An Authentication Technique to Handle DDoS Attacks in Proxy-Based Architecture

Poonam Dharam
Computer Science and Information Systems
Saginaw Valley State University
Saginaw, MI, USA
email: pdharam@svsu.edu

Jarin Musarrat
Computer Science and Information Systems
Saginaw Valley State University
Saginaw, MI, USA
email: jmusarra@svsu.edu

***Abstract*— Recent years have witnessed an increase in Distributed Denial of Service (DDoS) attacks that overwhelm available network and backend server resources such as bandwidth, buffers, etc. To handle such attacks, proxy-based network architectures have been implemented to manage and Load Balance incoming traffic by spawning new servers in the event of unexpected rise in network traffic. However, DDoS attacks continue to persist with the attack target shifting from the main backend application servers to proxy servers. The redirection of users to one of the available proxy servers results in the discovery of their (proxy server's) IP address. A botnet can then be used by the attacker to generate a huge amount of traffic and direct it to the proxy server, thus causing DDoS. In this paper, we propose an authentication technique to ensure the uniform distribution of the incoming requests and to avoid/drop the illegitimate requests from occupying servers' resources. Our simulation results show that the proposed solution detects and handles DDoS attacks in an efficient manner.**

Keywords-Distributed Denial of Service; proxy-based architecture; flooding attacks.

I. INTRODUCTION

Internet Distributed Denial of Service (DDoS) attacks have emerged as one of the biggest threats to Internet security, with thousands of them occurring every year. Hackers are turning to DDoS to bring down organizations' services and to compromise their sensitive data. Recent times have witnessed a dramatic increase in such attacks due to the declining cost of launching an attack and the popularity of Internet of Things (IoT) devices [1] that could be used as botnets. Recently, the Mirai botnet [2], that brought down major services including Twitter, Netflix, CNN (Cable News Network), and many others, was largely made up of IoT devices such as digital cameras and Digital Video Recorder (DVR) players [3].

The DDoS attack mainly targets the availability of a service by exhausting the resources associated with the service. In the context of computer and communications, the focus is generally on network services that are attacked over their network connection. A classic flooding DDoS attack [4] involves a significant amount of malicious traffic directed towards a target server. The volume of the attack traffic can be scaled up by using multiple systems that are either compromised user workstations, PCs, or IoT. For example, attackers identify devices that use default login credentials to gain backdoor access to them and install an attack agent that they can control. A large collection of such systems under the control of one attacker can be created, collectively forming a botnet. This traffic overwhelms any legitimate traffic, effectively denying legitimate users' access to the server.

Internet Control Message Protocol (ICMP), User Datagram Protocol (UDP), or Transmission Control Protocol (TCP) SYN packets are most commonly used for flooding attacks. Any packet that is permitted to flow over the links, towards the targeted system, can be used to fill up the available capacity. Such attacks flood the network link with a huge number of malicious packets in turn competing with regular user traffic flowing to the server. Many packets, mostly valid traffic, will be dropped on the path to the server due to the congestion caused by flooding. For example, a DDoS flooding attack on a Web Server involves several valid Hyper Text Transport Protocol (HTTP) requests, each using significant server resources. This then limits the server's ability to service requests from other users. For instance, HTTP requests use TCP as transport layer protocol. For each TCP connection made, some amount of buffer space at the server's end is reserved for reliable data transfer, congestion control, and flow control. Also, the server only has a limited amount of memory for user buffer space. Once the TCP connections fill up the server's buffer, future requests will be either cached or dropped until the buffer space frees up [5]. Another example would be a Web Server that includes the ability to make database queries. If a database query that takes a large amount of time for the server to respond can be constructed, then an attacker could generate many such queries to overload the server. This limits the ability to respond to valid requests from other servers.

Most of the DDoS attacks use forged source addresses to generate large volumes of packets with the target system as destination and randomly selected, usually different, source address for each packet. This, in turn, makes it harder to identify the attacking system. Also, the volume of network traffic can be easily scaled up by using multiple systems.

