privacy. More importantly, a domain can have as much as 5,000 neighbouring domains [33] and may simultaneously negotiate PIDs for paths that connect the domain with these neighbours. As a result, two neighbouring domains of a domain A may simultaneously choose a common PID for two paths. This in turn entails multiple rounds of negotiation, leading to a very long negotiation delay.

1. Guarantees the negotiated PIDs are unique; and

2) Requires only one round of negotiation for each path.

To achieve this, two domains negotiate a PID block represented by a PID-prefix (like an IP prefix) to every inter-domain path between them at the bootstrapping stage when they interconnect with each other. This means that a PID-prefix could be used to represent multiple inter-domain paths in the Internet. For instance, the PID-prefix 0x1B000000/8 is assigned by domains A and C to inter-domain path P2 in Fig. 3. At the same time, it is also assigned by domains B and E to inter-domain path P4. This way, it is scalable for domains to assign PID-prefixes since it is not necessary for domains to globally cooperate to assign PID-prefixes to inter- domain paths. As IPv4 addresses, PIDs are 32-bit long in our implementation. First, the largest AS in the current Internet has about 5,000 neighbouring domains [29]. Third, PIDs should not be too long since using longer PIDs consumes more network bandwidth.

C. Setting TPID and TGET

In D-PID, TPID and TGET should be carefully set so that a legal communication will not be interrupted when PIDs change dynamically. To achieve this, we build a mathematical model to calculate the appropriate values of TPID and TGET. Without loss of generality, we assume that there are N inter-domain paths along the path from a server to a client. In particular, we consider the i − th(1 ≤ i ≤ N) inter domain path and assume that it connects two domains A and B. Specifically, we call the timeout period in which a GET message arrives at domain A as the present timeout period and the GET message will be forwarded to domain B. We also assume that the present timeout period begins at time zero. In addition, we assume that the GET message arrives at domain A at time ti0 and the round trip time from domain A to the content provider is _iRTT. For ease of presentation, we denote the timeout period of the it path be TiPID. With these assumptions, if (ti0 +_iRTT) is less than TiPID, the corresponding data packets for the GET message will arrive at domain B in the same timeout period in which the GET message arrives at domain A. In this case, the data packets can be correctly forwarded to domain A. Similarly, if (ti0 + _iRTT) is less than 2TiPID, the corresponding data packets will arrive at domain B in the next timeout period, as shown by Fig. 4(b). In this case, the data packets also can be correctly forwarded to domain A because domain B is able to forward data packets based on the PIDs chosen for the present and the previous timeout periods. However, when (ti0+_iRTT) is larger than 2TiID, as shown in Fig. 5(c), data packets will be dropped by domain B.

This indicates that when we set TPID for an inter-domain path, the value should be greater than the super mum value of the network’s round-trip time. We know that with probability higher than 99.9%, the round-trip time is less than 1.5 seconds. Therefore, we can treat the value of sup (iRTT) as 2 seconds in practice. We now describe how to set the value of the GET retransmission period TGET for an active session. Obviously, the second GET message (i.e., the first retransmitted GET message) arrives at domain A at the time (ti0+TGET). We assume that the first one of these subsequent packets arrives at domain B at time (ti0 + TGET + _iRTT). In the first case, ti0 and (ti0 + TGET + _iRTT) are in the same timeout period. In this case, when domain B receives the data packets sent by the server, it can correctly forward these data packets to domain A. In the second case, (ti0 + TGET + _iRTT) is in the next server, it can correctly forward these data packets to domain A. In the second case, (ti0 + TGET + _iRTT) is in the next timeout period to ti0. In this case, domain B also can correctly forward data packets to domain A because now domain B can forward packets based on both PID1 and PID2. In the third case, (ti0 + TGET+ _iRTT) is larger than 2TiPID. In this case, some data packets will be discarded during the period (2TiPID, ti0 +TGET + iRTT). Therefore, to guarantee the correct data forwarding, it must hold that:

$$t_0^i + \partial_{RTT}^i < 2T_{PID}^i$$

As discussed before, ti0 may be very close to TiPID, so the above in equation can be rewritten as:

$$T_{PID}^i > \sup(\partial_{RTT}^i).$$

The second GET message (i.e., the first retransmitted GET message) arrives at domain A at the time (ti0+TGET). We assume that the first one of these subsequent packets arrives at domain B at time (ti0 + TGET + _iRTT). In the first case, ti0 and (ti0 + TGET + _iRTT) are in the same timeout period. In this case, when domain B receives the data packets sent by the server, it can correctly forward these data packets to domain A.

