DDoS Network Flow Forensics Analyser

Cristian Turetta, and Andrea Perazzoli

Abstract—Distributed denial-of-service (DDoS) is a rapidly growing problem. The multitude and variety of both the attacks and the defense approaches is overwhelming. This report introduces ...

1 Introduction

Denial of Service (DoS) attack is launched to make an internet resource unavailable often by overwhelming the victim with a large number of requests. DoS attacks can be categorized on the basis of single source and multi source. Multi source attacks are called distributed Dos or DDoS attacks [1].

There are various type of tool in order to detect and deal with DDoS attacks, these tools can be applied in real time, such as intrusion detection systems (IDS) or by analysing network flow records offline doing a forensics analysis, which is our focus. Computing forensics analysis over a recorded network flow can be useful, we may understand if someone is trying to flood a network to get a denial of service and eventually recognise it. Evidence of such intrusions is required in case the affected wants to pursue the court and legal action is to be taken against the adversary.

Forensics investigations are not trivial to accomplish and often done manually, this because the attacker can mask its attempts by mixing legitimate requests with malicious ones.

DDoS attacks aim to compromise the availability of a system or a network, the attack is launched by the adversary which has take control over bots, compromised machines connected to internet, that sends several requests to the victim and overwhelm it with large amount of traffic. This creates a bottleneck and the victim can no further deal with this traffic denying service to them.

During DDoS attacks, the log files swell up to huge sizes, these log files if analysed properly and effectively can help detect and recover from a DDoS attack [1]. Log files can take a long time if processed through conventional means thus we decide to use big data's tool and framework in order to get a faster processing and investigation.

In this report we present our tool which uses *Pig-latin* script embedded into a *Python* program that can be used to analyse network log file, `pcap` format [2], and returns statistical information about the recorded traffic. In Section 2 we present the statistical tool used in our analysis. In Section 3 we present the project structure and implementation. In Section 4 we focus on analysis results. In Section 5 we discuss the performance of our tool in terms of computational time and resources. Section 6 concludes the report with our considerations.

References

[1] Rana Khattak, Shehar Bano, Shujaat Hussain, Zahid Anwar. *DOFUR: DDoS Forensics Using mapReduce*. Frontiers of Information Technology, 2011.

[2] Wireshark Wiki, Development. Last access *May 15 2019*. https://wiki.wireshark.org/Development/LibpcapFileFormat

2 Project structure and implementation

Our project and its source code is freely downloadable on Github. Here, we focused on the UDP flood D(D)oS analysis of pcap records: the goal is to point out good and evil users given a pcap network sniff file. The project primarly consists in two executable Python 3 scripts:

- *DDoSAnalysis.py*: this script can be used in two different modes, depending on the line parameters used.
  - *-g dataset_name n_members n_lines n_attackers* Generates a random, bogus dataset in the current working directory with the name specified in the second argument, alongside with the number of normal network users specified in *n_members* argument, the dimension of the dataset (in lines) and the number of infected machines. At the end of the generation process, it copies the dataset into the Hadoop File System. We assumed that Hadoop is installed and a folder tree under `hdfs://user/your_user/project/input` exists.
  - *-a dataset_name* Begin the analysis of the dataset *dataset_name* using a Pig script. In order to work, the dataset must have been previously copied into the Hadoop input folder, which automatically happens if the dataset is generated using *-g* option. It saves the elaborated dataset under `outputs/dataset_name` with an image consisting of a plot of every agent average velocity (bps).
  - *-ga dataset_name n_members n_lines n_attackers* Launches both the generation and the analysis

  The script also automatically records infos about performance timing under `PerformanceHistory.csv`.

- *PerformanceAnalyser.py*: this script is used to automatically plot all the infos stored under `PerformanceHistory.csv`. It supports two modes:
  - *-a img_name*
    stores a plot under the current working directory named *img_name* of analysis statistics (history of dataset analyzed and time elapsed)
  - *-g img_name*
    stores a plot under the current working directory named *img_name* of dataset generation statistics (history of dataset generated and time elapsed)

  PerformanceAnalyser.py also exposes a method used as a wrapper to call the generation and analysis routines, using a **CProfile** python module to gain time statistics.

The first two mentioned scripts are the user interface of our tool. However, we have other core scripts which make the generation/analysis possible:

- *DatasetGenerator.py*: it contains the core generation routine of datasets. It generates a pool of innocent IPs and an attackers' one, then it fills line-by-line the dataset with random and bogus informations, extracting random users and attackers.
- *Evaluator.py*: it exposes the main routine which processes the Pig script output. It computes the mean velocity of all users, and then produces a plot consisting of the velocity of every single user, represented with a blue line, previously calculated by the Pig script (*udpfloodpcap.pig*) and the mean velocity of all users, represented with a red line. The data scientist could distinguish between evil and good users just looking at the deviation from the average.
- *udpfloodpcap.pig*: calculates the mean velocity of every machine given a dataset in input