# Supervised Learning for Detecting Cognitive Security Anomalies in Real-Time Log Data

Md. Al Amin
*Department of Information and Communication Technology*
*Bangladesh University of Professionals*
Dhaka, Bangladesh
alaminopu.me@gmail.com

2022 IEEE World Conference on Applied Intelligence and Computing (AIC) | 978-1-6654-7988-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/AIC55036.2022.9848922

***Abstract*— Every system generates a large quantity of logs. Logs are immensely crucial for monitoring a system, inspecting anomalous behaviors, and analyzing errors. By using log data, many recent studies suggest that efficient and accurate machine learning classifiers can use to detect anomalies. The source diversification and the unstructured nature of logs create difficulties in subsequent analysis. Even though the evaluation of automatic log parsing opened up the door to further research. To decrease manual work, many anomaly detection systems based on automated log analysis have been developed. However, due to the lack of a research and comparison of multiple anomaly detection methods, developers may still be unsure about which anomaly detection methods to utilize. Even if developers employ an anomaly detection technique, re-implementation takes time. To address these issues, we present a comprehensive study of detecting security anomalies from real-time log data based on different supervised machine learning algorithms and trained with publicly archived logs data. Two selected production log datasets with 15,923,592 and 365,298 log records were used to evaluate these algorithms. We also proposed an approach that the ability to provide visibility into the system, the proper way to acquire insight using a strong monitoring system that collects metrics, represents data, and automated methods to give notifications to administrators to bring to their awareness a significant change in the system's status. We assessed the model's performance using a variety of well-known assessment metrics, including "precision", "recall", "specificity", "F1 measure", and "accuracy".**

**Keywords— Security, log analysis, anomaly detection, supervised learning**

## I. Introduction

The internet is a vast, hypertext-enabled, linked, semi-structured, widely scattered, and immensely diversified information repository. As an information gateway, the Web continues to expand at an enormous speed [1]. When analyzing real-world data sets, knowing which examples stand out as being different from all others is a typical requirement. Anomalies can be caused by data errors, but they can also be symptoms of a previously undiscovered underlying process; Hawkins [2] describes an anomaly as an observation that differs so significantly from others that it raises the possibility that it was generated by a different system. Each log message is written by a logging statement and records a single system event with its message header and message body, as illustrated in Fig.1.

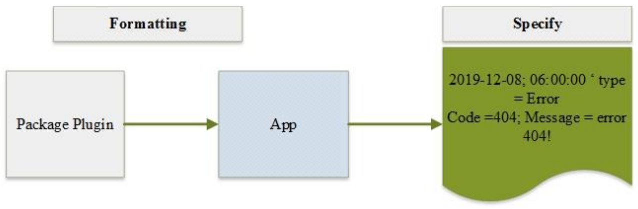
Formatting
Specify
Package Plugin
App
2019-12-08; 06:00:00 ' type = Error Code =404; Message = error 404!

Figure 1 Example of Log Formatting

Large online service systems like Microsoft's, Google's, and Amazon's are becoming increasingly huge and complicated, with hundreds of dispersed components and the ability to handle many concurrent users. Before deploying them in a production environment, engineers often test online service systems in a lab (testing) environment. The production environment has a wide and complex cloud architecture supporting a massive amount of data, whereas the lab environment often has a small, virtual cloud environment with a constrained amount of data. Even if they have been thoroughly tested in the lab, online service systems frequently encounter unexpected occurrences because of the differences between the lab and production situations. Logs are essential for the development and upkeep of software systems. Logs frequently provide comprehensive system runtime information that aids developers and support engineers in better understanding system behavior and tracking down potential issues. Numerous system administration and troubleshooting tasks, such as assessing use statistics, ensuring application security, identifying performance abnormalities, and spotting errors and crashes are made possible by the vast data and wide availability of logs [3]. Despite the enormous potential buried in logs, adequately analyzing them remains a major difficulty. To begin with, current software systems create many logs on a regular basis. Even using search and filter capabilities, manually inspecting log messages for critical diagnostic information is problematic regarding the huge number of logs. Second, most of the log messages are intrinsically unstructured because, for simplicity and flexibility, developers often record system occurrences using free text [4]. This complicates the automatic detection of log record even further. Strong text search and analytics based on machine learning are now widespread in many recent studies. The first and most crucial step in allowing such log analysis is log parsing, a method for turning a stream of ordered events from free-text raw log messages. Typical log parsing relies on hand-written regular expressions or grok