 ISBN: 978-1-61208-786-3

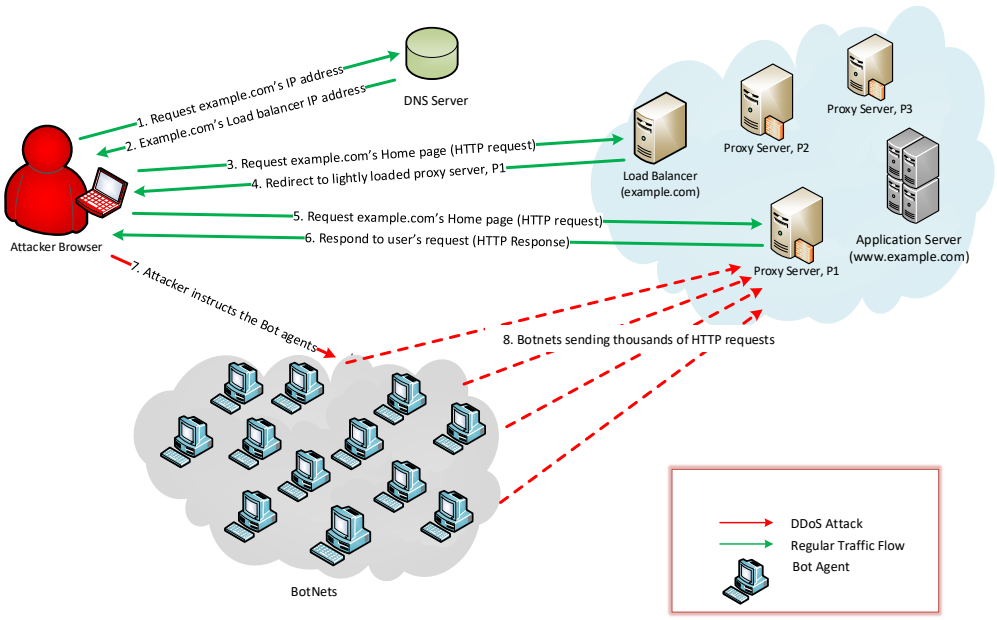

Figure 1. Example of a DDoS attack in proxy-based architecture.

In order to handle such attacks, proxy-based network architectures [6] have been implemented to manage and load-balance incoming traffic by spawning new servers in the event of unexpected rise in network traffic. Proxy-based architectures usually have multiple layers of redirection between the user and the application servers. A Load Balancer (LB) placed between the proxy server [7] and the backend server redirects incoming users to one of the available, lightly loaded proxy servers. Proxy servers hold a copy of the content present in the original servers and process the incoming user request on behalf of the original application server. A proxy server communicates with the original server in the event of missing or outdated information. Also, in case of unexpected rise in incoming traffic, cloud services are used to spawn additional proxy servers to handle the traffic. Thus, organizations do not have to invest a lot for in house proxy servers, but instead pay for the duration of usage. With such an architecture, the attack surface has shifted to proxy servers and DDoS attacks continue to exist.

To understand the limitations of using a proxy-based architecture in handling DDoS attacks, consider a Web service provided by a combination of proxy servers and a backend application server, as shown in Figure 1. Also, consider a client trying to access a Web page www.example.com**.** We now list the sequence of steps that take place:

1. The client types in the URL in the browser
2. The browser resolves the domain name by talking to the Domain Name Server (DNS) and getting an equivalent IP address (that actually corresponds to the Load Balancer's IP address)
3. Next, the browser sends an HTTP request to the LB which then finds a proxy server that is lightly loaded
4. The LB then redirects the user to the assigned proxy server
5. The client then directly talks to the proxy server.

