Exploring DDoS Mechanisms

Bruno de Souza Neves, Filipe Mourão Leite, Lucas da Silva Jorge, Rahyan Azin Gondin Paiva, Juliana de Melo Bezerra and Vitor Venceslau Curtis
Department of Computer Science, ITA, São José dos Campos, Brazil

Keywords: DDoS, Botnet, Election, Network Topology.

Abstract: The concern about cyber security has attracted attention by organizations and public services over the last few years due to the importance of confidentiality, integrity, and availability of services and sensitive data. Many recent episodes of cyber attacks causing strong impact have been performed using extremely simple techniques and taking advantage of the hidden weaknesses of current systems. This article has the main purpose to motivate the academy to mitigate unexpected potential threats from the attacker's perspective, exposing the latent weakness of current systems, since the majority of such papers focus only on the defense perspective. We analyze the potential impact of Denial of Service (DoS), a very popular cyber attack that affects the availability of a victim server, through different mechanisms and topologies. Specifically, we simulate and analyze the impact of Distributed DoS (DDoS) using the List and Binary Tree topologies along with Continuous or Pulsating stream of requests. In order to improve the potential of the analyzed DDoS methods, we also introduce a new technique to protect them against mitigation from security systems.

1 INTRODUCTION

Over the last decade, we have seen technologies as streaming, deep learning, IoT, blockchain and big data disrupting traditional business and enabling a prolific market of network-based systems where data are the principal value. In this context, secure and reliable systems are very important to not tarnish the image of a company and crucial in critical systems and public services.

A good example of potential damage a network cyber attack could lead was recently revealed by (McCallie et al., 2011). They expose the fragility of the Automatic Dependent Surveillance-Broadcast (ADS-B) system of the global air traffic control. Using about $1,000 worth of radio equipment, a hacker could simply flood the air traffic control system with as many fake airplanes as it wants.

In 2016, Mirai botnet attacked the Internet Service Provider (ISP) for sites such as Twitter, Amazon, PayPal, Spotify, Netflix, and others, making them unreachable for several hours. In such attacks, a worm propagates through networks and systems taking control of poorly protected IoT and embedded devices such as IP cameras, thermostats, Wi-Fi enabled clocks and washing machines (Kolias et al., 2017). At its peak, Mirai infected over 600,000 vulnerable devices and was able to attack the OVH services with a volume of network traffic around 1Tbps (Antonakakis et al., 2017).

These episodes expose considerable negligence by companies in the past years regarding security, which now reflects as latent threats to computer systems. Forecastings estimate that IoT and small devices will represent more than 75% of the global Internet reaching 10 billion in 2020 and representing a true potential threat specially because of new connected solutions with poor security pop up every day by small business and startups, (Rose et al., 2015), (Columbus, 2018).

With this scenario, we propose that the academy encourage more research papers from the attacker's perspective rather than the defense to force the exposure of current hidden weaknesses of computer systems as a measure of prevention from real potential attacks to organization and states.

According to (Schatz et al., 2017), the current terminology to discuss security aspects of digital devices and information is cyber security and it is basically defined as the actions and technologies followed by organizations and states to protect confidentiality, integrity, and availability of data and assets used in cyberspace.

Among the cyber threats, the Denial-of-Service (DoS) is a very easy and common kind of cyber at-

Neves, B., Leite, F., Jorge, L., Paiva, R., Bezerra, J. and Curtis, V.
Exploring DDoS Mechanisms.
DOI: 10.5220/0007835304600467
In *Proceedings of the 14th International Conference on Software Technologies (ICSOFT 2019)*, pages 460-467
ISBN: 978-989-758-379-7

tack that violates the availability of systems. Besides it looks harmless at first, as cited by the examples above, it can cause severe damages to organizations and public services. Furthermore, due to the contemporary importance of confidentiality and value of data, cyber attacks can cause an important devaluation of brands by inducing distrust about the integrity and confidentiality of personal data or services, becoming even a tool to commit financial crimes by influencing the stock markets.