Authorized licensed use limited to: Malardalen University. Downloaded on February 24,2023 at 07:51:26 UTC from IEEE Xplore. Restrictions apply.

patterns to extract event templates and important parameters [5]. Manually developing ad-hoc rules to analyze many logs, while simple, is a time-consuming and error-prone process. Modern software systems' logging code is routinely updated, resulting in the unavoidable cost of modifying these customized parsing rules on a regular basis. To reduce manual efforts in log parsing, a study [6] examined static analysis algorithms to extract event templates directly from source code. Source code is not always available in practice, despite the fact that it is sometimes a good strategy (e.g., when using third-party components). Building such a static analysis tool for software systems created using various programming languages, however, requires non-trivial work. In order to achieve the goal of automated log parsing, several data-driven strategies have been offered in academia and industry. These methods may be able to recognize trends in log data and automatically produce common event templates, in contrast to human rules and source code parsing [7]. Every day, a sizable number of logs are produced by the online service systems. There are more logs produced as service systems get more complex. Some massive systems that offer global services may produce daily log data volumes that go into the tens of gigabytes. A Microsoft service system generates over 1PB of logs each day. As a result, analyzing the logs by hand to identify an issue takes a long time after it occurs.

The "failover" technique [8], which dynamically assigns work across computing nodes based on parameters such as availability and performance, is commonly used in modern online systems.

The logs reveal a significant number of recurring difficulties. If a flaw is found in a typical software system, it will be corrected and, in most situations, will not appear in the next version. However, a large-scale online service system has a lot of recurrent issues, which might result in a lot of wasted time examining logs and identifying previously recognized issues. Fourth, the variety of log messages is huge. The complexity of an online service makes it possible for different execution strategies to fail in the same way. The regular modifications to service features and environments significantly broaden the range of log messages. Additionally, not all log messages are created equally when it comes to detecting faults; although some log messages occur in both normal and unsuccessful scenarios, some only do so and are more likely to be related to the problems. Developers must thus examine a huge number of extremely diversified logs to efficiently identify and differentiate distinct service faults [9].

Regular expressions created and maintained by humans are used in conventional log parsing techniques. Big systems with a variety of software and hardware components make this manual effort difficult to maintain. Existing log parsing algorithms can't reliably produce good parsing results, according to tests of their performance on various systems. This necessitates the selection of a parsing mechanism appropriate for the application or system in question, as well as the incorporation of domain-specific information. We contend that the least amount of human engagement is required for log parsing algorithms to be reliable on log data from a variety of systems, including single apps, mobile operating systems, and cloud infrastructure management platforms [10]. Meanwhile, the unavailability of publicly available tools makes automated log processing difficult to implement. Users may benefit from the benchmarking results by having a better understanding of the variations across log parsers and by using them to direct the adoption of automated log parsing in production. We anticipate that the accessibility of tools and benchmarks, as well as the commercial experiences exposed in this work, will facilitate future research and increase the use of automated log parsing in business [7]. In several application domains, data-driven anomaly detection technologies are gaining traction. When dealing with high-dimensional Anomaly Detection situations, machine learning algorithms have shown to be extremely successful. This paper will look at a few key characteristics of the approaches, such as computing costs and interpretability. We also provide a study of real-world industry applications of the explored techniques, as well as related libraries and open implementations, to assist practitioners.

Complex system logs are useful resources for tracking systems and looking into strange behavior while developing and maintaining software systems [7]. To create stable and secure systems, anomaly identification and reliability prediction are essential, and system logs record the current status of the system and significant events at different critical points. Millions of people might be impacted by a flaw or exception in the massive software systems of today. In order to allow intelligent operation and maintenance, it is becoming more challenging to manually retrieve important decision-making data from valuable log resources as log data volume increases. The ability to analyze methodically eludes even developers or analysts with years of experience in the subject [11].

