A Demonstration of the DeDoS Platform for Defusing Asymmetric DDoS Attacks in Data Centers

Henri Maxime Demoulin*
University of Pennsylvania

Tavish Vaidya*
Georgetown University

Isaac Pedisich
Nik Sultana
Bowen Wang
University of Pennsylvania

Jingyu Qian
Yuankai Zhang
Georgetown University

Ang Chen
Andreas Haeberlen
Boon Thau Loo
Linh Thi Xuan Phan
University of Pennsylvania

Micah Sherr
Clay Shields
Wenchao Zhou
Georgetown University

ABSTRACT

We propose a demonstration of DeDoS, a platform for mitigating asymmetric DDoS attacks. These attacks are particularly challenging since attackers using limited resources can exhaust the resources of even well-provisioned servers. DeDoS resolves this by splitting monolithic software stacks into separable components called minimum splittable units (MSUs). If part of the application stack is experiencing a DDoS attack, DeDoS can massively replicate *only* the affected MSUs, potentially across many machines. This allows scaling of the impacted resource separately from the rest of the application stack so that resources can be precisely added where needed to combat the attack. Our demonstration will show that DeDoS incurs reasonable overheads in normal operations and that it significantly outperforms naïve replication when defending against a range of asymmetric attacks.

CCS CONCEPTS

• **Networks** → **Denial-of-service attacks**; *Data center networks*;
• **Software and its engineering** → Scheduling;

KEYWORDS

Denial-of-Service attacks, Distributed Systems, Security, Real-time scheduling

ACM Reference format:
Henri Maxime Demoulin, Tavish Vaidya, Isaac Pedisich, Nik Sultana, Bowen Wang, Jingyu Qian, Yuankai Zhang, Ang Chen, Andreas Haeberlen, Boon Thau Loo, Linh Thi Xuan Phan, Micah Sherr, Clay Shields, and Wenchao Zhou. 2017. A Demonstration of the DeDoS Platform for Defusing Asymmetric DDoS Attacks in Data Centers.

In *Proceedings of SIGCOMM Posters and Demos '17, Los Angeles, CA, USA, August 22–24, 2017,* 3 pages.

*First Co-Authors

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCOMM Posters and Demos '17, August 22–24, 2017, Los Angeles, CA, USA
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5057-0/17/08.
https://doi.org/10.1145/3123878.3131990

https://doi.org/10.1145/3123878.3131990

1 INTRODUCTION

Distributed denial-of-service (DDoS) attacks have matured from simple flooding to pernicious asymmetric attacks that amplify the attacker's strength by exploiting asymmetries in protocols [8–10]. These attacks typically involve clients launching attacks that consume the computational resources or memory on servers in data centers. Types of asymmetric DDoS vary, and are often targeted at a specific protocol. An invariant of these attacks is that they exploit a fixed resource. For example, the SlowLoris/SlowPOST attacks function by establishing HTTP connections with the victim webserver; requests are sent at a very slow rate to inflate their lifetime, consuming connection resources at the target [11]. The ReDoS attack uses specially crafted regular expressions that are slow to evaluate, amplifying the cost of serving malicious clients' requests [7]. Renegotiation attacks exploit an asymmetry in the SSL/TLS protocol: the server's cost of engaging in a SSL/TLS handshake is ten times that of a client's [3].

A straightforward defense mechanism against asymmetric attacks is simply to deploy more resources within the data center. This is often the de facto defense deployed in production systems: during the course of an attack, the service is automatically replicated as virtual machines (VMs) on multiple machines to scale "elastically" to support the extra load. However, this approach is enormously costly and, as we will show, rather inefficient. When a virtual machine comes under the load of a DDoS attack, the entire machine is replicated. This causes replication of *all* resources of the VM, regardless of which are being consumed. For example, if only a TCP state table is being exhausted (e.g., due to a SYN flood), the replication of the entire monolithic software stack mitigates the attack, but does so at enormous overhead (since presumably the TCP state table represents a minuscule portion of the system's overall footprint).