In this article, we motivate the attacker's philosophy of cybersecurity by analyzing the performance of some Distributed DoS (DDoS), a non-trivial kind of DoS, as a way to analyze the potential damage a real scenario powered by the growth of IoT devices could cause.

First, we present some basic concepts of DoS and DDoS in section 2, followed by a common taxonomy of the main methods of DDoS based on some behavior mechanisms. Section 3 describes how to simulate the main methods of DDoS introduced in the previous section and the limitations of these simulations. Section 4 discusses the results of the simulations and the potential of each mechanism. In section 5, we propose a new technique difficulting DDoS mitigation by protecting the attacking network and, finally, 6 cites the main conclusions of this work.

2 BASIC CONCEPTS OF DOS AND TAXONOMY OF DDOS

DoS is a cyber-attack in which the perpetrator seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host connected to the Internet (Soltanian and Amiri, 2016), and it can be achieved by different methods.

Usually, the most common method of attack occurs when the hacker floods a network server or resource with superfluous or incorrect requests in an attempt to decrease its availability to attend legitimate requests and blocking all users at once. One example of such attack is known as SYN flood, where one sends a request to connect but never completes the connection through a three-way handshake. The incomplete handshake leaves the connected port in an occupied status and unavailable for further requests.

Many of these DoS attacks are easy to identify and block by only checking the source of the requests. Because of this, hackers usually use DDoS, which implements distributed computing to coordenate multiple attackers from different sources to send malicious traffic to a targeted server.

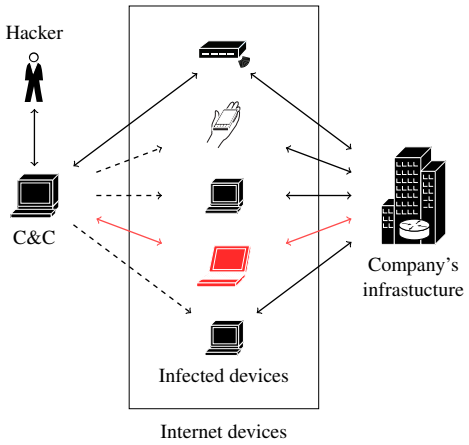


Figure 1: Arquitecture example of a DDoS botnet with one zombie, in red, blocked by the target firewall.

A DDoS starts with the hacker first spreading malicious softwares called malware with the intention to infect an army of Internet-connected devices, named zombie computers, with backdoors running in background and waiting for commands from a Command and Control (C&C) server, as shown in Figure 1, configuring a network of zombies called botnet.

After many devices infected, the hacker initiates a DDoS through the C&C, which is responsible to spread the attack command over the botnet, and each zombie performs the attack by requesting some service over the network to the target infrastructure. It makes the defense much more difficult since blocking one IP, see Figure 1, has almost no effect because of the massive amount of attacking points spread over the Internet.

DDoS is also much more powerful than a simple DoS because it multiplies the capacity of sending fake request to the target, the massive amount of available resources enables requests with signatures very close to genuine requests, and it can be implemented with no central point of C&C, making the interruption of an attack much more difficult.

In the literature, there are some taxonomies trying to capture all the methods involving DDoS. (Mirkovic and Reiher, 2004) and (Bhardwaj et al., 2016) classify DDoS attacks by the following main categories: Degree of Automation (DA), Exploited Weakness to Deny Service (EW), Source Address Validity (SAV), Attack Rate Dynamics (ARD), Possibility of Characterization (PC), Persistence of Agent Set (PAS), Victim Type (VT) and Impact on the Victim (IV), as described by Figure 2. For more information about the details of this taxonomy and the characteristics of each component, please, see (Mirkovic and Reiher, 2004) and (Bhardwaj et al., 2016).