Additionally, manual analysis is ineffective since it takes a lot of time and expertise. As a result, one of the key issues in research on analysis and prediction technologies based on system logs data is the quest for efficient and accurate automatic processing approaches for big logs. Based on a system analysis of the complete log processing process, includes log analysis, extraction and classification, intrusion detection, forecasting and evaluation, and real-time dependability, the massive system logs in this research are analyzed using the classification algorithm. For different data volume training and testing sets, “KNN, Decision Tree (DT), Random Forest (RF), Logistics Regression (LR), and SVM (Support Vector Machines)” prediction models of classical machine learning are explored for intrusion detection system [16]. In this paper, we offer an anomaly detection framework and a log identification method that takes into account all the characteristics of online service system logs. The paper's main contributions are as follows:

i) A review of cutting-edge research on large-scale data processing for anomaly identification.

ii) Define a taxonomy to classify current research into areas including big data, intrusion detection, computational intelligence techniques, and applications.

iii) Identifying research obstacles and providing recommendations for future researchers.

Authorized licensed use limited to: Malardalen University. Downloaded on February 24,2023 at 07:51:26 UTC from IEEE Xplore. Restrictions apply.

The remainder of the research is structured as follows: In Section 2, we present the context and motivation behind our work. The dataset, recommended architecture, and workflow of the suggested system are all described in Section 3. Section 4 illustrates the evaluation results. Finally, discusses conclusions and future work recommendations.

## II. Related Work

Each day, a data center produces terabytes or petabytes worth of log files. The storage and analysis of such vast amounts of log data is rather challenging. In addition to being large, log files are also heterogeneously structured, which makes evaluation challenging. Since traditional database systems are unable to manage such a large volume of logs, they are inappropriate for analysing such log files. When comparing SQL DBMS and Hadoop MapReduce in 2009 [12], Andrew Pavlo and Erik Paulson found that Hadoop MapReduce tunes up with the work sooner and loads data faster than DBMS. Standard DBMS are also unable to handle very large datasets. Hadoop is the best platform for storing log files and processing them in parallel using the MapReduce software since log files are a sort of huge data. Business organizations may now store and analyze data using Apache Hadoop. Doug Cutting established the open-source Hadoop project, which is run by the Apache Software Foundation [14]. It enables software to run over petabytes of data and tens of thousands of nodes. Although it can be used on a single system, its true power lies in its ability to scale to hundreds or thousands of devices, each with multiple CPU cores. Hadoop is specifically built to operate on massive amounts of data in parallel by linking inexpensive machines. Hadoop divides log files into blocks, which are then distributed evenly among thousands of servers in a Hadoop cluster. In order to provide features like reliability and fault tolerance, it also replicates these blocks over several nodes [12].

TABLE 1 SHOWS THE ANOMALY APPLICATIONS, ALGORITHMS FROM DIFFERENT LITERATURE

| Algorithms | Applications | Reference |
|---|---|---|
| Multilayer Perceptron | Network Intrusion Detection System | [20] |
| BN, Markov MMC | Network Intrusion Detection System | [21] |
| One Class SVM | Social Network | [22] |
| Dynamic Rule Based | Telephone Calls Anomalies | [23] |
| KNN | Adaptive anomaly detection | [24] |
| SVM, Regression | Sensor Network | [25] |
| Intrusion Detection | Entropy-based | [26] |
| Web Applications | PCA, ROCs , AutoRegressive | [27] |
| Our predictive algorithms | LR, DT, SVM, RF, KNN | - |

Schultz et al. [15] suggested employing data mining methods to detect malicious executables. They used a range of popular data mining techniques to find malware executables that had previously gone undetected. They found that some methods, such Multi-Label Naive Bayes Classification, can beat customary techniques based on signatures. The authors [16] performed a number of analyses utilizing a range of non-parametric statistical techniques to determine the trend of aging. They then used time-series analysis to predict future resource values and evaluate whether a resource will run out. They use statistical algorithms over resources with known anomalies, but we use machine learning to uncover software bugs on our own. They [16] investigated the modeling and prediction of deterministically software anomalies using three well-known machine learning algorithms: “Naive Bayes, Decision Trees, and Support Vector Machines”. The authors did not, however, evaluate their method in a dynamic setting or under multi-resource depletion, which could have led to software bugs.