$$t_0^i + T_{GET} + \partial_{RTT}^i < 2T_{PID}^i.$$

The TiPID and ti0 is very close, so the above in equation can be rewritten as:

$$T_{GET} < T_{PID}^i - \sup(\partial_{RTT}^i).$$

There is an N-inter domain paths between client and server, the data packets are forwarding correctly with the inter domain paths.

$$T_{GET} < \min_{i=1,2.3.....N}(T_{PID}^i - \sup(\partial_{RTT}^i)).$$

V. Results and Discussion

Prototype Design

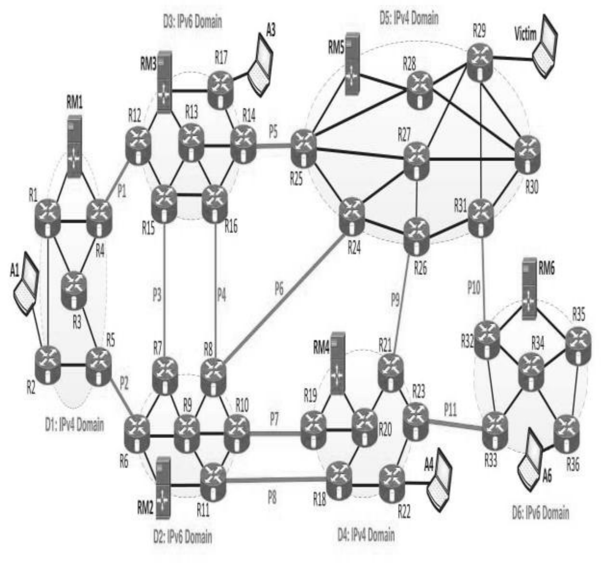


Fig 4: The topology of prototype

The prototype has six domains that use different intra-domain routing protocols. The six domains are inter-connected by 11 inter-domain paths, each of which is assigned with a PID (including the routers, the RMs, and the end-hosts) is running on an aTCA-9300 processor blade, with a four-core Intel Xeon E3 1275V2 processor, an 8 GB DDR3-1600 memory, and six Intel I210 Gigabit Ethernet controllers. The RMs are implemented based on the DPDK platform for fast packet processing, the routers are implemented by using the CLICK software platform, and the end-hosts are implemented as a module in Linux kernel version 2.6.35. We now present the implementation details of the prototype.

1) RMs: Fig. 5(a) shows the structure of the implemented RMs, where "X-protocol" represents the local routing protocol used by the domain where the RM locates. The Registration module is used to process registration messages, and it stores the reach ability information of the registered content names into the SID Table. The GET module is used to process GET messages, and it queries the SID Table in order to determine the next hop for a GET message. The PID Table stores the currently used PIDs for the inter-domain paths associated with the domain where the RM locates. To support D-PID, an entry in the PID table has a timer recording the time that a new PID should be negotiated. When the negotiation completes, the PID distribution module distributes new PIDs to border routers in a domain.

Fig 4(b): Border Router

2) Border Routers: Fig. 5(b) shows the structure of the implemented border routers, where "X-protocol" represents the local routing protocol used by the domain where the border router locates. The Packet Processing module is used to process CoLoR format packets based on the PIDs, and it queries the PID Table to determine the operation for an incoming packet (e.g., encapsulating the packet with an IPv4 packet header and sending it to another border router). The PID distribution module is used to process PID update messages from the RM. When it receives a PID update message, it adds the new PID into the PID table and sends an acknowledgement back to the RM.