The LB redirects the session to one of the active proxies at random. LB-to-proxy redirection by domain name requires that clients obtain proxy details (IP, port number) by DNS and then contact their proxies directly. Through this process, the attacker learns the IP address of an active proxy. Once the IP address of the proxy server is learnt, a DDoS attack can be launched by the botnets by generating a huge number of packets with the proxy server's IP address as the destination [8].

One of the main reasons for such attacks to happen is due to the attackers being aware of the identity of the application servers (IP address and port numbers) hosting the application – the LB-to-proxy redirection by domain name where the client gets the IP address of the proxy server and is on its own in communicating with the server. Once the IP address of a proxy server is known, the attacker can directly launch an attack using a botnet on that proxy server. To handle this problem, we need to ensure that every user directed to a proxy is done so by the Load Balancer. Thus, we can guarantee that the LB is aware of the number of users per proxy. In such cases, any user request directed to a proxy without contacting an LB would be a possible attacker traffic

 ISBN: 978-1-61208-786-3

The rest of our paper is organized as follows. Related work is discussed in Section II. Our proposed solution is described in Section III followed by the experimental setup and results in Section IV. Future work and Conclusions are discussed in Sections V and VI, respectively.

II. Related Work

Most of the existing solutions in this area focus on (a) Moving Target Defense (MTD), and (b) extending the available resources to support increased user requirements. MTD provides a dynamic environment to periodically shift or change the attack surface thus introducing uncertainty for the attackers, thereby hindering their ability to plan effective attacks.

In [9], Venkatesan et al. identify an attack pattern called proxy-harvesting attack which enables malicious clients to collect information about a large number of proxies before launching a DDoS attack. To mitigate ongoing attacks due to proxy-harvesting attack, the authors propose a static client-to-proxy assignment strategy to isolate compromised clients, thereby reducing the impact of attacks. Each client has a binding to a particular server, which persists even if the client logs outs and logs back in. The main challenge with such a strategy is the overhead of maintaining the assignment/ state information and mapping it every time a user request comes in.

In [10], Jia et al. use cloud platforms to host proxies. Incoming requests are validated by a lookup server and the authorized users are directed to one of the existing proxies. In case of an unexpected rise in traffic targeting them, instances of proxies are created in the cloud, for a short period of time, and the existing users associated with the attacked proxies are distributed among the newly spawned proxies. Random shuffling of users is done before assigning them to the new proxies, thus trying to weed out the illegitimate/attacker's traffic.

Another Web protection service is Moving Target Defense Against Internet Denial of Service Attacks (MOTAG) designed by Wang *et al.* [11], which works by hiding the application server location behind the proxy servers. MOTAG is based on a cloud environment where it decreases the availability of resources to limit the impact of an attack. However, there are down points in MOTAG as it does not handle the situation of overhead associated with instantiating and maintaining new proxies.

In Wood et al. [12], the authors proposed Denial of Service Elusion (DoSE), a cloud-based architecture. In DoSE, each client is associated with a risk value that estimates the chances of a client getting a DoS attack. Each proxy is then defined with an upper bound that it can handle. During the attack, the DoSE redirects the client to proxy servers based on the risk calculation. This is similar to MOTAG, and by maintaining a stage for each client, DoSE limits the proxy numbers used to identify insiders.

In MOVE [13], a subset of network elements and target services accept traffic from a subset of overlay nodes. Once the DDoS attack is mitigated, the target service is moved to a new host. However, in order to make this mechanism work, the solution has to rely a lot on large-scale adoption and network elements. This limits the defense approach that underlies behind the targeted servers.

In spite of the existence of various mitigation techniques, DDoS attacks in proxy-based architecture still continue to exist. One of the main reasons is the overhead involved in either migrating the existing clients or spawning additional resources to handle additional traffic. To overcome the identified challenges, in this paper, we design a client-to-proxy assignment and authentication scheme that finds a lightly loaded proxy and returns the IP address along with the unique ID to the client. The client then talks to the proxy server by exchanging its unique ID. Only the user with a valid ID is allowed to communicate with the proxy. We thus make sure that every user directed to a proxy is authenticated by the LB.