We compared our work to several current studies on anomaly detection and description are presents in Table 1.

Following the review of earlier research on anomaly identification from real-time log data, the classification algorithms were selected as the research's predictive model.

## III. Materials and Methods

### A. Experimental Setup

Log gathering, log analysis, extraction and classification, and intrusion detection make up the core of the experimental setup for anomaly detection. We focus on five classification techniques after acquiring the relevant features. The experimental setup for anomaly detection using machine learning is shown in Figure 2. This paper investigates classification techniques, pre-trained models in order to develop an effective solution for real-time identification, tracking, and implementation of machine learning based intrusion detection system.

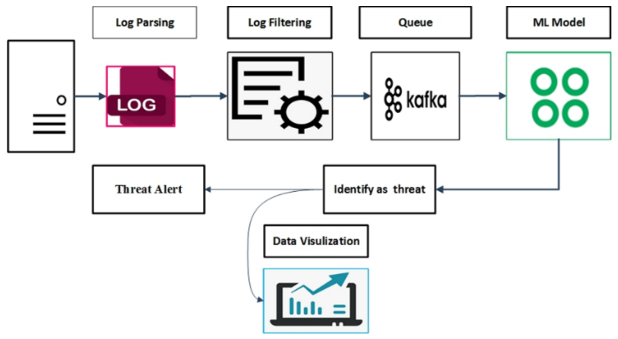

Figure 2: Experimental setup for real time log analysis and anomaly detection

In this experiment, we consider four steps for log processing and “anomaly detection”. As follows,

**Log collection:** A timestamp and a log message outlining what happened are included in each log that large-scale systems regularly produce to record system statuses and

Authorized licensed use limited to: Malardalen University. Downloaded on February 24,2023 at 07:51:26 UTC from IEEE Xplore. Restrictions apply.

runtime data. Logs are first collected for later usage since this crucial information can be used for a number of purposes.

**Log parsing:** Unstructured text files called logs contain freeform text. The purpose of log parsing is to organize raw logs by removing a set of event templates from them. Any log message might potentially be converted into an event template by applying specific parameters.

**Feature extraction:** In order to use machine learning models, we must first decompose logs into individual events and then encode those events into numerical feature vectors. First, we used a variety of grouping algorithms, including fixed windows, sliding windows, and session windows, to divide the raw logs into a number of log sequences. Then, for each log sequence, we create an event count vector (a feature vector) that represents the frequency of each event. A feature matrix, sometimes referred to as an event count matrix, can be created by combining all feature vectors.

**Anomaly detection:** The feature matrix can also be used to train machine learning models, which will create an "anomaly detection" model. A new incoming log sequence can be evaluated to see if it is an anomaly using the defined model.

## B. Data Collection

Production logs are a limited source of information because companies currently share them due to confidentiality issues. However, we were able to obtain two log datasets, HDFS data and BGL data [17], which are suitable for evaluating current anomaly detection algorithms after carefully examining the literature and getting in touch with the necessary authors. The combined 15,923,592 log messages and 365,298 anomaly samples in both datasets, which are obtained from production systems, have been meticulously categorized by the original domain experts. Therefore, for the purposes of accuracy evaluation, we treat these labels (anomaly or not) as the absolute truth. 11,175,629 log messages from the Amazon EC2 infrastructure are included in the HDFS data set [18]. 4,747,963 log records were gathered by the "BlueGene/L supercomputer machine at Lawrence Livermore National Labs" in the BGL data [17]. In contrast to HDFS data, BGL logs don't record any identifier for each job execution. We must first slice logs as log sequences using fixed or sliding windows before extracting the pertinent event count vectors from the resulting product. However, the size of the selected window determines the number of windows (and step size). A log sequence is labeled as an anomaly if it includes any failure logs, and 348,460 log messages are classified as failures in the BGL data.

## C. Classification Techniques

A model is created using labeled training data as part of the supervised learning process in machine learning. For supervised "anomaly detection", labeled training data that specify whether a state is normal or abnormal are necessary. The accuracy of the model increases with the labeling of the training data. Five supervised classification techniques will be used in this study, i.e. "logistic regression (LR), decision tree (DT), support vector machine (SVM), random forest (RF), and K-nearest neighbors (KNN)".

