A Systematic Approach for Analyzing Log Files Based on String Matching Regular Expressions

Keshav Kaushik, Gargeya Sharma, Gaurav Goyal, Asmit Kumar Sharma, and Ashish Chaubey

Abstract In the past few years, there has been a tremendous increase in cyberattacks and cybercrimes. Technology is changing at a very fast pace, thus inviting more advanced cyberattacks. Any event that is triggered on the system is recorded in the form of logs in log files. It may be any warning, any alert, and information, and all the things get stored in the logs. Therefore, from the security point of view, analyzing the logs plays a crucial role in the forensic investigation or for analytics purposes also. This paper highlights a systematic approach for analyzing the log files using a string-matching algorithm and regular expressions. Thus, it helps in log analysis, management, and analytics for future reference. Analyzing logs in a systematic way is always crucial in digital forensics, and it will help in the smooth conduction of forensic investigations.

Keywords Log analysis · String matching · Regular expressions · Cyberattacks

1 Introduction

We live in a world where data surrounds us in every direction. Where there is technology, some amount of data is always involved. One of the many data sources that accumulate around us is in the form of log files. These log files are automatically produced files that contain the record of events from various software and operating systems. According to statistics, an average enterprise accumulates up to 4 GB of log data a day from all their servers and other network devices. Over 95% of that data within those log files are the entries that record every successful/unsuccessful event or transaction taking place in those systems. For example, it can be a server crash, user logins, logouts, applications start-up and shutdown, file access, etc. These log files are unnoticed by many people working under that enterprise because their production is a regular process and relatively holds less priority and attention than other tasks. However, it is the active and complete

K. Kaushik (✉) · G. Sharma · G. Goyal · A. K. Sharma · A. Chaubey
University of Petroleum and Energy Studies, Dehradun, India

K. Khanna et al. (eds.), *Cyber Security and Digital Forensics*,
Lecture Notes on Data Engineering and Communications Technologies 73,
https://doi.org/10.1007/978-981-16-3961-6_1

analysis of these log data that creates the differences among organizations, how they handle failures, changes to adapt by looking at the behavior of daily analysis periodically, etc. So, underestimating these data files is not advised for the smarter functioning of any organization and its client interactions.

This paper focuses on the same fundamental of not ignoring these log files and analyzing them in a much faster and easier way by the owners of these log files. Data analysis is getting more popular in the current few years than it has ever been. Since, we are achieving some invaluable information using analysis on the unstructured data present around us with the help of big data analytics, machine learning, and other scientific techniques. Analyzing logs is also a part of them. They contain some very intuitive information about the users, services, machines, their patterns, and various relationships between them, which can help us formulate techniques to take advantage of such information and provide more personal and better services. In addition to better services, this also mitigates a large amount of risk involved in the process. With predictive and descriptive analysis, organizations can build a more secure and effective network, which in turn prevents losses and increases their overall profits. Due to its increased popularity and attention, log analysis has spread its role in various directions. For example, correlational analysis: This is where you find the correlation between data that is not visible in a single log, especially since multiple logs are relevant over a single incident. For instance, if an attack is experienced, then you can perform correlation analysis using logs generated by servers, firewall, network devices, and other possible sources involved and find those logs that are relevant to that particular attack. During such conditions, log files play an important role as a piece of evidence as their analysis helps to find the cause and suspect of the attack.

There are other useful use cases as well with log analysis, one of which is anomaly detection. Through this, we can overlook the usual less important logs and only focus on unusual and unique log entries for further investigation. This paper focused on achieving anomaly detection on IPs through their number of occurrences either overall or in a particular period. Other than just detection of any existing anomalies, the feature to block such IP addresses if the user wishes to be also included, while adding all the blocked IPs in a separate file to maintain for reporting or future references. All the functionalities are further discussed in detail in the working methodology section. This paper fulfills the intention of providing enterprises and organizations with a tool that will help reduce the extensive searching and crawling through the entire log files related to the required situation and makes the process easier for further investigation and analysis. The implementation for this paper worked on a command–line interface and built using C language on top of the Linux operating system. With this, many users can be covered in the industry, as most of the servers out there are Linux kernel-based, contributing toward speeding the process of analyzing logs for various use cases.

