In-Network Defense Against AR-DDoS Attacks

Xin Zhe Khooi, Levente Csikor, Min Suk Kang
National University of Singapore

Dinil Mon Divakaran
Trustwave

ABSTRACT

The prevalence of the disruptive amplified reflection DDoS (AR-DDoS) attacks is one of the biggest concerns of all network operators today. The increasing magnitude of new attacks are rendering existing measures (e.g., scrubbing services) inefficient. This work demonstrates DIDA, an efficient, topology independent, in-line AR-DDoS detection and mitigation architecture that operates entirely in the data plane.

CCS CONCEPTS

• **Security and privacy → Denial-of-service attacks**; • **Networks → Programmable networks**.

KEYWORDS

Denial-of-service attacks, reflection attacks, amplification attacks, detection and mitigation, in-network, programmable switches

ACM Reference Format:
Xin Zhe Khooi, Levente Csikor, Min Suk Kang and Dinil Mon Divakaran. 2020. In-Network Defense Against AR-DDoS Attacks. In *ACM Special Interest Group on Data Communication (SIGCOMM '20 Demos and Posters), August 10–14, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3405837.3411375

1 INTRODUCTION

Recent trends in cyber security have proven the increasing prevalence and devastating power of Amplified Reflection Distributed Denial-of-Service (AR-DDoS) attacks [5, 12, 14]. Two years after the infamous Mirai attack against Dyn (a major DNS provider) that knocked down most of North Americas's and Europe's Internet for hours [18], the biggest DDoS attack recorded afterwards has targeted Github, the popular online source-code management service, topping out at 1.35 Tbps [12]. Unlike Mirai, attackers did not need any botnet; they exploited vulnerable `memcached` servers to launch an AR-DDoS attack [12]. Later, a customer of a US based service provider was attacked with the rate of 1.7 Tbps using the same `memcached` attack vector [14]. Very recently, furthermore, Amazon has been hit by the largest ever AR-DDoS attack at 2.3 Tbps exploiting the CLDAP protocol (Connection-less Lightweight Directory Access Protocol) [5].

The essence and success of AR-DDoS attacks lie in the disparity in bandwidth consumption between the victim and the attacker itself. In particular, an attacker exploits the connection-less nature of UDP protocol (fundamental services rely on, e.g., DNS, NTP, SSDP, CLDAP) with spoofed requests sent to misconfigured open servers[1] on the Internet, which reply with amplified responses (in size) to a victim. When the disparity is magnified across many requests, whole networking infrastructures can be easily disrupted.

Besides the minimal effort required for an adversary to launch an attack and the millions of vulnerable reflectors (publicly) available on the Internet [1], detecting an attack *in time* is challenging as it appears completely legitimate on the wire. While the former fall beyond the control of a victim network operator, to detect whether a legitimately looking traffic belongs to an attack, stateful inspection of the ingress and egress traffic is required. Particularly, if a vast amount of unsolicited responses are observed with no requests sent before, we can deduce the presence of an AR-DDoS attack.

While network operators traditionally over-provision their networks [13] to absorb small and short-lived attacks, besides the waste of resources in normal operation (i.e., high capital expenditures), we cannot simply rely on it to deal with AR-DDoS attacks of even hundreds of Gbps only. On the other hand, existing scrubbing services already utilize highly scalable IDS/IPS and DPI functions within data centers (either on-premise or on the cloud). However, to comply them to keep track of each user connection (i.e., a request and its corresponding response), complex network-wide tagging mechanisms [8] are required to ensure that both ingress and egress traffic of a given user are processed by the same VM. While some of these issues *could* be alleviated by involving the network controller (e.g., aggregation of switch statistics, sophisticated tagging), keeping track of all user connections in a timely manner would result in a huge control-plane overhead [9]. Furthermore, scrubbing imposes additional latency and high operational costs, not to mention privacy concerns when using third-party cloud-scrubbers (e.g., Akamai [2], Arbor's NETSCOUT [15]).

Here, we showcase **DIDA**, a **D**istributed **I**n-network **D**efense **A**rchitecture [10], which leverages on commodity programmable switches distributed at the network edges (i.e., border and access) and provides real-time in-line detection and mitigation of all variants of AR-DDoS attacks without the need for any interaction with the control plane or any third-party. In particular, we cast DIDA in a realistic environment, enhance the architecture with an efficient monitoring dashboard, and most importantly, we extend our preliminary analysis [10] with multiple different use cases and QoS measurements.

2 DIDA: ARCHITECTURE DESIGN

As a running example, we consider Fig. 1, where an ISP network is under a DNS amplification attack. The green dashed arrows denote the benign requests and responses to and from a public DNS resolver, while the red dashed arrow shows the unsolicited responses (i.e., the attack traffic). First, we need to **keep track of requests and responses**, which an efficient data structure is needed for that fits within the constraints of programmable switches. Hence,

[1] A service is considered vulnerable if no basic countermeasures are implemented, e.g., rate-limiting the number of requests from a particular source.

