Anti-DDoS prevention measures through MRU task scheduling principles

Brandon Vo   
*New York University: Tandon School of Engineering*

*Abstract*-This research paper will conduct several experiments investigating the effects that a Distributed Denial of Service Attack has on locally hosted networks by creating a simulated network. The simulated network will be used to observe how low-end servers would suffer under having to service groups of clients and attackers that occasionally or constantly send queries to the server. Then, the research paper will observe the effects of a server experiencing a DDoS attack and network congestion by measuring traffic flow directed at the server. Finally, the research paper will observe the change in network performance by implementing task scheduling principles and guidelines to allow a server to counter large influxes of incidental client requests or intentional flooding attacks directed by malicious actors.

*Keywords*-Operating Systems, OS, DDoS, Cloud-Based, botnet, adversary, host, Mininet, mitigation, priority queue, packets, I/O, IP Address, MRU, Most Recently Used

1. Introduction

As our world becomes more interconnected, small businesses, companies, and even certain individuals may opt to provide online services to anyone around the world through the internet. As computer architecture becomes more faster, more accessible, and more powerful, adversarial groups start to have more options to disrupt such activities through DDoS attacks, leading to a virtual arms race between server owners and adversaries to see whose network processing power can outmatch the opponents. While more sophisticated methods of identifying and bypassing DDoS attacks have been theorized and tested, many solutions struggle to the exponentially increasing scale of botnets networks that can overtake even the networks of industrial giants of the industry.

This issue is caused in part because many of these solutions question how to identify and stop an attack rather than how to adapt to such a large-scale attack. Rather than finding out how to defeat a DDoS attack, this research instead aims to find a method to adapt to such an attack such that the core functionality of the defending network can remain in control. The goal of this experiment would determine the possibility whether it could be possible to use the scheduling disciplines of Operating Systems to manage the scale of network queries in such a way that a DDoS attack would not disrupt the server’s control of its network. This research paper will describe the software being used to simulate a network under normal circumstances and a network undergoing a DDoS attack along with the layout of the network itself. Then, the experiment will be conducted showing three experiment trials: One where the network is experiencing normal traffic flow, one where the network is experiencing a DDoS attack, and one where the server is experiencing a DDoS attack but using the described DDoS prevention measures implemented. Data will be gathered and recorded along with an explanation regarding the effectiveness or ineffectiveness of the experiment trials and perceived flaws of the experiment along with improvements that could be utilized to improve the reliability of the experiment.

1. Related Research

[3] A. Bremler has attempted to formulate an anti-DDoS scaling mechanism before. His method was named “Scale Inside-Out” where the researcher attempted to mitigate DDoS attacks by attempting to identify the strength and availability of a server’s hardware undergoing a DDoS attack and determining how many requests the server can cut down to reduce resource contention. Compared to the goal of my method, this researcher’s method specifies that their method sacrifices a server’s own resources in order to establish and maintain its own mitigation scheme. This method tackles DDoS attacks by cutting as much of the incoming traffic as possible which would come at a cost at an honest user’s expense. On the other hand, my method is focused on fulfilling all requests but focusing on requests from assumed trusted users.

[4] Researcher Osanaiye reviewed and discussed a mitigation method using an emergency deployment of extra resources to assist in combatting the effects of a DDoS attack. One related research method looked into how cloud computing could take advantage of this opportunity by allowing idle hardware to be deployed as part of a server in an effort to temporarily inflate the scale of the server’s resource power to better handle a DDoS attack. This, unfortunately, requires having extra resources on demand and being able to quickly retailor server software and hardware to service the needs of a user, a situation which mainly Cloud Computing can take advantage of. Compared to my own, this method is designed specifically for handling DDoS attacks for Cloud-Based Networks where an owner already has extra resources available. This situation isn’t applicable for all users. On the other hand, my method is designed with the assumption that the resources being provided by the server are the only resources available. Their method is about increasing the power granted by a server while my method is focused on managing with the resources we already have.

[1] Bremler-Barr’s method has a similar goal of establishing control and maintaining a connection with already-existing users. This solution uses a high priority queue and a low priority queue where honest users are given higher priority while malicious users are given lower priority in order to allow high priority users to maintain basic levels of quality of service. This method involves analyzing packet data and calculating the suspicion based on the Harmonic Mean. Packets that exceed the mean are deemed suspicious and put in low priority queue.