## D. Evaluation Measurement

The accuracy of predictions made by classification algorithms is evaluated using a classification report. The report shows the major classification metrics' "accuracy, recall, and f1-score" for each class. The measurements are produced using both "true and false positives" as well as "false negatives". "True positive, false positive, true negative, false negative, and false negative" are the four qualities of the metrics [19]. "Precision, recall, and f1-score" were determined using equations (6), (7), (8), and (9) respectively.

"Precision": the proportion of the actual positive estimate to the overall positive estimate for the model (correct and incorrect).

It is expressed as —

$$\text{Precision\_}_i = \text{TP\_}_i/(\text{TP\_}_i+\text{FP\_}_i) \quad (6)$$

"Recall / Sensitivity": There is a significant likelihood of being able to forecast. It is expressed mathematically as —

$$\text{Recall\_}_i = \text{TP\_}_i/(\text{TP\_}_i+\text{FN\_}_i) \quad (7)$$

"F1 score": The "harmonic mean of Precision and Recall" yields a more accurate estimate than the Accuracy Metric of erroneously categorized instances. Mathematically, it's written as —

$$\text{F1\_}_i = 2\cdot(\text{Precision\_}_i*\text{Recall\_}_i)/(\text{Precision\_}_i+\text{Recall\_}_i) \quad (8)$$

"Accuracy": It refers of all the occurrences that could have been predicted accurately. It is formatted as - $\text{Accuracy\_}_i = (\text{TP\_}_i+\text{TN\_}_i)/(\text{TP\_}_i+\text{TN\_}_i+\text{FP\_}_i+\text{FN\_}_i)$ (9)

## IV. Result and Discussions

In this study, we evaluated five different "machine learning classification" algorithms for anomaly identification through analysis. Model assessment measures are required to assess the model's efficiency. The selection of assessment measures is based on the machine learning function used (such as, among others, classification, regression, ranking, clustering, and topic modeling). Precision-recall and other measures are important for a variety of jobs. Most machine learning applications are supervised learning problems like classification and regression.

We use HDFS and BGL data to test the accuracy of supervised techniques. The event count matrix is generated using session windows to slice HDFS data. We fitted the algorithms using "training" data before applying them to "testing" data to assess the validity of five supervised approaches (LR, DT, SVM, RF, and KNN). Both datasets were split into two parts: 30% for testing and 70% for training. With both fixed and sliding windows, HDFS data has significantly greater accuracy than BGL data. The HDFS

Authorized licensed use limited to: Malardalen University. Downloaded on February 24,2023 at 07:51:26 UTC from IEEE Xplore. Restrictions apply.

system only records 29 event categories, which is significantly less than the 385 event types found in BGL data. The performance of "anomaly detection" on HDFS and data is shown in Table 2, and all five techniques perform very well on this data, with an F-measure close to 1. Table 3 also displays the effectiveness of anomaly detection on BGL and data.

TABLE 2 PERFORMANCE MEASUREMENT OF FIVE SUPERVISED CLASSIFIERS ON HDFS LOG DATA

| **Algorithms** | **Accuracy_i** | **Precision_i** | **Recall_i** | **F1 Measure_i** |
|---|---|---|---|---|
| LR | 1.0 | 0.98 | 0.97 | 1.0 |
| DT | 0.83 | 0.97 | 1.0 | 1.0 |
| SVM | 0.89 | 0.97 | 0.92 | 0.74 |
| RF | 0.99 | 1.0 | 0.94 | 1.0 |
| KNN | 0.85 | 0.95 | 0.91 | 0.93 |

According to Table 1, the DT achieved the lowest performance and the LR had the highest accuracy performance. The results unmistakably demonstrate that, compared to the other algorithms, the RF attained the best precision and KNN the lowest accuracy. On the other hand, DT and KNN outperformed all other classifiers in terms of recall. We find that LR, with an F-measure of 1, obtains the best overall accuracy among supervised approaches. For the BGL data set, Table 2 displays the results of five supervised machine learning techniques. With the highest classification accuracy of 0.98, LR and RF exceed all other classification techniques when accuracy is taken into account. KNN also performs the least well of any other classification method.

