Batch-processing of Cowrie log files

Bachelor's thesis proposal

Dominic Rudigier
Dominic.Rudigier@student.uibk.ac.at

11832156

April 26, 2021

1 Motivation

A honeypot is a sacrificial computer system which is intended to attract potential hackers. It mimics a target and is built safely to gain information about how a attack was intended to be executed. Once connected to the internet the honeypot generates log files for all events, but the gathered information is hard to handle for a research honeypot. A research honeypot is built to extract as much information about the specific methods and tactics used as possible, unlike a production honeypot which is used by businesses to improve their overall state of security.

One might ask why there is a need for additional tools as the quantity of honeypots and related tools is already slightly overwhelming.[1]

There are multiple tools available showing log files and statistics of honeypots but they generally focus on high-level aggregate statistics and not about individual log anomalies. These anomalies might be user/password combinations, weird looking strings or other executables, indicating new device vulnerabilities of specific vendors. Imagine a device which was rushed to be released as the competition was already ahead. Someone might have forgotten to remove some kind of default code to access the system internals or the system was simply built unsafe because of the lack of time in development. If someone finds this vulnerability he can harm the system. This happens regularly by hackers and automated botnets, trying to exploit already known or new vulnerabilities of specific versions of software systems.

So with a honeypot we can potentially find new exploits, although a hacker might find out that we are not what we pretend to be. That is our main problem. We need to batch-process the log files to be able to further improve our honeypot systems. We want to find out why the attacker disconnected and view the previously executed commands, which might give us a chance to find ways to improve it. Malware could have found some ways to detect our system by executing common commands known to be honey analyzing the results. The main goal of this work is to find new ways to be one step ahead of the attacker and iteratively improve our deployed honeypots.

As every honeypot is generating its own log files, it can be seen that it would be beneficial to analyze and process the data where they are generated, parallel across all nodes. There are approaches like the Hadoop Distributed File System (HDFS) based on the Google File System (GFS) [2] and programming models like MapReduce which provide us this possibility and are built solely for the purpose to execute distributed and scalable software. Those frameworks will assist our thesis.

Therefore the main focus lies on techniques to adapt a honeypot using the information queried from potentially thousands of log files across different systems.

[1] `https://github.com/paralax/awesome-honeypots`

2 Status quo

The current state of the art for logging brute-force attacks and shell interactions of connection-based intrusion attacks is a SSH and Telnet honeypot called Cowrie[1] by micheloosterhof (Github). Cowrie grants the possibility to catch attacker actions and provides logs of all kinds like JSON, UML, TTY, Splunk HEC and different databases like MongoDB, SQLite and MySQL [1]. Although there exist data analyzation platforms like Splunk Enterprise[2] (which is not for free) with Cowrie analysis apps like Tango[3] they are all more concentrated on the individual analysis of honeypots and collection of statistics than on a severe analysis of individual attacker behaviors and action paths. Malware has become more intelligent recently which enlights the need for analysis and improvement of intrusion detection systems.[4]

3 Modus operandi

3.1 Goals

The primary goal of the work is to batch-process connection-based honeypot log data and extract useful information to improve existing systems and have as well a possibility to view interaction based attacker behavior. There are tons of attacks happening all the time, the tool should provide a possibility to detect changes in attacker behaviour over time like sudden disconnects after specific commands or general log data anomalies not previously shown up. As an initial try the Hadoop Framework[5] is used to process and gather information of the log files. As this is non-trivial there will only be time for an easy visualization and probably no time for machine learning attempts for detection of outliers, this will be left for future work. The software structure of the Hadoop framework can be viewed in 1. We will probably use the HDFS and MapReduce for distributed processing. For a further overview of what exactly the framework provides and what the thesis work has to provide view 2.

[1] https://github.com/cowrie/cowrie
[2] https://www.splunk.com/
[3] https://github.com/aplura/Tango
[4] https://www.avira.com/en/blog/new-mirai-variant-aisuru-detects-cowrie-opensource-honeypots
[5] https://hadoop.apache.org/

