INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

Impact Factor: 5.725

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 04, April 2024, pp: 213-217

AN SECURITY LOG ANALYSIS FOR IT SYSTEMS

Balasubramaniyan B[1], Revathi A[2], Dhanush S[3], Ukesh Kumar SB4

[1,3,4]Department of Computer Science (of Affiliation) Rathinam College of Arts and Science (of Affiliation) Coimbatore, India.

[2]Assistant professor Department of computer science Rathinam college of arts and science Coimbatore India.

ABSTRACT:

This research explores the realm of automated log analysis, focusing on the critical role of semi-structured software logs in ensuring the reliability of contemporary software systems. As software scales, the surge in log volume has driven research into efficient log utilization. The paper covers diverse topics, including automated logging statement generation, log compression, structured event template parsing, and the application of logs for anomaly detection, failure prediction, and diagnosis.

Additionally, the survey highlights open-source toolkits and datasets, offering insights into available resources for the research community. By discussing recent advancements, the paper identifies promising directions for both real-world applications and the next generation of automated log analysis. With CCS Concepts encompassing Software maintenance tools and Software creation and management, this survey provides a succinct overview, serving as a valuable resource for researchers and practitioners.

1. INTRODUCTION

In our daily lives, we use a lot of software without even realizing it - from the apps on our phones to the websites we visit. Imagine if suddenly these apps or websites stopped working; it could cause a lot of problems! That's why making sure software is reliable, meaning it works well all the time, is super important. To make sure everything is running smoothly, software keeps track of what it's doing through something called "logs." These logs are like notes or messages that tell us what the software is up to - whether it's running smoothly or if there's something going wrong.

Now, here's where it gets interesting. In the past, people used to read these notes (logs) manually to check if everything was okay. However, as you can imagine, with so much software out there, and each one generating lots of notes, it's become a bit like trying to read too many messages at once. It's just too much! That's where smart researchers come in. They are working on ways to teach computers to understand these notes (logs) automatically, without humans having to read each one. This is a big deal because software has become really big and complicated. The old way of checking logs just doesn't work well anymore. Now, think of it like a huge and busy kitchen with many chefs. Each chef writes notes about what they're cooking, and the menu changes frequently. Trying to keep track of everything manually would be impossible! It's a bit like what happens with software and its logs. Researchers are like the kitchen managers figuring out better and faster ways for the kitchen (software) to keep track of what's happening. They're making it easier for the chefs (developers) from all around the world to work together and cook up new things without causing a mess in the kitchen. It's all about making sure our digital world runs smoothly, just like a well-managed and efficient kitchen.

2. PROJECT OVERVIEW

A. Scope of the project

The Automated log analyzer have the potential to ensure the reliability and performance of modern software systems.

1) Reliability Assurance:

   The primary goal is to enhance the reliability of software systems. By analyzing logs, the analyzer helps identify and address issues promptly, minimizing downtime and potential revenue loss.

2) Anomaly Detection:

   Log analyzers are equipped to identify unusual or unexpected behavior in the software. This is vital for detecting potential problems or security threats that might otherwise go unnoticed.

3) Performance Monitoring:

   Log analyzers play a crucial role in monitoring the performance of software. They track how well the software is running, helping developers and administrators optimize and improve its efficiency.

4) Log Parsing into Structured Event Templates:

   Log analyzers aim to convert semi-structured log data into structured event templates, making it easier to organize and analyze information systematically.

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

Impact Factor: 5.725

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 04, April 2024, pp: 213-217

3. RELATED WORK

Wei Xu and collaborators [Xu et al. 2009b] uses machine learning to detect anomalies by mining log data. When an anomaly is detected, a decision tree visualization is presented to the user which attempts to explain why it is an anomaly. This setup, while useful, is fundamentally untrustable: no algorithm can successfully detect all anomalies, and the user will eventually have to resort to commandline grep. We adopt their machine learning features for our clustering algorithm. HP Operations Analytics4 provides many similar features to our work, including interactive time-window zooming. It does not seem to cluster any data, nor does it focus on software bugs, but instead analyze single-node server issues.