TABLE 3 PERFORMANCE OF FIVE SUPERVISED CLASSIFIERS ON BGL LOG DATA

| **Algorithms** | **Accuracy_i** | **Precision_i** | **Recall_i** | **F1 Measure_i** |
|---|---|---|---|---|
| LR | 0.98 | 0.99 | 0.84 | 1.0 |
| DT | 0.85 | 0.81 | 0.71 | 0.85 |
| SVM | 0.88 | 0.82 | 0.87 | 0.98 |
| RF | 0.98 | 0.70 | 0.82 | 0.87 |
| KNN | 0.81 | 0.75 | 0.81 | 0.81 |

Tables 2 and 3 make it very evident that LR performs better than other classification algorithms while KNN performs worse than the other 5 classification algorithms. All machine learning classifiers have accuracy levels above 80%, proving the effectiveness of these methods. Performance across the board shows that the five-classification technique largely predicts accurate outcomes.

### *A. Limitations*

Even while current technologies are capable of detecting anomalies, accuracy issues make the results' dependability unreliable. More precision can sometimes be obtained at the cost of lengthy computations and processing times. Combining real-time big data technologies with hybrid machine learning algorithms will be the key to solving this issue and creating a powerful meta-learning tool that can accurately analyze the enormous volume of data produced by contemporary applications while using less memory and processing power.

## V. Conclusion

In this paper, we investigated real-time log data with machine learning with the aim of anomaly detection. From the perspective of use cases, we examined recent real-time and anomaly detection research. We were able to discover the issues related to real-time anomaly detection by looking at these use cases. We talked about the limitations and issues with the current approaches of identifying abnormal threats in the targeted area.

Real-time consistency and "anomaly detection" are the main topics of this study. The real-time calculation of composite services based on system logs and the analysis of anomaly source causes were not carried out. This article gives a framework for making decisions on the operation and maintenance of the system's reliability through a series of tasks that start with processing raw log data and end with "feature extraction, anomaly detection, and consistency evaluation". The entire log processing process is described. We also emphasized cutting-edge techniques that present research challenges in real-time anomaly identification. To forecast anomalies in log data, traditional machine learning techniques including "LR, SVM, DT, RF, and KNN" are utilized. The system's real-time reliability is evaluated after "anomaly detection", and when the recall rate is low, the measurement of real-time reliability is changed, greatly enhancing the accuracy of real-time reliability measurement. We hope that our work, together with the tools and benchmarks offered, it will inspire more research into log analysis in the future.

## References

[1] "[PDF] An Overview of Preprocessing on Web Log Data for Web Usage Analysis | Semantic Scholar." https://www.semanticscholar.org/paper/An-Overview-of-Preprocessing-on-Web-Log-Data-for-Lakshmi-Rao/876fec354d167ee7fab369c3a1b9351300e67220 (accessed Apr. 24, 2022).

[2] "Identification of Outliers | SpringerLink." https://link.springer.com/book/10.1007/978-94-015-3994-4 (accessed Apr. 24, 2022).

[3] A. Oprea et al., "Detection of Early-Stage Enterprise Infection by Mining Large-Scale Log Data," Proc. Int. Conf. Dependable Syst. Networks, vol. 2015-September, pp. 45–56, Sep. 2015, doi: 10.1109/DSN.2015.14.

[4] P. He et al., "An evaluation study on log parsing and its use in log mining," Proc. - 46th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Networks, DSN 2016, pp. 654–661, Sep. 2016, doi: 10.1109/DSN.2016.66.

Authorized licensed use limited to: Malardalen University. Downloaded on February 24,2023 at 07:51:26 UTC from IEEE Xplore. Restrictions apply.

[5] "A Beginner's Guide to Logstash Grok | Logz.io." https://logz.io/blog/logstash-grok/ (accessed Apr. 24, 2022).

[6] W. Xuet al., "Detecting large-scale system problems by mining console logs," SOSP'09 - Proc. 22nd ACM SIGOPS Symp. Oper. Syst. Princ., pp. 117–131, 2009, doi: 10.1145/1629575.1629587.