The method described above is focused on identifying what packets and what source IP Addresses are malicious while my method is focused on identifying what sources are legitimate. My method is based on identifying packets based off their sources and prioritizing IP Addresses that were being serviced under the Most Recently Used (MRU) principle used in Operating Systems. This researcher emphasizes the challenge of identifying and reducing the chances of misidentifying users as adversaries. This method prefers false positives. My method, on the other hand, is focused on handling all messages and focusing on priority rather than what messages to discriminate against. My queue system has a goal of servicing all messages even if some requests won’t be handled in just time due to lower priority. While both of our methods and goals are similar, our solutions approach the problem from different angles.

1. Motivating Example

DDoS attacks are becoming more widespread and accessible for adversaries and attackers. Likewise, it has become profitable for these same attackers to profit off botnet services through ransom or selling their services to malicious clients. These attacks will continue to grow as computer hardware becomes cheaper, more efficient along with the restrictions of bandwidth slowly being mitigated as network speeds grow. It has become increasingly less viable to purchase expensive servers and hardware to outperform an attacker who can rent out thousands of botnets. Rather, finding a solution to maintaining control over the host’s network, server control, and quality of service for its users during intense periods of DDoS attacks must be found in order to prevent leaving a host’s service and client’s session from falling into ransom through these DDoS attacks.

1. Network Design and Specifications

The network was designed on Mininet, software designed to create and host a simulated local network where every computer is physically connected to one another.

The network topology created for this experiment involves the use of one root switch for the Core Layer, 2 switches are connected to the root switch to form the Aggregation Layer, and a total of 8 switches where 2 switches are each connected to one switch in the aggregation layer in order to form the Edge Layer. The host layer forms the final layer where a total of 32 hosts are created. Small LAN networks of 4 hosts are each connected to a switch in the edge layer, forming our network topology. This allows every host to be connected to each other through a series of intermediary switches depending on a host’s relative location from one another.

This network topology was used in order to imitate the 2-layer network topology commonly used in data centers. In addition, each switch was intentionally designed with a 5ms delay before forwarding a packet. However, every packet sent in the network also has a 5% chance of being dropped during transmission with no means of recovery as a means to simulate the network traffic required for a client to reach a server host. This means that hosts located furthest away from the server will have to travel through more switches, increasing the delay and increasing the possibility of packet lost similar to an actual network.

In order to further emulate the resource strain of low-end hardware, the topology created each host with a limited amount of CPU power based off the total number of hosts in the topology. This means that the total amount of CPU power allotted to the network topology is evenly distributed to every host in the network. Of the 32 hosts chosen, one server is chosen to serve as the server where it will perform an action upon receiving a request from any users in the network. The remaining 31 hosts will be provided code to serve as either a standard client user or an attacker using botnet functionality.

# Empirical Evidence

The following trials used I/O graphs to measure the traffic flow of the localized, simulated network of 32 hosts where one host would act as the server while the remaining 31 hosts would act as the attackers. These experiments were recorded on a simulated network created through Mininet with the I/O network performance recorded by Wireshark. The measurements are based on the number of packets being sent out (X-axis) over the time in seconds (Y-Axis).

1. I/O graph of a server experiencing standard network flow with clients

A picture containing text, boat

Description automatically generated

1. I/O graph of a server experiencing standard network flow with clients

A picture containing text, boat, water

Description automatically generated

1. I/O graph of a server experiencing standard network flow with clients for a third trial

A picture containing diagram

Description automatically generated

After the network performance of a standard network was recorded, the next set of simulations were done on the same setup of one host and 31 other clients. However, these clients were changed into attackers where they would not stop sending request queries similar to an actual botnet.