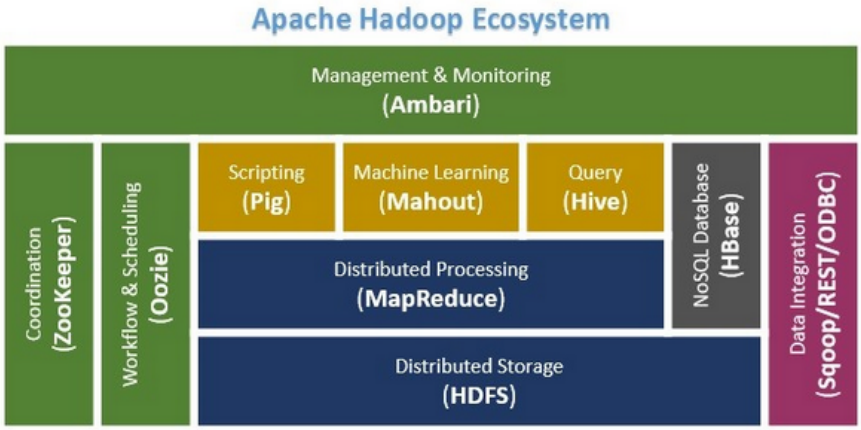

Figure 1: Hadoop framework [3]

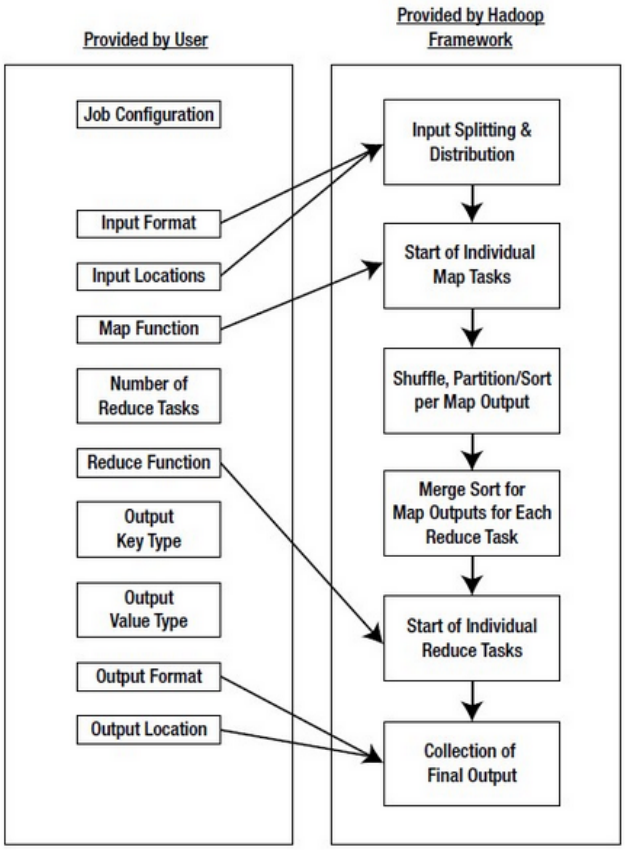

Figure 2: What does Hadoop provide? [3]

3.2 Timeline

February	Research ssh honeypots
March	Set up multiple cowrie honeypots, test Splunk, test logging functionalities
April	Research hadoop framework
April	Research batch-processing log data & aggregating information
May	Develop initial MapReduce job extracting some information
May	Run MapReduce job across multiple log files locally
May	Run MapReduce job on deployed nodes
June	Extract pre-disconnect commands frequency across all nodes
June	Detect outliers for different log tables
June	Handle bugs/improvements
July	Dream: Visualize, Provide hints to improve honeypots, Mahout for machine learning
July	Finish batch-processing system
August	Write thesis

3.3 Risks

- Gathering Data
  As the goal is to have test data for our web application in the first place there is a need to set up honeypots and create those log data for cowrie. With the attraction of attackers there is always a slight risk of them escaping our sandbox and using a honeypot as pivot node to penetrate productive systems. There were used dedicated droplets on DigitalOcean[1] to secure this.

- Database
  A database system with potential user connection data might attract some black-hats as well. A basic mongodb authentication and a firewall denying all incoming connections except for our honeypots and our webapp will grant proper security.

[1] `https://www.digitalocean.com/`

References

[1] W. Cabral, C. Valli, L. Sikos, and S. Wakeling. Review and analysis of cowrie artefacts and their potential to be used deceptively. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 166–171, 2019.

[2] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 29–43, 2003.

[3] I. Inc. Hadoop MapReduce tutorial, 2021.

[4] S. Kumar, B. Janet, and R. Eswari. Multi platform honeypot for generation of cyber threat intelligence. In *2019 IEEE 9th International Conference on Advanced Computing (IACC)*, pages 25–29, 2019.

[5] A. Kyriakou and N. Sklavos. Container-based honeypot deployment for the analysis of malicious activity. In *2018 Global Information Infrastructure and Networking Symposium (GIIS)*, pages 1–4, 2018.

[6] C. Sanders. *Intrusion Detection Honeypots, Detection Through Deception.* Applied Network Defense, 2020.