SIGCOMM '20 Demos and Posters, August 10–14, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8048-5/20/08…$15.00
https://doi.org/10.1145/3405837.3411375

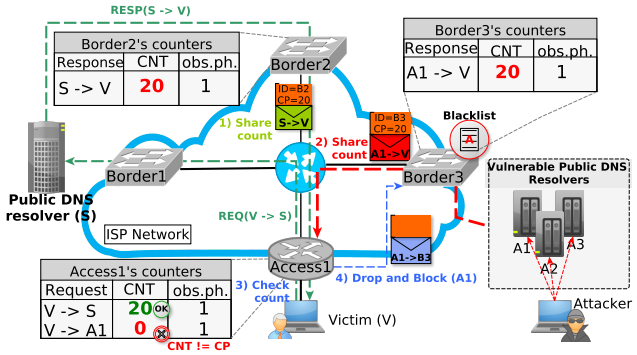

Figure 1: Illustration for the proposed defense mechanisms against a DNS AR-DDoS attack within an ISP. Green arrows: benign traffic; red arrow: attack traffic.

we adopted Count-Min-Sketches (CMS, [7]) due to its sub-linear space requirements. Also, to be independent of the control plane for data structure management tasks (e.g., resetting the counters after a monitoring period), we extend CMS with in-network time management [10, 11]. This is materialized by dividing the precise timestamps of the high-resolution clocks on the commodity programmable switches into broader observation phases, i.e., 10 seconds. Whenever the clock wraps around, we skew all consecutive observation phases accordingly.

Second, for **accurate in-network AR-DDoS attack detection**, network-wide connection tracking is required. As traffic engineering and routing policies may result in different ingress and egress points for a given network traffic, network-wide connection tracking must be topology independent. Thus, the `Border` (i.e., peering side) and the `Access` (i.e., customer-facing) routers are strategically chosen to be replaced by commodity programmable switches (cf. Fig. 1). While a `Border` keeps track of the responses, an `Access` counts the requests of a given communication (e.g. DNS). Then, we need an efficient communication protocol among them with minimal overhead to detect (and mitigate) an attack. In particular, whenever the number of responses at a `Border` reaches a suspicious threshold (say, 20), it shares its counts with the corresponding `Access` by piggybacking the production traffic, i.e., via appending custom headers (tags & counters) to it (step 1 and 2 in Fig. 1). Note that different border routers can have different thresholds. If there is a significant difference w.r.t. the number of request at the `Access` (step 3), an attack is confirmed, and the corresponding `Border` will be notified (by an extra packet) about the attack (step 4).

Lastly, to achieve **timely mitigation**, the `Border` dynamically manages an in-network ACL, where the source address of the abused servers are added upon the confirmation of an attack (`A` is added to the blacklist after step 4). Note, only traffic related to the abused service (e.g., port 53 for DNS traffic) will be filtered and dropped. This is done entirely in the data plane, i.e., the ACL is materialized by an adapted version of a cuckoo hash table [17] using the registers of the programmable switch, hence being controller independent. DIDA is designed to be generic, easily adaptable to other network topologies and connection-less protocols (e.g., NTP, SSDP, CLDAP). For more details, refer to [10].

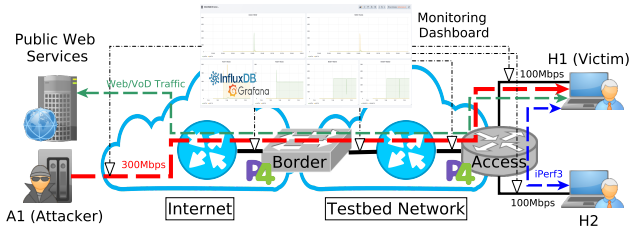

Figure 2: Demonstration setup.

3 DEMONSTRATION

We demonstrate DIDA in our test-bed having Intel Xeon Gold 6230 CPUs, 96GB of memory, equipped with programmable software switches (Version 1.13.0-d447b6a8) and connected back to back.

It comprises the topology depicted in Fig. 2, with hosts *H1* (victim), *H2*, and *A1* (acting as a reflection server). DIDA is prototyped in P4 [4], and the `bmv2` [16] software switches `Access` and `Border` are running the corresponding parts of the architecture. Since `bmv2` is not optimized for performance [3], we limit the maximum bandwidth below the maximum performance of the switches for each link at 500 Mbps using Linux TC to ensure consistent throughput[2] across the setup.

iPerf use case. As a first use case, a persistent `iperf3` session is initiated (as a baseline of the effective bandwidth available) between *H1* and *H2* to fully saturate the links. Then, *A1* replays a DNS amplification trace having 7000 different resolvers as origins [6] at maximum rate (300 Mbps) to exhaust the available link capacity of *H1*. We show that right after the attack has started, DIDA promptly mitigates it within seconds, while exhibiting negligible communication overhead (caused by notifying the `Border`).