III. Proposed Solution

In this section, we present a unique tag technique that can be used to authenticate if a client is directed by a LB or not.

The DDoS attack exploits the LB-to-proxy redirection scheme i.e., when a client request arrives at the Load Balancer, it returns the IP address of a lightly loaded proxy server P_i to the client. The client then initiates a TCP connection directly with the proxy server P_i. An insider client is an attacker who manages to bypass the authentication system and connect to the proxy server. Once the insider gets some information related to a proxy server such as IP address and port number, the insider in turn shares that information with an external botnet. Thus, an insider client, aware of the IP address of the proxy server, can in turn initiate/launch a DDoS attack targeting the proxy server by directing the attack traffic from distributed bots towards the proxy server's IP address.

One of the possible ways to handle the above discussed attack scenario is ensuring that each client, requesting a TCP connection with an available proxy server, is directed by the Load Balancer.

Let us consider a simple example where Bob wants to request a Web page from www.example.com and, hence, types in the URL in his browser, as shown in Figure 2.

 ISBN: 978-1-61208-786-3

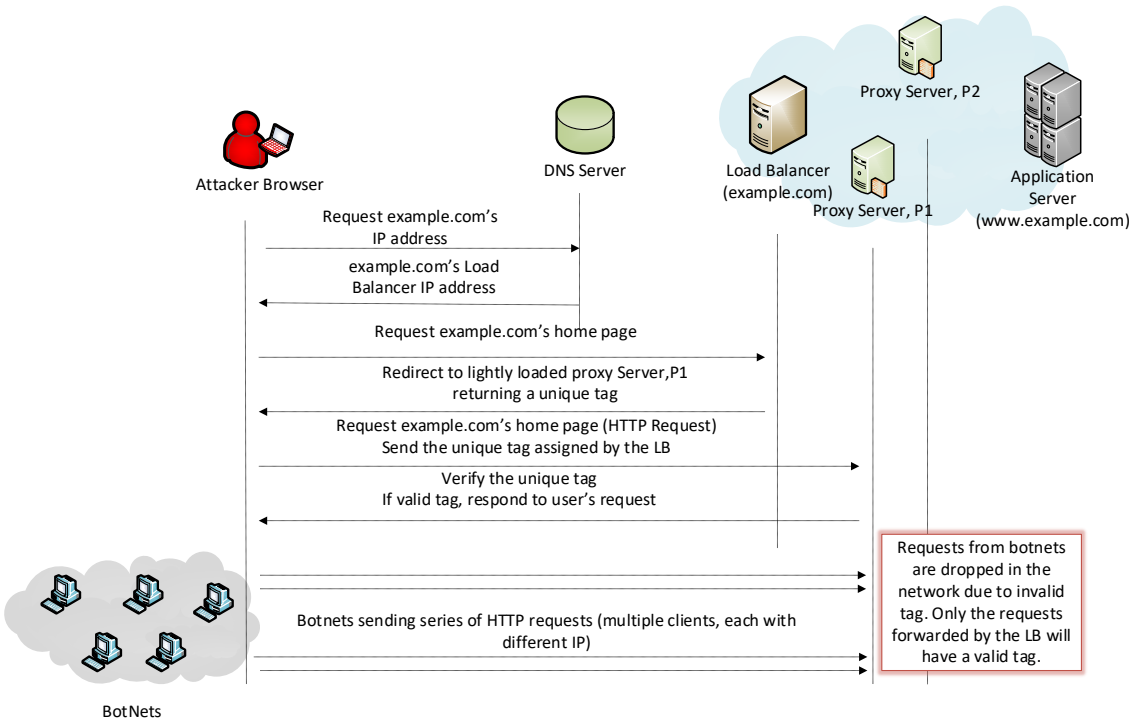

Figure 2. Our proposed solution.

