ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 9, March 2018

PREVENTION OF ZOMBIES ATTACKS IN DISTRIBUTED NETWORKS USING DYNAMIC PATH IDENTIFIER

A. George Arokiaraj[1],
Assistant Professor/IT
Idhaya Engineering College for Women,
Chinnasalem

Sr. Maria Anand Milani. S[2]
Assistant Professor/IT
Idhaya Engineering College for Women,
Chinnasalem

ABSTRACT:

The PIDs used in existing approaches are static, which makes it easy for attackers to launch distributed denial-of service (DDoS) flooding attacks. To address this issue, in this paper, we present the design, implementation, and evaluation of D-PID, a framework that uses PIDs negotiated between neighboring domains as inter-domain routing objects. In DPID, the PID of an inter-domain path connecting two domains is kept secret and changes dynamically. We describe in detail how neighboring domains negotiate PIDs, how to maintain ongoing communications when PIDs change.

Keywords: Inter-domain routing, security, distributed denial-of-service (DDoS) attacks, path identifiers.

I.INTRODUCTION

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. Many approaches have been proposed in order to prevent DDoS flooding attacks, including network ingress filtering, IP trace back, capability-based designs, and shut-up messages.

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, but also can facilitate the innovation and adoption of different routing architectures. Luo *et al* proposed an information-centric internet architecture called CoLoR that also uses *PIDs* as inter-domain routing objects in order t o enable the innovation and adoption of new routing architectures.

There are two different use cases of *PIDs* in the aforementioned approaches. In the first case, the *PIDs* are globally advertised. 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. In the latter case, the network adopts an information-centric approach 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 makes it difficult for attackers to launch DDoS flooding attacks since they do not know the PIDs in the network. However, keeping PIDs secret to end users is not enough for preventing DDoS flooding attacks if PIDs are static. For example, Antikainen et al argued that an adversary can construct novel zFilters (i.e., PIDs) based on existing ones and even obtain the link identifiers through reverse-engineering, thus launching DDoS flooding attacks .attacks by learning PIDs if they are static.

To address this issue, in this paper, we present the design, implementation and evaluation of a dynamic PID (D-PID) mechanism. 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 attacker.

All Rights Reserved © 2018 IJARTET

ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 9, March 2018

II. 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. 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 inter domain paths to forward packets from one domain to the other. For this purpose, the routers in a domain maintains an inter domain routing table, which records the *PID* of each inter domain path and the border router that the *PID* originates, as illustrated at the upper right corner in Fig. 1. For instance, the border router in domain *N2* connecting *PID2* in Fig. 1 is *R5*. On the other hand, each domain is free to choose its preferred intra-domain routing architecture so that a domain *A* uses IPv4for intra-domain routing while another domain *B* may use IPv6 for intra-domain routing.

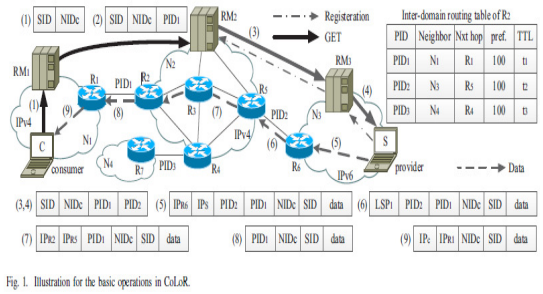


Fig. 1. Illustration for the basic operations in CoLoR.

Furthermore, every domain in the Internet maintains a logically centralized (but may be physically distributed) resource manager (RM) used to propagate the reachability 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 [2]. When a content consumer wants to obtain a piece of content, it sends out a GET message to its local RM. If the desired content is hosted by a local node, the RM forwards the GET message to that node. Otherwise, the RM forwards the GET message to the RM in a neighboring 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.

CoLoR offers several features, First as an information-centric network architecture, routers in the network can locally cache the popular contents so as to serve nearby users, thus reducing redundant transmission and content retrieval delay. Second, it is easy to accurately, timely estimate the traffic matrices of a network since an ingress border router of a domain can know the egress border router of a packet by looking up the inter-domain routing table. Third, CoLoR makes it easy to efficiently integrate information centric networking and software-defined networking. In addition, the data plane in CoLoR is scalable. Finally, CoLoR offers some security benefits while avoiding Interest flooding attacks suffered by both routers and RMs in CoLoR do not maintain pending Interest tables, the PIDs carried in GET messages can be used to trace back attackers.