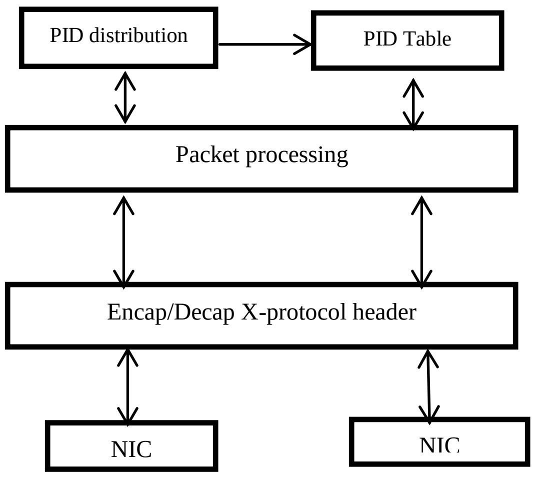
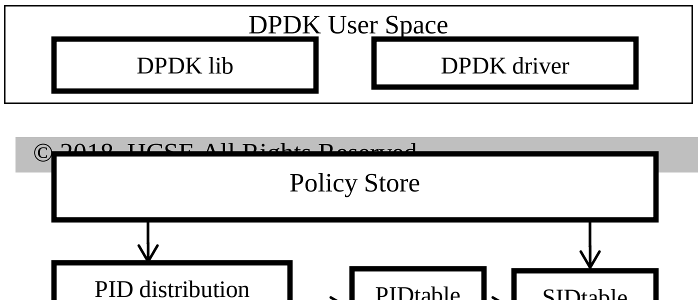
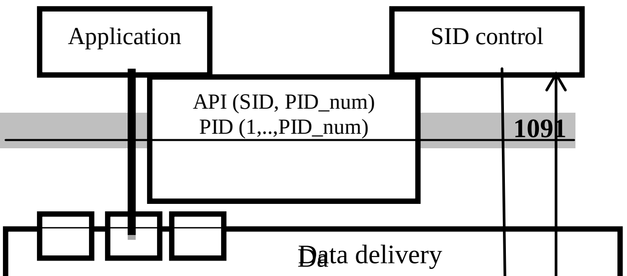


Fig 5(a): Resource Manager





User space

Color

Active socket

Fig 5(b): Border Router

Physical Physical

NICs

Fig 5(c): End hosts

3) End Hosts: Fig. 5(c) shows the structure of the implemented end hosts. In particular, we embed several functionalities into the CoLoR stack in the Linux kernel. To collect the minimum TPID, the DATA module reads the MINIMUM PERIOD field when it receives a data packet, and sets the timer to resend GET messages for the associated session based on MINIMUM PERIOD. When the timer for the session times out, the GET module re-sends the GET message to the content provider in order to refresh the PIDs. When the source receives a resent GET message for an active session, the PID update module refreshes the PID sequence used by the session based on the PIDs contained in the GET message.

It should include important findings discussed briefly. Wherever necessary, elaborate on the tables and figures without repeating their contents. Interpret the findings in view of the results obtained in this and in past studies on this topic. State the conclusions in a few sentences at the end of the paper. However, valid colored photographs can also be published.

VI. PERFORMANCE ANALYSIS

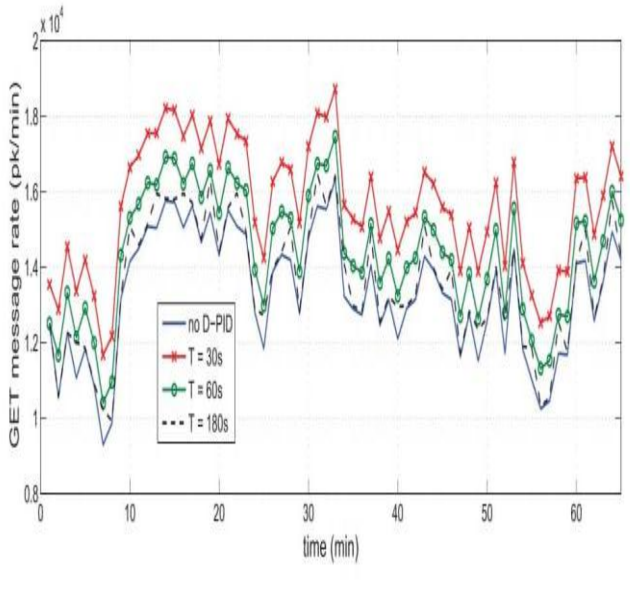


Fig 6: Datacenter trace

Fig. 6 shows the CDF of the PID update rates of all domains per second, the CDF of the peak PID update rates of every domain, and the CDF of the mean PID update rates of every domain, respectively, when the mean value of the update period is set to different values. From the results, we can observe that even if the mean PID update period is 30s, the PID update rate is less than 10 per second with a probability higher than 99%.