We now list the sequence of steps that takes place:

1. Bob's browser talks to the DNS server requesting for example.com's IP address; DNS returns the IP address corresponding to the LB;

2. Bob's browser then sends a HTTP request to the LB, requesting to access example.com's home page;

3. The LB generates a unique tag and returns it to the user along with the IP address of the proxy. The unique tag is generated as a function of (proxy server, LB, client) IP address and client's port number. To avoid the tag being forged, the tag is encrypted using proxy's public key;

4. The user then sends a HTTP request to the proxy server along with the unique tag assigned to it;

5. The proxy first decodes the unique tag, using its private/secret key and verifies the credentials present in the tag. On successful verification of the client, the proxy sends a corresponding HTTP response; otherwise, the client request/connection is dropped.

We now discuss a few attack scenarios and how our proposed scheme helps in dealing with such attacks.

1. If more than one user arrives at the proxy with the same unique tag, this situation implies that the unique tag was forged and used for another user. In that case, the user IP address and port number are monitored for possible attack traffic.

2. If an attacker manages to find the IP address of another proxy server, through another client, it is possible that the attacker might direct all the clients with a unique tag ID towards a single server. To handle this situation, the function to generate tags is dependent on the client's IP address, LB's IP address, and proxy server's IP address. Thus, only when the client arrives at the right proxy server, its request will be serviced. Thus, we make sure that the flow of traffic is regulated.

IV. Experimental Setup And Simulation Results

For our experiments, we simulate a simple network using socket programming in Java. Each component (LB, clients, proxy servers, and application servers) is a Java class running on the localhost i.e., 127.0.0.1. For our experimental setup, we have a LB that processes incoming requests from the client, and four proxy servers, which are, in turn, connected to the application servers.

 ISBN: 978-1-61208-786-3

Our implementation works as follows:

1. Initially, when a user request arrives at the LB, it generates a unique tag which is a function of Source/client, LB, and proxy server's IP address and encodes it into a secret tag.
2. The user request is then redirected by the LB using HTTP response 302 Found. The secret tag generated is placed as a part of the Location field in the HTTP response, along with the proxy server's IP address.
3. The redirected user request then arrives at the proxy server, where the server first extracts the unique tag and verifies the IP addresses.
4. If the secret tag is a valid one, the user request is processed; otherwise, the request is dropped. The requests will be dropped due to invalid/missing secret tags.

We simulated about 50 valid user requests by hosting client programs and sending simple HTTP GET requests for a valid document available at the Application server. Additionally, we simulated about 30 requests which were mainly attack traffic. The way we simulated the attack traffic is described below. We assume user-1 is the attacker.

1. A valid user request is sent to the LB from user-1;
2. The LB then finds a proxy server with the lowest load and returns the IP address of the proxy with a unique tag that contains user-1's IP address and port number;
3. User-1 then generates 30 requests directed to the proxy servers chosen in step-2;

In terms of performance evaluation, our primitive implementation resulted in a uniform load distribution, each proxy server having an average of about 7 users. Additionally, our proposed model was able to detect malicious/ illegitimate traffic successfully. All the valid requests were successfully redirected by the LB to available proxy servers such that the load on each proxy server was close evenly distributed. In case of attack traffic, the user requests with unique valid tags were successfully authenticated and processed by proxy servers, whereas the attack traffic without valid tags was dropped by the proxy servers.

V. Future Work

In our proposed solution, the encryption of the tag using secret key requires both the LB and the proxy server to exchange a key periodically that will be used to protect the transferred data. The encryption process may add a little bit of overhead during the initial key exchange, converting plain text to ciphertext at the LB's end and vice versa at the proxy server's end. We intend to study their effects on performance in terms of the time take to direct an incoming client to one of the proxy servers, the time it takes to process the client's HTTP request, as well as the number of false positives and negatives during DDoS detection. Additionally, we would like to compare our work with available solutions and current commercial state.