CoLoR also has some drawbacks that need to be addressed before its real deployment in the future. First, carrying the NID of the content consumer and the desired SID in packet headers reveals user privacy. Second, border routers need to encapsulate/decapsulate outer packet headers (*e.g.*, IPv4 headers), which makes it challenging to realize line-speed packet forwarding. Third, attackers can learn PIDs in the network and launch DDoS attacks in the data plane, if *PIDs* are static. As an attempt to address these drawbacks, in this paper we propose D-PID to prevent DDoS attacks in the data plane.

Why Dynamically Changing PIDs

In this subsection, we explain why it is necessary to dynamically change *PIDs* in CoLoR. To this end, we first present two approaches to learning *PIDs* whey they are static. We then present an example to show that an attacker can launch DDoS attacks when he have learnt some *PIDs* in the network.

1) Two approaches to learning PIDs:

The first approach 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. Since the corresponding *PIDs* are carried by the GET messages, the attacker then can learn a part of *PIDs* in the network. We call such a process as the *PID* learning stage in the rest of this paper. Fig. 2 illustrates the process of GET luring.

All Rights Reserved © 2018 IJARTET

ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 9, March 2018

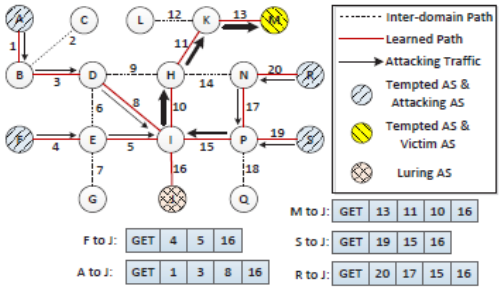
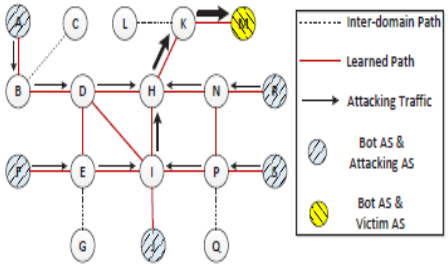


Fig. 2. Illustration for the GET luring.

Another approach to learning *PIDs* is botnet *cooperation*. In botnet cooperation, an attacker is assumed to have controlled a distributed botnet by using various methods such as worms or instant messaging applications. In particular, zombies in the botnet register content names to the network and send GET messages mutually, thus learning the *PIDs* in the network. Fig. 3 illustrates botnet cooperation.



2) Fig. 3. Illustration for the botnet cooperation.

Launching DDoS Attacks:

Once the attacker has learned a part of PIDs in the network, it can freely send packets along the paths represented by the learned PIDs. We assume that the attacker can compromise a number of computers along the paths as zombies, by using similar methods with the ones in the current Internet (*e.g.*, by using worms). Note that this is a pessimistic assumption since the integrality of a content in information-centric networking is usually easy to verify. 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.

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. Therefore, we propose to defend against DDoS attacks by dynamically changing PID.

III.THE D-PID DESIGN

A. Overview of D-PID

From Sec. II-B, 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 neighboring 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.*, PID) 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. Thus, when the PID of the path changes to PID3, ongoing communications may be interrupted. To address this issue, in Sec. III-F 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. However, the cost for

All Rights Reserved © 2018 IJARTET

ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 9, March 2018

updating PIDs in the latter case is significantly less than the update cost of IP-prefixes in the Internet today.

IV. PROTOTYPE IMPLEMENTATION

We verified D-PID's feasibility and effectiveness by implementing it in a 42-node prototype. Our implementation effort was instrumental in refining our design, leading to several revisions. For example, we initially use the approach discussed in the first paragraph in Sec. III-B to negotiate *PIDs*. Below we describe our implementations and present results from running experiments on the prototype.

A. Prototype Design

The prototype has six domains (*i.e.*, *D1 - D6*) that use different intra-domain routing protocols, as shown in Fig. 4. The six domains are inter-connected by 11 inter-domain paths (*i.e.*, *P1 - P11*) each of which is assigned with a *PID*- prefix based on the design in Sec. III. Every domain has one centralized *RM*. Every node in the prototype (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 [3] platform for fast packet processing, the routers are implemented by using the CLICK software platform [4], 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.