2 Related Work

The extensive quantity of research proposing different technologies that can be used for analysis is increasing every year. Among them, many are focusing on the automated systems for the analysis of log files. Pinjia He in [1] reveals his dataset "log hub" that contains 17 datasets including stand-alone software, web server, mobile, and operating systems logs. These can be used for AI-powered log analytics which focuses on anomaly detection. It is an approach for log analysis in combination with machine learning. In this, regularly generated usual logs are ignored, and the unique or different ones in the log files are recognized as anomalies and treated as potential threats, respective to how the model was trained earlier. Zengchao NI and Honqui Liu in [2] discussed the idea of dealing with massive web system logs by extracting log templates based on the label recognition method effectively solving the problem of insufficient log format. Effective log monitoring allows us to do behavioral analysis. Xiaojian Liu and Yi Zhu in [3] talk about generating valued genealogy information that helps in generating attributes for specialized profiles. Specialized profiles lead to personalized recommendations, and thus biased monitoring and blocking of users are attained. Analyzing the data stored in logs of web search services provides a great view of the information searching tactics of the user. This can help in developing information systems, design, interface development, and information architecture construction for content collection. Robust learning of web search logs is quite important. Laura M. Koesten in [4] describes the drawback of web search engines in finding the most relevant datasets that meet the user's query. They suggested the first query analysis for dataset search containing logs of four open data portals. Logs from that reveal that queries stated on the data portal are quite different from those issued to web search engines in their length and structure, which finally draws the result that portals are used explorative rather than answer focused questions. Nevertheless, the exploration does not stop here as researchers are using log datasets to find anomalies in the system. Traditionally, anomalies are detected by using keyword search and regular expression match that is the root idea of our project. In modern times, many machine learning-based detection mechanisms have been proposed as discussed in [1]. Qimin Cao and Yinrong Qiao in [5] carry the above-stated idea by implementing a two-level machine learning algorithm and using a decision tree model to classify between normal and anomalous datasets, thus adding future inspiration for our work.

3 Working Methodology

The log file of any server consists of multiple records, and each record takes up one row in the log file. These log files could be in terabytes or more, thus it becomes very difficult to filter out required data. Each row of a log file could contain multiple columns, for example, the log file of a web server contains, source IP address, date, time, time zone, status code, response code, etc. These columns depend on a server that generates them. Thus, we often require a tool that can filter out required data whenever needed. Our project aims to create a tool that can filter the log files based on the requirement. As soon as the tool is executed, it displays a menu to the user that lists all the features provided by the tool. While writing the code, we kept the basic requirements in mind.

This tool provides five features to deal with logs as of now. These features are as follows:

I Convert the log file into CSV format.
II Search for a particular keyword in the log file.
III Block and unblock any IP address from the server.
IV Visualize the log files.
V Filter the logs in a particular time frame (Fig. 1).