Seaview has the closest goals to ours [Hangal 2011], and provide similar visualizations. However, they do not work with arbitrary log types (notably, log messages with multiple variables), nor does it extend to multiple nodes or large amounts of data. They include no machine learning or reduction of the amount of data shown, other than a planned grep text search. Makanju and collaborators present spatio-temporal clustering of log data as well [Makanju et al. 2012]. Their method does not seem to be, nor claim to be, interactive. Finally, Saganowski and collaborators present a statistical method for preprocessing features that will be used in anomaly detection [Saganowski et al. 2013]. We find that these two methods for clustering, feature extraction, and feature preprocessing to be complementary to ours, and it would fit cleanly in our interactive system.

4. LOG ANALYSING

The general workflow of log mining involves four key steps: log partition, feature extraction, model training, and online deployment. In the log partition step, where modern software systems often produce interleaved logs, the challenge lies in segregating them into distinct groups. This is typically achieved through either timestamp-based partitioning, utilizing fixed or sliding windows, or log identifier-based partitioning, where tokens like user IDs or task IDs are employed. The chosen method significantly impacts log mining performance. Following log partitioning, the feature extraction step aims to derive relevant information from log groups, crucial for constructing effective log mining models. Subsequently, model training employs machine learning or statistical techniques to develop models based on the extracted features. Finally, in the online deployment phase, the trained models are implemented in real-time systems to continuously analyze logs. The success of log mining hinges on accurate log partitioning, effective feature extraction, and the seamless deployment of trained models for ongoing operational insights.

Graphical features play a crucial role in log mining by revealing hierarchical and sequential relations among system components and events through the construction of directed graph models. These models, depicting system behaviors such as the execution path of a process, serve as foundational elements for various log mining tasks, including monitoring and diagnosis. Existing work in graphical feature extraction can be broadly categorized into two lines of work. The first line involves leveraging identified objects in logs, such as process IDs and system component names, to construct graphs for system state monitoring. Sophisticated algorithms are employed to handle the intricate hierarchical relations among objects. For instance, Zhao et al. constructed a stack structure graph considering various mappings among different objects. The second line focuses on utilizing the statistical distribution of log events, such as the order of events and the temporal and spatial locality of dependent events, to build graphs for system behavior analysis. Some approaches aim to recover the exact behavioral model from log traces, requiring unique identifiers to tie together events associated with a program execution. Others, acknowledging the absence of unique identifiers in certain scenarios, mine behavioral models using interleaved logs.

Techniques such as statistical inference and nearest neighbor groups are applied to capture temporal dependencies and co-occurrence of log events, contributing to the construction of program Control Flow Graphs spanning distributed components. These graphical features provide a powerful basis for tasks like log-based behavioral differencing, system evolution, testing, and security applications.

Anomaly detection involves identifying unusual patterns in log data that deviate from expected behaviors, often indicating errors or faults in software systems. The approaches in this field can be broadly categorized into traditional machine learning algorithms and deep learning models. Traditional machine learning relies on features provided by practitioners, like log event count vectors. The task can be formalized and addressed using various algorithms, including clustering, classification, and regression. The surveyed approaches are summarized in Table 4, outlining the algorithm/model, features used, and whether the approach is unsupervised or online. Unsupervised approaches do not require labeled training data, and online approaches can process input data in a streaming fashion. This discussion focuses on general software anomalies impacting system reliability, excluding external attacks relevant to system security.