According to this classification, our proposed

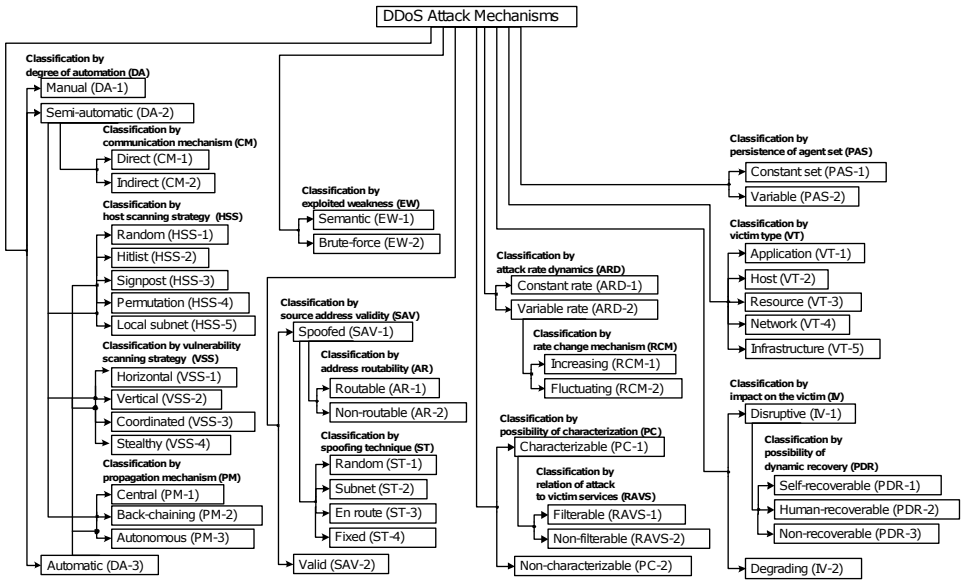


Figure 2: Taxonomy of DDoS attacks.

methodology has the following characteristics:

- Semi-Automatic (DA): the attack is initiated by the user command, while all the other processes are automatic;
- Brute Force (EW): it sends a high volume of seemingly legitimate requests to exhaust the victim's capacity;
- Valid Source Address (SAV): the attacker does not use a spoofed source address;
- Constant or Variable Rate (ARD): both dynamics are tested;
- Non-Characterizable (PC): no specific protocol or application from the victim was chosen to be targeted. The attack only intends to consume network bandwidth;
- Constant Agent Set (PA): when a node is infected, it becomes a permanent participant in each of the subsequent attacks;
- Network Attack (VT): the attacks mean to exhaust the bandwidth of a target;
- Disruptive (IV): the attack has the object to disrupt the entire target's resources, even though sometimes it cannot accomplish it.

Among these characteristics, the Degree of Automation and Attack Rate Dynamics are crucial in our experiments once our goal is to measure the efficiency of a botnet. Thus, the following subsections 2.1 and 2.2 respectively describe in more details the mechanisms of DA and ARD analyzed in our simulations.

2.1 Degree of Automation

The infection of new zombies, the dynamic management of the network, and the control of a botnet may be implemented in different degree of automation (Bhardwaj et al., 2016):

- Automatic: zombie nodes autonomously organize themselves, including a pre-programmed attack pattern;
- Manual: the hacker starts all the commands: attack, network management and infection;
- Semi-automatic: some degree of automation.

In this work, we preferred the semi-automatic degree of automation once the automatic restricts the flexibility of attack and the manual exposes the botnet network with many control messages. The semi-automatic approach is better once it has all the benefits of common distributed systems: no-central server, resilient to failures, scalable, etc.

Our semi-automatic solution implements a distributed botnet system that dynamically adapts the network, the topology of zombie nodes, once they become available or not due to a new successful infection or mitigation by a security system. Only the start of an attack command is kept manual in order to have more control of the experiments, while the dissemination of any command in the botnet continues autonomously.

Botnets can be formed in different network structures, topologies, for which the most common are:

- Star topology: there is a C&C server with the bots organized around it;

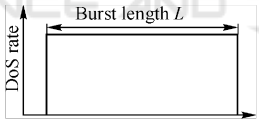
- Hierarchical topology: bots organized in layers of C&C servers;
- Random topology: Peer-to-Peer (P2P) communication among bots.