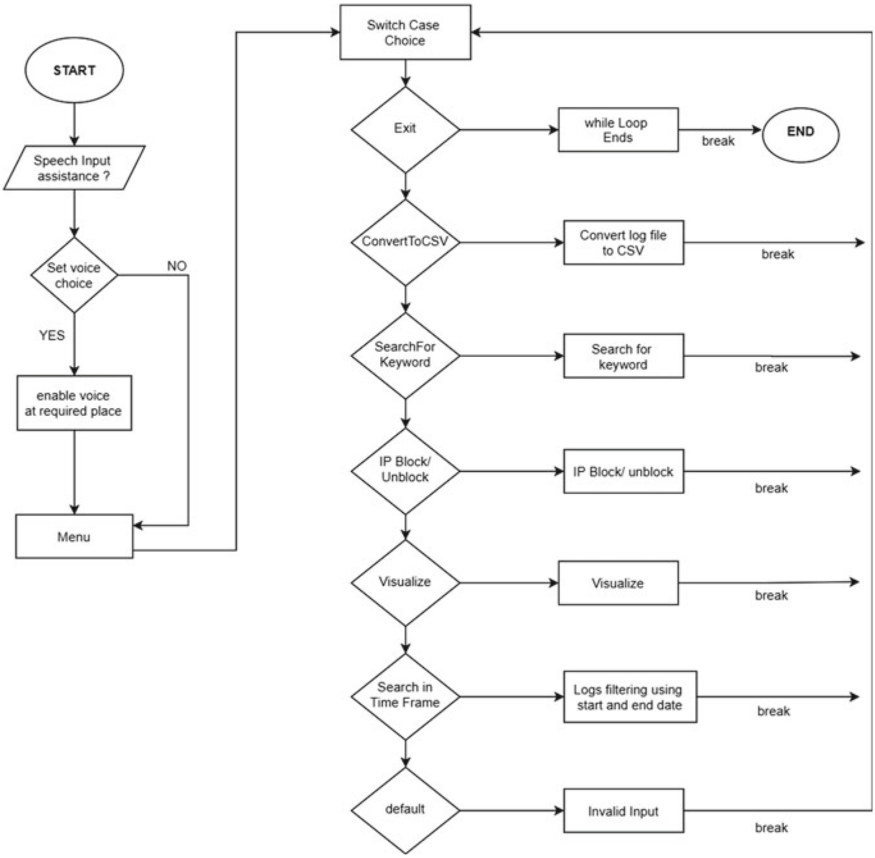

Fig. 1 Flow diagram explaining the working methodology

3.1 Converting the Log File into CSV Format

Log files could be very difficult to read most of the time, and this is why we need to convert them into such a format so that further processing can be done with ease. We wrote the code to convert the log file into comma-separated values (CSV) format. Nevertheless, why did we choose the CSV format specifically? Performing data cleaning, data processing, and data analysis can be done easily on a CSV file. We used Linux utilities like "sed", "awk", and regular expressions to convert the log [6] file into CSV format. We used "awk" to separate fields of a log file and write the output into a separate file. Then we used "sed" to remove unwanted special characters and separate different columns with a comma (",").

3.2 Search for a Particular Keyword in the Log File

Sometimes or the other, we might run into a situation where we want to search a pattern or keyword in a log file [7]. We implemented this feature using a basic pattern-matching algorithm. We used the "strstr()" method from the "string.h" header file of the C programming language. This algorithm has the quadratic worst case time complexity with respect to the searched time. Through the median of this tool, we are providing five options to search for a keyword. The user can search for either an IP address, date, HTTP method, status code, response code, or other. These options are specifically provided to deal with the web server logs, though this can be tweaked accordingly.

3.3 Block and Unblock an IP Address

This is a vital feature of any log analysis tool. While dealing with logs, we might need to blacklist certain IP addresses belonging to various users trying to access the server. To block and unblock the IP address, we use the "firewall-cmd" command–line tool to do the same. The program initially creates a new file "block.txt" which will store all the blocked user's IP addresses. We take the IP addresses to be blocked or unblocked from the user.

- **Command to block the IP**
  "firewall-cmd –permanent –add-rich-rule=rule family='ipv4' source address={input IP here} reject"
- **Command to unblock the IP**

  "firewall-cmd –permanent –remove-rich-rule=rule family='ipv4' source address={input IP here} reject"

The input IP address is concatenated within the command. The command is then executed to block/unblock the file. However, this is not it; we will be maintaining the list of blocked IP addresses (block.txt) so that our tool is unambiguous. Before blocking the IP address, the "block.txt" is traversed and if the IP address, is already blocked, then it will prompt an alert to the user. Similarly, while unblocking the IP address, it should be present in the "block.txt", only then it will be unblocked.

3.4 Visualizing Log Files

The detailed working of this feature will be explained in the next section. In short, we will take a list of IP addresses from the user as input and then plot a graph that will represent the number of occurrences of an IP address in the log file. This representation is not limited to occurrences of IP addresses only; it can also be used to visualize the number of requests in a day that were made to the server.

3.5 Search for Logs in a Time Frame

This feature provides an option to search for logs [8] in a particular period (e.g., we can filter out logs that were recorded between January 1, 2021, to January 5, 2021). Both beginning and end dates were taken from the user. To print the output, we traverse the log file line by line. If the beginning date is found in any line, then the "flag" variable is set to "true". Then if the "flag" variable is set to "false", the end date is found in any line. Within each iteration, we print out the corresponding line if the "flag" is set to "true". This will print out logs from the beginning date to the end date (exclusive). Now we have to print only those records of the end date. This can be easily done by traversing the log file and print only those records that contain the end date. Pattern matching is again the core algorithm for this feature.

4 Results and Discussion

After the experimentation for further convenience and better understanding, we visualized the final output in the form of a bar graph. As mentioned above, we provided functionality in our tool for visualizing IP addresses (refer to Sect. 3.4). We know that visualizing statistics [9] in a proper way provides additional intuition for a better understanding of the data, which often leads to a better examination of that same data. The below-represented bar graph depicts the visualization of the number of occurrences of different IP addresses required by the user in the required time frame in order to examine if any one of them shows abnormal behavior or in simple words shows a huge increase in its occurrences which could be directing