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 04, April 2024, pp: 213-217

Impact Factor: 5.725

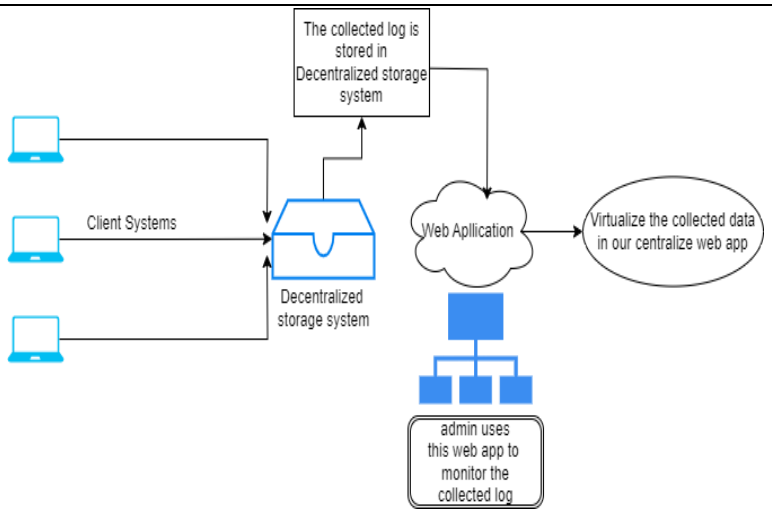

Fig 1- Proposed Architecture

SOFTWARE MODULES:

EXISTING SYSTEM

1. Log Managers
2. Anomaly Detectors
3. App Performance Trackers

PROPOSED SYSTEM

Remote monitoring from various location

1. Monitoring the IT systems as an easy one.
2. In proposed, Log monitor is used as a central hub.
3. Multi parameter monitoring system.