The star topology was chosen to facilitate the implementation of the semi-automatic attack, since there is a central server that can receive manual input from a user and then transmit a flow of commands in the direction from the root to leaves.

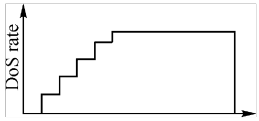
A weakness of this topology is the possibility of a security system to explore the center of each start. Using this vulnerability, it is possible to reach the C&C and make it unavailable. In order to avoid this type of mitigation and still preserve the manual control of the attack, in section 5, we propose a new method of automation which controls the botnet with resilience, where zombies have only partial information about the topology.

2.2 Attack Rate Dynamics

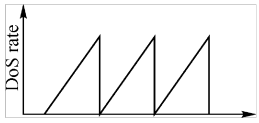
A DDoS can also follow different dynamics depending on the duration, form, and distribution of the traffic in the stream of superfluous requests, where the more similar its signature is to a genuine request, the more difficult it is to detect the DDoS. A taxonomy with two main classes of ARD patterns are described in (Liu et al., 2012).



(a) Constant rate FDoS stream



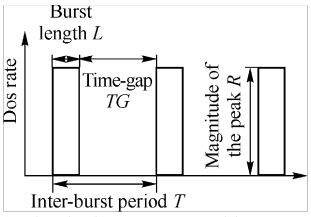
(b) Increasing rate FDoS stream



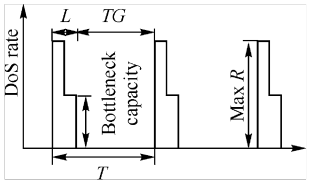
(c) Fluctuating rate FDoS stream

Figure 3: Typical FDoS waveforms.

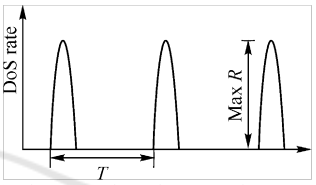
Flood DoS (FDoS), as showed in Figure 3, has three typical types of a waveform based on the ARD: classical constant rate flood, increasing rate flood, fluctuating rate flood. In constant rate, the botnet attacks the target with full force; in increasing rate flood, the botnet gradually increases the attack rate resulting to slow exhaustion of victim's resources; and in fluctuating rate, the botnet performs any other waveform, including occasional changes.



(a) Classical square-wave ldos stream



(b) Two rate square-wave ldos



(c) Sine waveform low rate dos stream

Figure 4: Typical LDoS waveforms.

Low Rate DoS (LDos) is similar to fluctuating rate flood but it sends just one short burst, usually a square wave pattern, every each T period. For large botnets, T may even be long in order to difficult the detection of the malicious attack. The Figure 4 presents some typical different patterns of LDoS.

In this work, we analyze two main patters of a ARD: a continuous stream and a pulse waveform. The first is very similar to the constant rate FDoS and the second is closely related to LDoS pattern. However, it is important to note that the propagation delay of a command from the C&C into the botnet may drastically change the waveform, resulting in interesting patterns.

3 SIMULATIONS OF DDOS

In this work, the two common mechanisms of ARD described in Section 2.2 are analyzed, each using two network topologies: a tree-based, where each node in the botnet has some children, and list-based network, where each node can forward an attack only to one next node.

In order to simulate the experiments, we implemented programs in GoLang supported by lightweight threads to achieve a high scale for the bot-

net simulation. Basically, the simulations are composed by three programs:

- Server: It is basically the C&C server, responsible for the botnet commands and for tracking when a node first joined the botnet;
- Client: It is the zombie node, a node in a botnet network, capable of autonomously discovering other zombies and handling attacks;
- Targeted Server: It represents the target infrastructure and stores the attack data for further analysis.

All programs communicate with each other by UDP packets and the Client implements distributed solutions to handle the network topology autonomously. All the information about the simulation is retrieved from logs and the packets exchanged among the nodes of the network. More specifically, each packet contains:

- Sent to the Targeted Server: an identification of the sender, a timestamp of a sent packet, and a sequence number to help identifying lost packets;
- Ordering a attack (to zombies): the Targeted Server identification, the type of the attack, and a timestamp identifying the initial attack time in case of a pulse DDoS pattern;
- Regarding the botnet construction: identification of the new Client (zombie).