[7] J. Zhu et al., "Tools and benchmarks for automated log parsing," ieeexplore.ieee.org, Accessed: Apr. 24, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8804456/.

[8] "Failover - Wikipedia." https://en.wikipedia.org/wiki/Failover (accessed Apr. 24, 2022).

[9] Q. Lin et al., "Log clustering based problem identification for online service systems," Proc. - Int. Conf. Softw. Eng., pp. 102–111, May 2016, doi: 10.1145/2889160.2889232.

[10] S. Nedelkoski et al., "Self-supervised Log Parsing," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 12460 LNAI, pp. 122–138, 2021, doi: 10.1007/978-3-030-67667-4_8.

[11] S. He et.al., "Experience report: System log analysis for anomaly detection," ieeexplore.ieee.org, 2016, doi: 10.1109/ISSRE.2016.21.

[12] A. Pavlo et al., "A comparison of approaches to large-scale data analysis," SIGMOD-PODS'09 - Proc. Int. Conf. Manag. Data 28th Symp. Princ. Database Syst., pp. 165–178, 2009, doi: 10.1145/1559845.1559865.

[13] Y. Pingleet al.,"Big data processing using apache hadoop in cloud system," Citeseer, Accessed: Apr. 24, 2022. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.470.7009&rep=rep1&type=pdf.

[14] "Apache Hadoop." https://hadoop.apache.org/ (accessed Apr. 24, 2022).

[15] M. Schultz et al., "Data mining methods for detection of new malicious executables," ieeexplore.ieee.org, Accessed: Apr. 24, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/924286/.

[16] M. Grottke et al., "Analysis of software aging in a Web server," IEEE Trans. Reliab., vol. 55, no. 3, pp. 411–420, Sep. 2006, doi: 10.1109/TR.2006.879609.

[17] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," Proc. Int. Conf. Dependable Syst. Networks, pp. 575–584, 2007, doi: 10.1109/DSN.2007.103.

[18] "logpai/loghub: A large collection of system log datasets for AI-powered log analytics." https://github.com/logpai/loghub (accessed Apr. 24, 2022).

[19] M. R. Ahmed et al., "The impact of software fault prediction in real-world application: An automated approach for software engineering," ACM Int. Conf. Proceeding Ser., pp. 247–251, Jan. 2020, doi: 10.1145/3379247.3379278

[20] L. Peng et al.,"Dynamic network anomaly detection system by using deep learning techniques." International Conference on Cloud Computing. Springer, Cham, 2019

[21] Shiravi et al. "Toward developing a systematic approach to generate benchmark datasets for intrusion detection." Computers & security 31.3 (2012): 357-374

[22] Zhang et al. "Network anomaly detection using one class support vector machine." Proceedings of the International MultiConference of Engineers and Computer Scientists. Vol. 1. 2008

[23] Duffield et al. "Systems and methods for rule-based anomaly detection on IP network flow." U.S. Patent No. 9,258,217. 9 Feb. 2016

[24] Jing Tian , Michael H. Azarian , Michael Pecht, "Anomaly detection using self-organizing maps-based k-nearest neighbor algorithm." Proceedings of the European Conference of the Prognostics and Health Management Society. Citeseer, 2014

[25] S. Osman, et al. "Anomaly detection in medical wireless sensor networks using SVM and linear regression models." International Journal of E-Health and Medical Communications (IJEHMC) 5.1, 2014.

[26] Neeraj Chugh, GS Tomar, RS Bhadoria, Neetesh Saxena, "A Novel Anomaly Behavior Detection Scheme for Mobile Ad-hoc Network", Electronics, Vol.10, No.14, pp.1-18, 2021

[27] B. Robert et al., "Information theoretic anomaly detection framework for web application." 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC). Vol. 2. IEEE, 2016.

[28] E. Hilmi et al., "Spectral anomaly detection using graph-based filtering for wireless sensor networks." 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2014.

Authorized licensed use limited to: Malardalen University. Downloaded on February 24,2023 at 07:51:26 UTC from IEEE Xplore. Restrictions apply.