We present the demonstration of a radically different approach, called DeDoS, which aims to *defuse* DDoS attacks via fine-granularity replication. DeDoS is the next version of our initially proposed *SplitStack* architecture [1]. DeDoS has two key elements. First, we

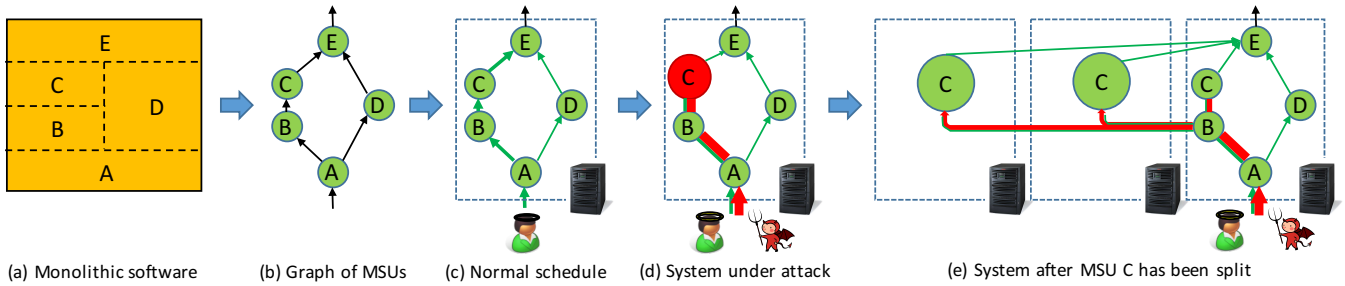

Figure 1: Example use case of DeDoS. The monolithic software (a) is transformed into a dataflow graph (b) with smaller components, called MSUs, which are then scheduled on the available machines (c). When an attacker attempts to overload one of the components (d), DeDoS disperses the attack by generating additional instances on other machines (e).

propose to break up the monolithic network stack into small components that can be moved and replicated independently. This is inspired by the current trend towards micro-services, but our vision goes much further: we aim to operate at a much smaller granularity with composable components, each of which can handle some small, focused aspect of an application that may be vulnerable to resource exhaustion. Example components include code specifically for performing TCP or TLS handshakes. Second, we propose an adaptive controller that makes real-time decisions on placing these components within physical resources in a data center, and then adaptively clones, merges, or migrates these components in order to meet service-level agreement (SLA) objectives. When SLA objectives are violated, this is treated as a potential attack, and individual components that are overloaded due to a DDoS attack are replicated.

The DeDoS architecture offers two benefits for defending against asymmetric attacks. First, the fine-grained components make it easier for the defender to deploy all available resources on all machines against the attacker, exactly as needed. For instance, DeDoS could respond to a TLS renegotiation attack by temporarily enlisting other machines with spare CPU cycles to help with TLS handshakes. Second, and more importantly, the reactive replication approach is not attack-specific and can thus potentially mitigate unknown asymmetric attacks. Once DeDoS recognizes that a component is overloaded or its throughput appears to drop, it can respond by replicating that particular component – without having seen the attack before, and without knowing the specific vulnerability that the attacker is targeting. This allows a flexible and automatic response against mixed attacks – which is especially useful because DDoS attacks today tend to use multiple attack vectors [4].

2 DEDOS DESIGN

In DeDoS, each application consists of several small components that we call *minimum splittable units (MSUs)*. Each MSU is responsible for some particular functionality: for instance, a web server might contain an HTTP MSU, a TLS MSU, a page cache MSU, and several other MSUs of a similar size (Figure 1a). Even the TCP/IP stack itself could be a MSU, or it could be broken into even smaller components, such as MSUs for congestion control, buffering, or the three-way handshake.

Related MSUs can and do frequently communicate with each other. For instance, the packets of an incoming HTTPS connection might enter the system at a network MSU; from there, the data might flow through the TCP/IP MSUs, it might be decrypted by the TLS MSU, the request might be decoded by the HTTP MSU, etc. Collectively, the MSU form a *dataflow graph* that contains a vertex for each MSU and an edge for each pair of MSUs that can communicate (Figure 1b). This dataflow graph is usually in the form of a directed acyclic graph (DAG).