VI. Conclusions

In this paper, we propose an authentication mechanism to detect and prevent DDoS attacks in a proxy-based architecture. Our proposed technique ensures that each client request arriving at a proxy server is directed by the Load Balancer. A proxy server will only service those clients that are originally redirected by the Load Balancer. Since the Load Balancer's job is to uniformly distribute the incoming client traffic among existing proxy servers, the chances of a DDoS attack due to a huge amount of incoming traffic is mitigated. Thus, the DDoS attacks caused due to botnets can be easily handled.

References

[1] M. Sysel and O. Doležal, "An Educational HTTP Proxy Server," In Procedia Engineering, vol. 69, 2014, pp. 128–132.

[2] M. Antonakakis et al., "Understanding the Mirai Botnet," USENIX Security Symposium, 2017, pp. 1093–1110.

[3] N. Woolf, "DDoS Attack That Disrupted Internet Was Largest of Its Kind in History, Experts Say," in The Guardian, Guardian News and Media, 26 Oct. 2016, Retrieved from: www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet [accesses Oct., 2016].

[4] S. Mahrach and A. Haqiq, "DDoS Flooding Attack Mitigation in Software Defined Networks," in International Journal of Advanced Computer Science and Applications, vol. 11, 2020.

[5] K. S. Vanitha, S. V. UMA, and S. K. Mahidhar, "Distributed denial of service: Attack techniques and mitigation," International Conference on Circuits, Controls, and Communications (CCUBE), 2017, pp. 226-231, doi: 10.1109/CCUBE.2017.8394146.

[6] S. Kumar, P. Tiwari, and M. Zymbler, "Internet of Things is a revolutionary approach for future technology enhancement: a review," Journal of Big Data, vol. 6, 2019 doi: 10.1186/s40537-019-0268-2

[7] A. Baptiste, "Use a Load Balancer as a First Row of Defense Against DDOS," in Haproxy, Feb. 27, 2012, Retrieved from: www.haproxy.com/blog/use-a-load-balancer-as-a-first-row-of-defense-against-ddos/ [accesses Feb., 2012]

[8] P. Jeff, J. Blankenship, and A. Cser, "How The Mirai Botnet is Fueling Today's Largest and Most Crippling DDoS Attacks," in Akamai Forrester Research 24 Oct. 2016, Retrieved from: https://www.akamai.com/uk/en/multimedia/documents/white-paper/akamai-mirai-botnet-and-attacks-against-dns-servers-white-paper.pdf [accesses Oct., 2016]

[9] S. Venkatesan, M. Albanese, K. Amin, S. Jajodia, and M. Wright, "A Moving Target Defense Approach to Mitigate DDoS Attacks against Proxy-Based Architectures," in Proceedings of the Communications and Network Security (CNS), 2016, pp. 198-206.

[10] Q. Jia, H. Wang, D. Fleck, F. Li, A. Stavrou, and W. Powell, "Catch me if you can: A cloud-enabled DDoS defense," in Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2014, pp. 264 – 275.

[11] H. Wang, Q. Jia, D. Fleck, W. Powell, F. Li, and A. Stavrou, "A Moving Target DDoS Defense Mechanism," Computer Communications, vol. 46, June 2014, pp. 10 – 21.

[12] P. Wood, C. Gutierrez, and S. Bagchi, "Denial of Service Elusion (DoSE): Keeping clients connected for less," in

Proceedings of the 34th IEEE Symposium on Reliable Distributed Systems (SRDS), 2015, pp. 94–103.

[13] A. Stavrou, A. D. Keromytis, J. Nieh, V. Misra, and D. Rubenstein, “MOVE: An end-to-end solution to network denial of service,” in Proceedings of the Network and Distributed System Security Symposium (NDSS), February 2005, pp. 81–96.