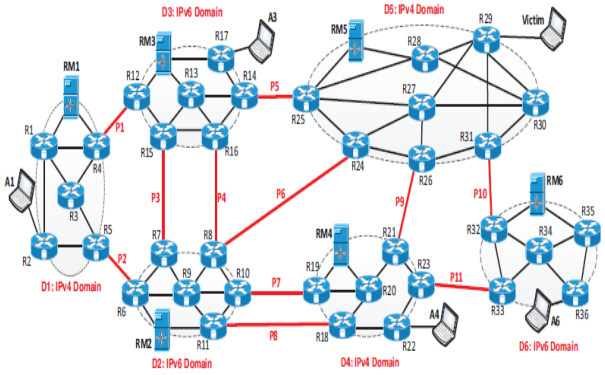
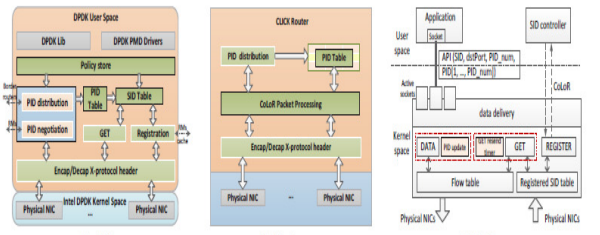


Fig :4



(a) *RMs* (b) Border routers (c) End hosts

Fig:5

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 reachability 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 timer of a *PID* entry times out, the *PID* negotiation module negotiates a new *PID* for the inter-domain path with the associated neighbor *RM*. When the negotiation completes, the *PID* distribution module distributes new *PIDs* to border routers in a domain.

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*. In addition, a *PID* entry in the *PID* table also has a timer recording the time that the *PID* should be removed from the *PID* table. Once the timer of a *PID* entry in the *PID* table expires, the entry is deleted from the *PID* table.

3) End Hosts: Fig. 5(c) shows the structure of the implemented end hosts. We implement CoLoR as an independent protocol stack (as same as the TCP/IP stack) in the Linux kernel, and provide APIs (Application Program Interfaces) for applications to call the CoLoR socket that can send/receive GET, data, and registration messages. 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.

VI. RELATED WORK

Because of the complexity and difficulty in defending against DDoS flooding attacks, many approaches have been proposed in past two decades. A

All Rights Reserved © 2018 IJARTET

ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 9, March 2018

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 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 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 also share the same spirt 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. On 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 neighboring domains that can verify the authenticity of *PIDs* when they forward GET messages. These two reasons makes it convenient to trace back the attackers.

Third, the ubiquitous in-network caching in CoLoR reduces the GET messages sent to the target victim. Named data networking (NDN) [1] is another approach closely related to our work. In NDN, a content consumer sends out an Interest packet when it wants a piece of content. The Interest is routed (by the content name) to the content provider by routers in the Internet. When a router forwards the Interest toward the content provider, it inserts an entry into its pending Interest table (PIT) that stores the content name and the incoming interface of the Interest packet. When the content provider receives the Interest packet, it sends the corresponding Data packet back to the subscriber. The routers then forward the Data packet back to the content consumer according to the PIT entries stored by them. Unfortunately, maintaining a PIT table at routers makes NDN vulnerable to Interest flooding attacks [5]. By contrast, routers in D-PID do not maintain any forwarding state.

In addition, as stated in the previous paragraph, carrying *PIDs* along the path from attackers to the victim makes it convenient to trace back the attackers, thus help preventing them from launching attacks by sending plenty of GET messages.

VII. CONCLUSION

In this paper, we have presented the design, implementation and evaluation of D-PID, 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. We have also conducted extensive simulations to evaluate the cost in launching DDoS attacks in D-PID and the overheads caused by D-PID. The results show that D-PID significantly increases the cost in launching DDoS attacks while incurs little overheads, since the extra number of GET messages is trivial (only 1.4% or 2.2%) when the retransmission period is 300 seconds, and the *PID* update rate is significantly less than the update rate of IP prefixes in the current Internet. To the best of our knowledge, this work is the first step toward

All Rights Reserved © 2018 IJARTET

ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 9, March 2018

using dynamic *PIDs* to defend against DDoS flooding attacks. We hope it will stimulate more researches in this area.

References:

[1] 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.

[2] 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.

[3] Data Plane Development Kit.http://www.dpdk.eu/.

[4] Click Router. http://www.read.cs.ucla.edu/click/.

[5] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "DoS&DDoS in named-data networking," in *Proc. IEEE ICCCN'13*, Aug. 2013, Nassau, Bahamas.

All Rights Reserved © 2018 IJARTET