Web browsing use case. While the attack is still on, to verify the correctness of our architecture, we show that our mitigation mechanism does not affect the benign Internet browsing activity of the victim *H1*.

VoD use case. As QoS and QoE also depend on other factors than available bandwidth (e.g., latency), we show that under the same attack, even if *H1* is streaming an online video, there are no noticeable impact either.

A brief showcase of our demo is available at https://youtu.be/aDkMkAMw9v4.

ACKNOWLEDGEMENT

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Corporate Laboratory@University Scheme, National University of Singapore, and Singapore Telecommunications Ltd.

REFERENCES

[1] A10 Networks. [n.d.]. DDos Weapons Intelligence Map. https://threats.a10networks.com/ [Accessed: Jun 2020].
[2] Akamai. [n.d.]. Why Akamai Cloud Security for DDoS Protection? Online. https://www.akamai.com/us/en/products/security/ddos-protection-service.jsp [Accessed: Jul 2020].

[2]Based on repeated measurements on our test-bed, the maximum end-to-end throughput across the two software switches is at 500 Mbps.

[3] Antonin Bas. [n.d.]. Performance of bmv2. GitHub, https://github.com/p4lang/behavioral-model/blob/master/docs/performance.md [Accessed: Jun 2020].

[4] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: Programming Protocol-Independent Packet Processors. *SIGCOMM Comput. Commun. Rev.* 44, 3 (July 2014), 87–95. https://doi.org/10.1145/2656877.2656890

[5] Catalin Cimpanu. Jun 2020. AWS said it mitigated a 2.3 Tbps DDoS attack, the largest ever. ZDNet, https://zd.net/3hZ06oF [Accessed: Jun 2020].

[6] Cloudflare. [n.d.]. DNS Amplification DDoS Attack. Blog post, https://bit.ly/31dgJXN [Accessed: Jun 2020].

[7] Graham Cormode and S. Muthukrishnan. 2005. An Improved Data Stream Summary: The Count-Min Sketch and Its Applications. *J. Algorithms* 55, 1 (April 2005), 58–75. https://doi.org/10.1016/j.jalgor.2003.12.001

[8] Seyed K. Fayaz, Yoshiaki Tobioka, Vyas Sekar, and Michael Bailey. 2015. Bohatei: Flexible and Elastic DDoS Defense. In *Proceedings of the 24th USENIX Conference on Security Symposium (SEC'15)*. 817–832.

[9] Rob Harrison, Qizhe Cai, Arpit Gupta, and Jennifer Rexford. 2018. Network-Wide Heavy Hitter Detection with Commodity Switches. In *Proceedings of the Symposium on SDN Research (SOSR '18)*. Article 8, 7 pages. https://doi.org/10.1145/3185467.3185476

[10] Xin Zhe Khooi, Levente Csikor, Dinil Mon Divakaran, and Min Suk Kang. 2020. DIDA: Distributed In-Network Defense Architecture Against Amplified Reflection DDoS Attacks. In *2020 IEEE Conference on Network Softwarization (NetSoft)*.

[11] Xin Zhe Khooi, Levente Csikor, Min Suk Kang, and Dinil Mon Divakaran. 2020. Towards In-Network Time-Decaying Aggregates for Heavy-Hitter Detection. In *ACM Special Interest Group on Data Communication (SIGCOMM'20 Demos and Posters)*. https://doi.org/10.1145/3405837.3411402

[12] Lily Hay Newman. Jan 2018. GitHub Survived the Biggest DDoS Attack Ever Recorded. Wired, https://www.wired.com/story/github-ddos-memcached/.

[13] Nick Martin. May 2014. Overprovisioning VMs may be safe, but it isn't sound. Blog post, https://bit.ly/37oCELJ.

[14] Carlos Morales. Mar 2018. NETSCOUT Arbor Confirms 1.7 Tbps DDoS Attack. NETSCOUT blog, https://www.netscout.com/blog/asert/netscout-arbor-confirms-17-tbps-ddos-attack-terabit-attack-era [Accessed: Jun 2020].

[15] NETSCOUT. [n.d.]. Arbor Cloud DDoS Protection Services. Online. https://www.netscout.com/product/arbor-cloud [Accessed: Jul 2020].

[16] p4lang. [n.d.]. The BMv2 Simple Switch target. GitHub, https://github.com/p4lang/behavioral-model/blob/master/docs/simple_switch.md [Accessed: Jun 2020].

[17] Rasmus Pagh and Flemming Friche Rodler. 2004. Cuckoo Hashing. *J. Algorithms* 51, 2 (May 2004), 122–144. https://doi.org/10.1016/j.jalgor.2003.12.002

[18] Nicky Woolf. Oct 2016. DDoS attack that disrupted internet was largest of its kind in history, experts say. The Guardian, https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet [Accessed: Jun 2020].