1. I/O graph of a server experiencing a DDoS attack

Chart

Description automatically generated with low confidence

1. I/O graph of a server experiencing a DDoS attack during a second trial

Chart

Description automatically generated with low confidence

The next set of graphs highlight the experiment of the same Mininet network of one host and 31 attackers during a DDoS attack. However, the server is now using an implementation of the DDoS priority system for this experiment. The server would now record a limited list of IP Addresses from each message received and store these messages into a queue. When the queue reaches maximum capacity, the server will check its list of remembered IP Addresses to see if the message was from a previously recorded client. The server will service the request if a matching IP Address was found or force the message back into queue if there is no match. However, if there is no match and the server has noticed that an IP Address has been serviced over a certain threshold, the IP Address will be replaced with the new IP Address from the message before the server responds to the message.

1. I/O Graph of a server using an implementation of anti-DDoS scaling defense against a DDoS attack

Chart, line chart

Description automatically generated

1. I/O graph of the second trial of the server using anti-DDoS scaling measures during the second experiment

Chart, line chart

Description automatically generated

1. I/O graph of the third trial of the server using anti-DDoS scaling measures during the third experiment

Chart, line chart

Description automatically generated

As shown in the experiment results, the mitigation mechanism has removed the largest peak in activity, although the rest of the I/O activity shows an overall increase in network activity similar to the trial of a DDoS attack with no mitigation techniques. The most noticeable change is that there are fewer intervals of activity, but each interval is noticeably wider as though the batch of requests have been smoothened out longer periods of time. This indicates that instead of having sudden influxes of activity, the server takes the large batch of requests and takes longer time to handle the activity at its own pace.

1. Conclusions and Future Work

The intent of this research paper was to observe the effects a DDoS attack has on a server. With DDoS attacks becoming an inevitable threat faced by any computer open for public access, it is becoming more important for server owners to have accessible means to counter the growing threats of online botnets. Unfortunately, the results of this paper’s experiments have been determined to be inconclusive. While the experiment has successfully shown a marginal change in behavior experienced by the server, the state shown by the server is not enough to determine if this altered behavior provides beneficial, malingering, or neutral effects. The server’s performance could not be closely identified due to the limitations of being a simulated host created by software that does not allow for host networks to be placed under closer scrutiny.

If this research experiment needed to yield more conclusive evidence, the experiment would need to be repeated under more true to life scenario with circumstances that allow for more data to be gathered. Until then, server operators and hosts will have to rely on methods with more verifiable proof of work and reliability.

##### References

1. A. Bremler-Barr, E. Brosh and M. Sides, "DDoS attack on cloud auto-scaling mechanisms," IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, 2017, pp. 1-9, doi: 10.1109/INFOCOM.2017.8057010.

<https://ieeexplore.ieee.org/abstract/document/8057010>

1. A. Sardana and J. R.C., "An Integrated Honeypot Framework for Proactive Detection, Characterization and Redirection of DDoS Attacks at ISP Level," *Journal of Information Assurance and Security 1,* vol. 1, no. 1, pp. 1-15, 2008.

<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.580.3346&rep=rep1&type=pdf>

1. G. Somani, M. S. Gaur, D. Sanghi, M. Conti and M. Rajarajan, "Scale Inside-Out: Rapid Mitigation of Cloud DDoS Attacks," in IEEE Transactions on Dependable and Secure Computing, vol. 15, no. 6, pp. 959-973, 1 Nov.-Dec. 2018, doi: 10.1109/TDSC.2017.2763160.

<https://ieeexplore.ieee.org/abstract/document/8068248>

1. Opeyemi Osanaiye, Kim-Kwang Raymond Choo, Mqhele Dlodlo, Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework, Journal of Network and Computer Applications, Volume 67, 2016, Pages 147-165, ISSN 1084-8045  
   https://doi.org/10.1016/j.jnca.2016.01.001. (https://www.sciencedirect.com/science/article/pii/S1084804516000023)
2. R. V. Deshmukh and K. K. Devadkar, "Understanding DDoS Attack & its Effect in Cloud Environment," ScienceDirect, vol. 49, pp. 202-210, 2015.
3. S. Yu, W. Zhou and R. Doss, "Information theory based detection against network behavior mimicking DDoS attacks," in IEEE Communications Letters, vol. 12, no. 4, pp. 318-321, April 2008, doi: 10.1109/LCOMM.2008.072049.

<https://ieeexplore.ieee.org/abstract/document/4489680>