Using data of the packages sent to the Targeted Server it is possible to acquire information regarding the packet loss, packet delay and transmition delay throughout the botnet. We use these data to implement two kinds of attack patterns: constant (Clients begin the attack as soon as they receive the message), and the pulse-like (a timestamp states when the Clients should start the attack).

Since the experiment was not performed in a real network, real computers with their latencies and bandwidth, we had to simulate these characteristics by controlling the rate of the package exchanges among the nodes of the network.

The latencies are simulated by implementing two basic botnet topologies: a list topology and a random binary tree topology. In both cases, the addition of a new zombie is decided by the zombies already set in the network: it greedily accepts a new zombie as its own child (two in the case of binary tree and one for list) or ramdonly choose one of its children to delegate the addition in a recursive way. The goal is to simulate a balanced tree that minimizes its depth (information delay among bots) and the message passing, and further compare it to a topology with high latency, i.e., the list-based topology.

All the codes are available open-source for further researches. Please, contact the authors for getting access to it.

4 RESULTS

This section presents the main results of our simulations for the tree-based and list-based topologies using continuous stream and pulse wave pattern of DDoS attacks. From the logs obtained by the simulation, the most relevant measurements extracted are the package loss and the attack propagation delay.

The tests were performed in different machines configurations and the results were similar. In the following sections, we preferred to show the results for a single regular Windows PC with the following configuration and tools: Windows 10 O.S., Intel Core i7 processor, 8GB(RAM). The amount of lightweight threads is described in each experiment.

4.1 Attack Propagation Delay

In a real scenario, a central server of a star topology usually contaminates zombies around it in a tree-like structure. Thus, in order to simulate this approach, we implemented the system with a binary tree topology. Furthermore, to observe the effects of having a large number of levels in the tree architecture with a reasonable number of nodes, we use a botnet with list structure, which in practice is unlikely to happen but it is enough to observe results.

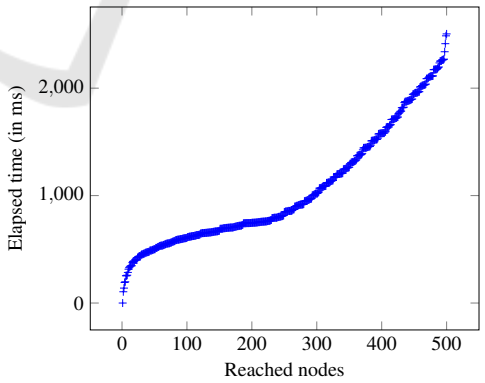


Figure 5: Propagation delay within a botnet with 500 nodes using constant DDoS in binary tree topology.

During the contamination phase, for a similar number of nodes, the list-based topology presents a bigger number of layers than a tree-based topology, implying in a larger delay of propagation through the botnet. Because of this, we use 100 nodes for the list implementations. For the binary tree topology, since

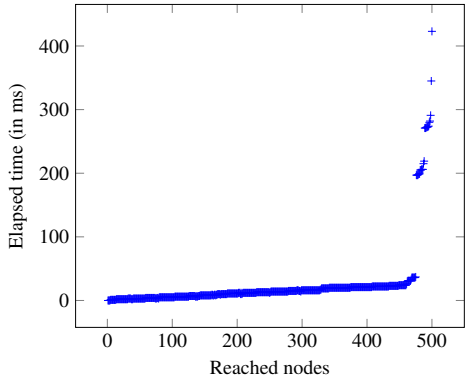


Figure 6: Propagation delay within a botnet with 500 nodes using pulse DDoS in binary tree topology.