Each DeDoS deployment contains a central *controller* that decides how many instances of each MSU should exist, and which nodes they should run on. Initially, the controller makes a normal scheduling decision, based on the application's performance requirements. For instance, it might decide that a certain number of HTTP MSUs—along with the corresponding TLS MSUs, page cache MSUs, etc.—is needed to answer each web request within 50ms, and it might then instantiate this many MSUs and place them on different physical machines (Figure 1c).

However, at runtime, the controller keeps collecting statistics about the available resources and the performance of each MSU. If it detects that some MSU instances are overloaded due to an unknown attack (Figure 1d), it creates additional instances of the MSUs that are under attack, and places them on machines where the relevant resources are still available (Figure 1e).

DeDoS does not require applications to be written from scratch: it is possible to split existing codebases into MSUs, although the necessary (manual) effort depends on the codebase. In some cases, the modular nature of the code lends itself naturally to splitting; this was the case for the user-level TCP library that we use in our demo. There are other networking codebases – such as Click [5] – that already have a modular architecture and could presumably be split easily. Even for legacy applications where full-scale manual splitting is impractical, DeDoS can still be useful if a few particularly vulnerable components can be split out, moved and replicated independently.

Another possible approach is to – partially or fully – automate the partitioning. Some domain-specific languages are already written in a structured manner that lends itself naturally to this approach; for instance, a declarative networking [6] application can be compiled to a MSU graph that consists of database relational operators and operators for data transfer across machines. Work in the OS community [2] has shown that even very complex software, such as the entire Linux kernel, can be split in a semi-automated fashion. We are developing ways to further automate this process in our ongoing work.

Acknowledgements. This material is based upon work supported in part by the the Defense Advanced Research Projects

Agency (DARPA) under Contract No. HR0011-16-C-0056, and NSF grants CNS-1513679, CNS-1563873 and CNS-1527401. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA or NSF.

REFERENCES

[1] Ang Chen, Akshay Sriraman, Tavish Vaidya, Yuankai Zhang, Andreas Haeberlen, Boon Thau Loo, Linh Thi Xuan Phan, Micah Sherr, Clay Shields, and Wenchao Zhou. 2016. Dispersing Asymmetric DDoS Attacks with SplitStack. In *ACM Workshop on Hot Topics in Networks (HotNets)*.

[2] Alain Gefflaut, Trent Jaeger, Yoonho Park, Jochen Liedtke, Kevin J. Elphinstone, Volkmar Uhlig, Jonathon E. Tidswell, Luke Deller, and Lars Reuther. 2000. The SawMill Multiserver Approach. In *Proc 9th ACM SIGOPS European Workshop*. 109–114.

[3] IETF. 2011. SSL Renegotiation DoS. Accessed July 10, 2017 from https://www.ietf.org/mail-archive/web/tls/current/msg07553.html.

[4] Christine Kern. 2016. Increased Use Of Multi-Vector DDoS Attacks Targeting Companies. http://www.bsminfo.com/doc/increased-use-of-multi-vector-ddos-attacks-targeting-companies-0001.

[5] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. 2000. The Click Modular Router. *ACM Trans. Comput. Syst.* 18, 3, 263–297.

[6] Boon Thau Loo, Tyson Condie, Minos Garofalakis, David E. Gay, Joseph M. Hellerstein, Petros Maniatis, Raghu Ramakrishnan, Timothy Roscoe, and Ion Stoica. 2009. Declarative networking. *Comm. ACM* 52, 11, 87–95.

[7] OWASP. 2017. Regular expression Denial of Service - ReDoS. Accessed July 10, 2017 from https://www.owasp.org/index.php/Regular_expression_Denial_of_Service_-_ReDoS.

[8] John Pescatore. 2014. *DDoS Attacks Advancing and Enduring: A SANS Survey*. Technical Report. SANS Institute.

[9] Christian Rossow. 2014. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *Proc. NDSS*.

[10] Fabrice J. Ryba, Matthew Orlinski, Matthias Wählisch, Christian Rossow, and Thomas C. Schmidt. 2015. Amplification and DRDoS Attack Defense – A Survey and New Perspectives. *CoRR* abs/1505.07892. http://arxiv.org/abs/1505.07892

[11] David Senecal. 2013. Slow DoS on the Rise. https://blogs.akamai.com/2013/09/slow-dos-on-the-rise.html.