Table 1 The Mean GET Message Rate per second

TGET	30s	60s	180s	300s	600s	No PID
DC	260.4	239.7	226.9	220.1	221.0	196.5
Tier-1	20481	17027	14994	14202	14103	13638

Table I shows the average number of GET messages per second for different TGET. From the results, we observe that the extra number of GET messages is about 8.2% (= (239.7−221.5)/221.5) when TGET is 60 seconds. When TGET increases to 300 seconds, however, this number is reduced to about only 1.4% (= (224.6−221.5)/221.5) .The number of GET messages sent by the content consumers per minute when TGET is 30, 60, and 180 seconds, respectively, for the Tier-1 network data trace. From them, we observe that the extra number of GET messages is about 18.7%

(= (17027−14348)/14348) when TGET is 60 seconds. However, this number is reduced to about only 2.2% (= (14662−14348)/14348) when TGET increases to 300 seconds.

VII. CONCLUSION

The design, implementation of D-PID prototype is presented a framework that dynamically changes path identifiers (PIDs) of inter-domain paths in order to prevent DDoS flooding attacks, when PIDs are used as inter-domain routing objects. We have described the design details of D-PID and implemented it in a 42-node prototype to verify its feasibility and effectiveness. We have presented numerical results from running experiments on the prototype. The results show that the time spent in negotiating and Distributing PIDs are quite small (in the order of ms) and D-PID is effective in preventing DDoS attacks. To detect and defend against DDoS flooding attacks by using D-PID is the best approach.

REFERENCES

[1]Francois, I. Aib, and R. Boutaba, “Firecol: a Collaborative Protection Network for the Detection of Flooding ddos Attacks,” IEEE/ACM Trans.on Netw., vol. 20, no. 6, Dec. 2012, pp. 1828-1841.

[2] OVH hosting suffers 1Tbps DDoS attack: largest Internet has ever seen. [Online] Available: https: //www.hackread.com/ovh-hostingsuffers- 1tbps- ddos-attack/.

[3] 602 Gbps! This May Have Been the Largest DDoS AttackinHistory.http://thehackernews.com/2016/01/biggest-ddos-attack.html.

[4] S. T. Zargar, J. Joshi, D. Tipper, “A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks,” IEEECommun. Surv. & Tut., vol. 15, no. 4, pp. 2046 - 2069, Nov. 2013.

[5] P. Ferguson and D. Senie, “Network Ingress Filtering: Defeating Denial of Service Attacks that Employ IP Source Address Spoofing,” IETF Internet RFC 2827, May 2000.

[6] K. Park and H. Lee, “On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets,” In Proc. SIGCOMM’01, Aug. 2001, San Diego, CA, USA.

[7] A. Yaar, A. Perrig, D. Song, “StackPi: New Packet Marking and Filtering Mechanisms for DDoS and IP Spoofing Defense,” IEEE J. on Sel. Areas in Commun., vol. 24, no. 10, pp. 1853 - 1863, Oct. 2006.

[8] H. Wang, C. Jin, K. G. Shin, “Defense Against Spoofed IP Traffic Using Hop-Count Filtering,” IEEE/ACM Trans. on Netw., vol. 15, no. 1, pp. 40 - 53, Feb. 2007.

[9] Z. Duan, X. Yuan, J. Chandrashekar, “Controlling IP Spoofing through Interdomain Packet Filters,” IEEE Trans. on Depend. and Secure Computing, vol. 5, no. 1, pp. 22 - 36, Feb. 2008.

[10] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, “Practical Network Support for IP Traceback,” In Proc. SIGCOMM’00, Aug. 2000, Stockholm, Sweden.

[11] A. C. Snoeren, C. Partridge, L. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer, “Hash-Based IP Traceback,” In Proc.SIGCOMM’01, Aug. 2001, San Diego, CA, USA.

[12] M. Sung, J. Xu, “IP traceback-based intelligent packet filtering: a novel technique for defending against Internet DDoS attacks,” IEEE Trans. OnParall. and Distr. Sys., vol. 14, no. 9, pp. 861 - 872, Sep. 2003.

[13] M. Sung, J. Xu, J. Li, L. Li, “Large-Scale IP Traceback in High-Speed Internet: Practical Techniques and Information-Theoretic Foundation,” IEEE/ACM Trans. on Netw., vol. 16, no. 6, pp. 1253 - 1266, Dec. 2008.