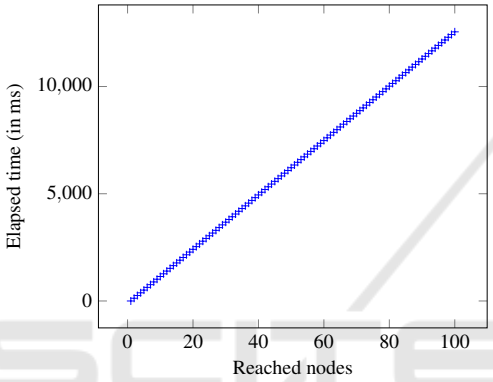


Figure 7: Propagation delay within a botnet with 100 nodes using constant DDoS in list topology.

the height tends to be proportional to $\log_2 n$, we use 500 nodes in the simulation to get more levels and make the effect of the topology clear in the results. The results are presented in Figures 5-8.

Using pulse DDoS mechanism, Figure 6 and Figure 8, it can be noticed that almost all nodes attack at the same time, delay near 0ms, except for outliers that do not attack immediately for some unexpected

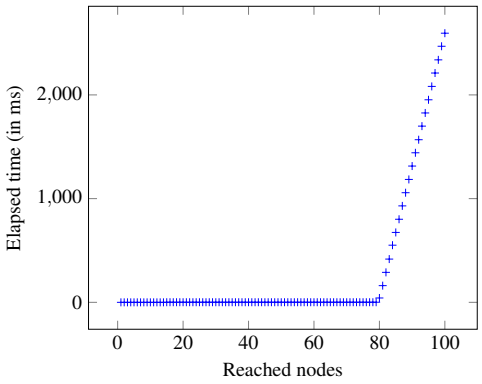


Figure 8: Propagation delay within a botnet with 100 nodes using pulse DDoS in list topology.

fail or oscillation, a phenomena also present in a real network.

The graphs of the constant DDoS mechanism, Figure 5 and Figure 7, show how the nodes are gradually mobilized by the attack wave, exposing the hierarchy of a network with many layers. Particularly, the propagation delay using constant DDoS and list topology, Figure 7, results in perfect straight line once the information in the list is passed from node to node with a similar delay.

4.2 Package Loss in Target Server

The graphs on Figures 9-12 show that the binary tree topology is much more effective in damaging the traffic of the target server. Even considering a margin on a simulation where a tree-based topology has 5 times more node than a list-based topology, 500 nodes versus 100 nodes, the difference of the results is much bigger than this proportion.

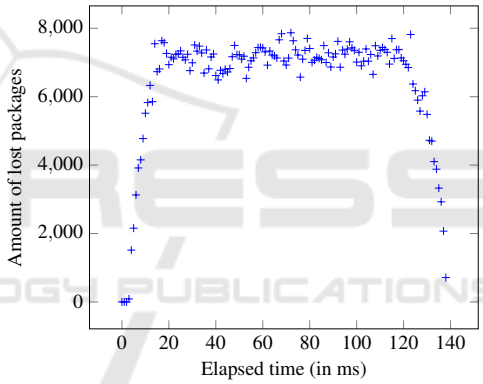


Figure 9: Package loss for each ten thousand packages using constant DDoS in Binary Tree topology.

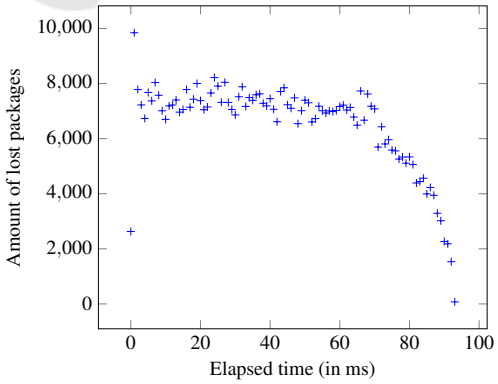


Figure 10: Package loss for each ten thousand packages using pulse DDoS in Binary Tree topology.