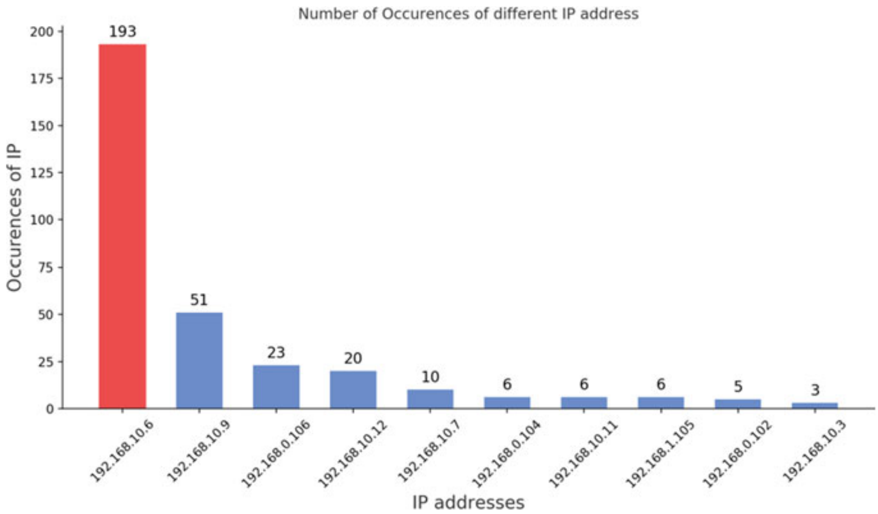

Fig. 2 Occurrences of IP addresses in the log file

toward some kind of malicious behavior or signs of malicious activity taking place. In Fig. 2, we can see that the first IP address (192.168.10.6) that is selected by the user within the selected time frame occurred the most (193 times to be precise) which is undoubtedly a high number for any IP address. This is also the case, due to the current threshold being set at a limit of 100 occurrences within a period, which implies that there is some issue or unethical activity is being performed. (Note: The threshold can be changed/set according to the requirements of the user). In Fig. 3, two pie plots are presented, one of the left represents the visualization over the comparison of occurrences of the different types of requests that were in the dataset,

Fig. 3 Distribution of various parameters (request type and status codes) in the log file

i.e., if the request was a GET or a POST request. The other plot on the right represents occurrences [10] of the number of requests that were associated with different status codes as the result of that request.

5 Conclusion

As we all know the importance of logs in event correlation and event reconstructions in cybercrime investigation, this article focuses on analyzing the logs in a systematic way with the help of a string-matching algorithm and regular expressions. The authors observed the visualization of log files, searching of logs, and finding the occurrences of IP addresses. The approach discussed in this paper can be further extended in various aspects like analyzing the logs in real time, deploying them using Docker containers, etc. The authors will extend the work in the near future and would like to explore the various possible dimensions of analyzing the logs.

References

1. He, S., Zhu, J., He, P., Lyu, M.R.: Experience report: system log analysis for anomaly detection. In: 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), Ottawa, ON, pp. 207–218 (2016). https://doi.org/10.1109/ISSRE.2016.21
2. Ni, Z., Liu, H., Chen, Y., Wu, D.: Research and implementation of a method for web log analysis template extraction. Procedia Comput. Sci. **162**, 673–681 (2019). ISSN: 1877-0509
3. Liu, X., Zhu, Y., Ji, S.: Web log analysis in genealogy system. In: 2020 IEEE International Conference on Knowledge Graph (ICKG), Nanjing, China, pp. 536–543 (2020). https://doi.org/10.1109/ICBK50248.2020.00081
4. Kacprzak, E., Koesten, L., Ibáñez, L., Simperl, E., Tennison, J.: A query log analysis of dataset search. In: Lecture Notes in Computer Science, pp. 429–436 (2017). https://doi.org/10.1007/978-3-319-60131-1_29
5. Cao, Q., Qiao, Y., Lyu, Z.: Machine learning to detect anomalies in web log analysis. In: 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, pp. 519–523 (2017). https://doi.org/10.1109/CompComm.2017.8322600
6. Best Practices: Event Log Management for Security and Compliance (2021). https://www.whatsupgold.com/resources/best-practices/event-log-management
7. Jansen, B.: The methodology of search log analysis. In: Handbook of Research on Web Log Analysis, pp. 100–123 (2009). https://doi.org/10.4018/978-1-59904-974-8.ch006
8. Sultana, N., Paira, S., Chandra, S., Alam, S.: A brief study and analysis of different searching algorithms. In: 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT) (2017). https://doi.org/10.1109/icecct.2017.8117821
9. IBM Knowledge Center.: Ibm.com (2021). https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_72/rzasp/rzasp_regularexpression.html
10. Lu, X.: The analysis of KMP algorithm and its optimization. J. Phys. Conf. Ser. **1345**, 042005 (2019). https://doi.org/10.1088/1742-6596/1345/4/042005