[14] Y. Xiang, K. Li, W. Zhou, “Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics,” IEEE Trans. on Inf.Foren. and Sec., vol. 6, no. 2, pp. 426 - 437, May 2011.

[15] H. Ballani, Y. Chawathe, S. Ratnasamy, T. Roscoe, S. Shenker, “Off by default!,” In Proc. HotNets-IV, Nov. 2005, College Park, MD, USA.

[16] A. Yaar, A. Perrig, and D. Song, “SIFF: a stateless internet flow filter to mitigate DDoS flooding attacks,” In Proc. IEEE Symposium on Securityand Privacy, May 2004, Oakland, CA, USA.

[17] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y. Hu, “Portcullis: Protecting connection setup from denial-of-capability attacks,” In Proc. SIGCOMM’07, Aug.2007, Kyoto, Japan.

[18] X. Yang, D. Wetherall, and T. Anderson, “TVA: A DoS-Limiting Network Architecture,” IEEE/ACM Trans. on Netw., vol. 16, no. 3, pp. 1267 - 1280, Jun. 2008.

[19] X. Liu, X. Yang, and Y. Lu, “To Filter or to Authorize: Network-Layer DoS Defense Against Multimillion-node Botnets,” In Proc. SIGCOMM’ 08, Aug. 2008, Seattle, WA, USA.

[20] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker, “Accountable Internet Protocol (AIP),” In Proc. SIGCOMM’ 08, Aug. 2008, Seattle, WA, USA.

[21] P. B. Godfrey, I. Ganichev, S. Shenker, and I. Stoica, “Pathlet routing,” in Proc. SIGCOMM’09, Aug. 2009, Barcelona, Spain, pp. 111 - 122.

[22] T. Koponen, S. Shenker, H. Balakrishnan, N. Feamster, I. Ganichev, A. Ghodsi, P. B. Godfrey, N. McKwoen, G. Parulkar, B. Raghavan, J. Rexford, S. Arianfar, D. Kuptsov, “Architecting for innovation,” ACM Comput. Commun. Rev., vol. 41, no. 3, July 2011, pp. 24 - 36.

[23] P. Jokela, A. Zahemszky, C. E. Rothenberg, S. Arianfar, P. Nikander, “LIPSIN: Line Speed Publish/Subscribe Inter- networking,” in Proc. SIGCOMM’09, Aug. 2009, Barcelona, Spain, pp. 195 - 206.

[24] H. Luo, Z. Chen, J. Cui, H. Zhang, M. Zukerman, C. Qiao, “CoLoR: an information-centric internet architecture for innovations,” IEEE Network, vol. 28, no. 3, pp. 4 - 10, May 2014.

[25] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, “Named data networking,” ACM Comput. Commun. Rev., vol. 44, no. 3, pp. 66 - 73, Jul. 2014.

[26] T. Koponen, M. Chawla, B. C G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, I. Stoica, “A data-oriented (and beyond) network architecture,” in Proc. SIGCOMM’07, Aug. 2007, Kyoto, Japan, pp. 181 - 192.

[27] D. Raychaudhuri, K. Nagaraja, A. Venkataramani, “MobilityFirst: a robust and trustworthy mobility-centric architecture for the future Internet,” Mobile Comput. and Comm. Rev., vol. 16, no. 3, pp. 2 - 13, Jul. 2012.

[28] M. Antikainen, T. Aura, M. Sarela, “Denial-of-service attacks in bloomfilter- based forwarding,” IEEE/ACM Trans. on Netw., vol. 22, no. 5, pp. 1463 - 1476, Oct. 2014.

[29] BGP Peer Report. http://bgp.he.net/report/peers/.

Authors Profile

Mr. N Vijay pursed Bachelor’s of Technology from University of Jawaharlal Nehru Technological University, Anantapur, Andhra pradesh in 2015 and He is *currently* pursuing *his M.Tech Computer Networks and Information Security in* Sree Vidyanikethan Engineering College Tirupati, Andhra Pradesh. His main research work focuses on Cryptography, Information Security, Big Data Analytics, Data Mining and IoT.

Dr. .M. Sakthivel, Associate Professor,Sree Vidyanikethan Enginnering College, Tirupathi, Andhra Pradesh.