This can be interpreted as a consequence of the small delay in propagating commands from the C&C

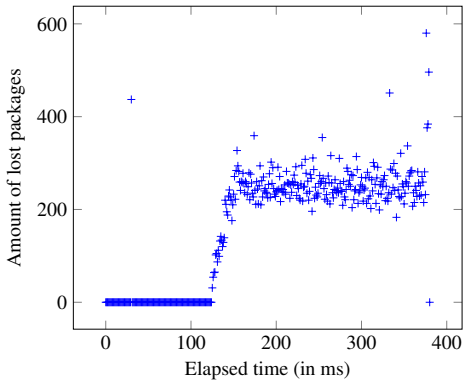


Figure 11: Package loss for each ten thousand packages using constant DDoS in List topology.

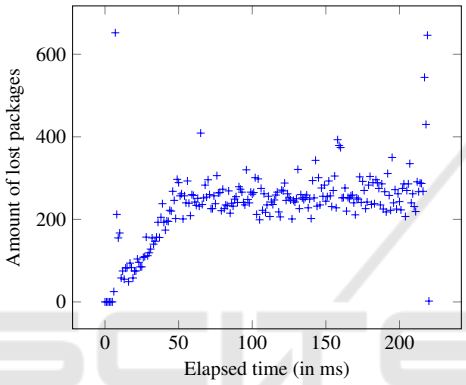


Figure 12: Package loss for each ten thousand packages using pulse DDoS in List topology.

to far nodes in the botnet for a smaller number of levels. In the list, even though the DDoS algorithm orders that all nodes attack at the same time, the delay of the propagation makes the attacks to reach the target asynchronously resulting in a very ineffective approach.

As it can be seen in both topologies, Figure 9 and Figure 11, respectively, the tree-based and list-based topologies, the constant rate DDoS presents a considerable delay before achieving a stable rate of package loss. It is the time necessary to mobilize all the nodes in the botnet.

On the other hand, Figure 10 and Figure 12 show that the pulse DDoS presents a stable rate since the first time of the attack, given that the C&C orders that all nodes attack at the same clock time, and resulting in a more powerful capacity for using it against big infrastructures.

Thus, we may conclude that that the shorter the attack propagation delay is, the worse are the impacts. As it can be seen in Figure 10 and Figure 12, it is very likely that no action could be taken before the server is down, the worse case scenario. In constant attacks patterns, like the ones from the Figures 9 and 11, it is possible to identify the increasing workload and take action to protect the servers before it is too late.

5 PROTECTION AGAINST MITIGATION

In a real botnet over the Internet, zombie nodes may be turned off at any time. As the transmission of the C&C commands is forwarded node by node on the topology, it may brake the botnet. Usually, botnets solve it by implementing some distributed solution to autonomously adapt the topology on such cases.

Security systems may take advantage of this situation by infiltrating a zombie spy in the botnet and blocking the other nodes in such a way to push it as close as possible to the root of the tree topology. Using this strategy, the invader could raise hierarchically in the topology and get a huge part of the tree as its descendant. Since the command flow occurs from the root to the leaves, the spy can stop sending malicious traffic to this part of the botnet or sabotage it.

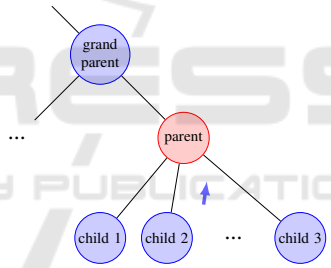


Figure 13: Child detects the absence of it failed parent.

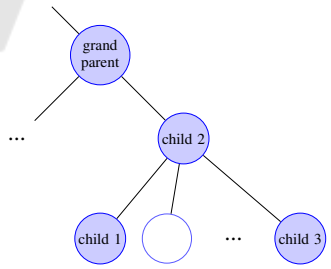


Figure 14: Oldest child (2) replaces the missing parent.

Next, we describe a solution to protect the botnet against this kind of security systems. If a node detects that its parent is missing, it starts an election process to decide a new one. In the adopted strategy, exemplified in the Figures 13 and 14, the oldest child of the missing parent takes its place.

The timestamp on the system clock, when the zombie joined the botnet, is used to compare the zombie nodes and to decide who is the oldest brother. This

is sufficient to avoid an invader from forging its own time by pretending to be older in order to substitute a missing parent. Since the malware in a real system could take hours or even days to spread, the invader has to wait for a time of the same order of magnitude to ascend a substantial level in the hierarchy. Meanwhile, the other branches of the tree grow independently of what the security system is doing in the spy branch. This is more than enough for the botnet to make substantial damage to the server.

It is important to notice that, in our botnet configuration, each node is only able to get information about its adjacent nodes: child, parent, and some close nodes (siblings and grandparent) due to the election procedure. This prevents a possible attacker to have easy access to the control server and allows a semi-automatic Degree of Automation.

Another approach to avoid the vulnerability of having a centralized server taken down is to change the hierarchy to peer-to-peer, in which each node would be exposed only to its adjacent bots and every node can be a command center. However, this would forbid the manual control by the Server: time, rate and the dynamic. Thus, the implemented topology is similar to a P2P in the way nodes join the botnet and communicate with neighbors, but it is similar to a hierarchical network in the way command flows throughout the level of the topology.

6 CONCLUSION

The simulation of DDoS mechanisms implemented in this work achieved interesting results preserving the functionality of a real DDoS and clarifying the dissemination of information within a botnet along with its interacting behavior.

The list-based implementation exposed the differences in each ARD mechanism, since it generated many levels of hierarchy in the topology using a relatively small number of nodes. Thus, phenomena as the network propagation delay can be noticed easier: the time to mobilize all bots in the network is more expressive and the two mechanisms of attack generate very different results.

In terms of package loss in the targeted system, both mechanisms (Continuous and Pulsating) generated similar outputs. This is expected for a limited target infrastructure and it can be interpreted as the botnet having more network bandwidth capacity than the target.

We also proposed a new election strategy to manage the dynamic structure of the botnet network, when zombie nodes detect failed parents, improving the security against mitigation from security systems. The proposed method theoretically meets the objective of preventing the botnet mitigation by outside invaders. However, since the simulation of a real scenario of security system attacking botnets is not trivial, we preferred to explore such simulations in future projects.

As DDoS from the attacker perspective is not a common topic in the academia, we expect that this work can be used as material to new computer network and distributed system courses in order to improve the security discussions.

REFERENCES

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., and Zhou, Y. (2017). Understanding the mirai botnet. In *26th USENIX Security Symposium*, pages 1093–1110, Vancouver, BC.

Bhardwaj, A., Subrahmanyam, G. V. B., Avasthi, V., Sastry, H., and Goundar, S. (2016). DDoS attacks, new DDoS taxonomy and mitigation solutions - A survey. In *2016 Inter. Conf. on Signal Proc., Commun., Power and Embedded System (SCOPES)*, pages 793–798. IEEE.

Columbus, L. (2018). 2018 Roundup of internet of things forecasts and market estimates. *Forbes*.

Kolias, C., Kambourakis, G., Stavrou, A., and Voas, J. (2017). DDoS in the IoT: Mirai and other botnets. *Computer*, 50(7):80–84.

Liu, X., Cheng, G., Li, Q., and Zhang, M. (2012). A comparative study on flood DoS and low-rate DoS attacks. *The Journal of China Universities of Posts and Telecommunications*, 19:116–121.

McCallie, D., Butts, J., and Mills, R. (2011). Security analysis of the ADS-B implementation in the next generation air transportation system. *International l Journal of Critical Infrastructure Protection*, 4(2):78–87.

Mirkovic, J. and Reiher, P. (2004). A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53.

Rose, K., Eldridge, S., and Chapin, L. (2015). The internet of things: An overview. *The Internet Society (ISOC)*, pages 1–50.

Schatz, D., Bashroush, R., and Wall, J. (2017). Towards a more representative definition of cyber security. *Journal of Digital Forensics, Security and Law*, 12(2):8.

Soltanian, M. R. K. and Amiri, I. S. (2016). *Theoretical and Experimental Methods for Defending Against DDoS Attacks*. Syngress, first edition.