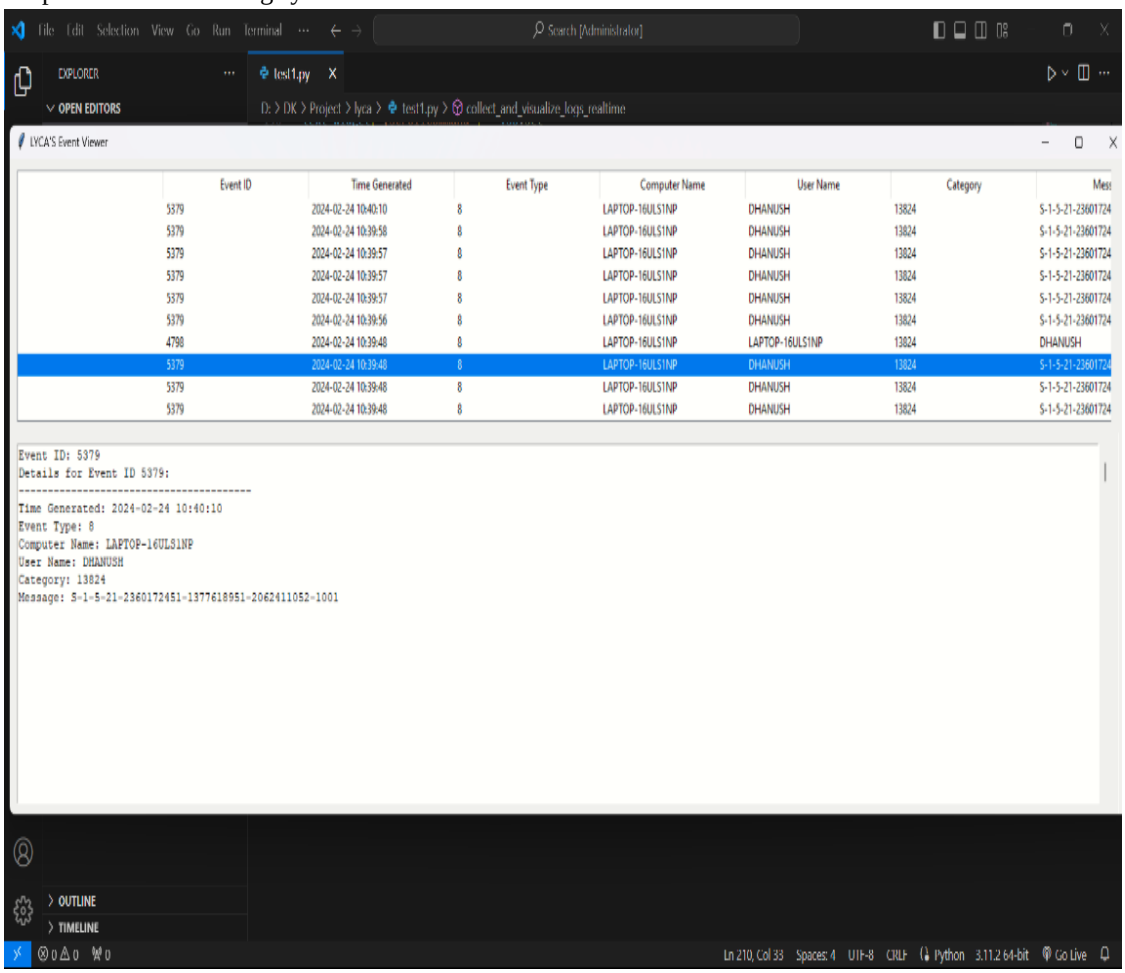

Fig 2 - Log Viewer

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

Impact Factor: 5.725

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 04, April 2024, pp: 213-217

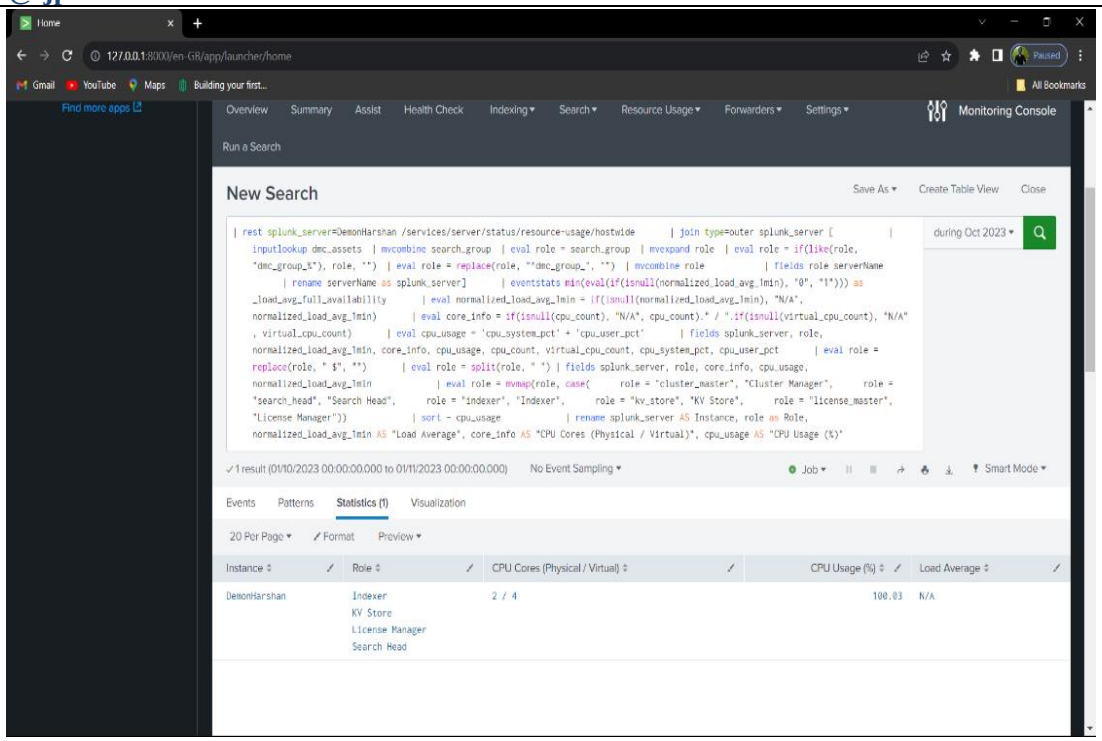

Fig 3 – Raw Data Log View

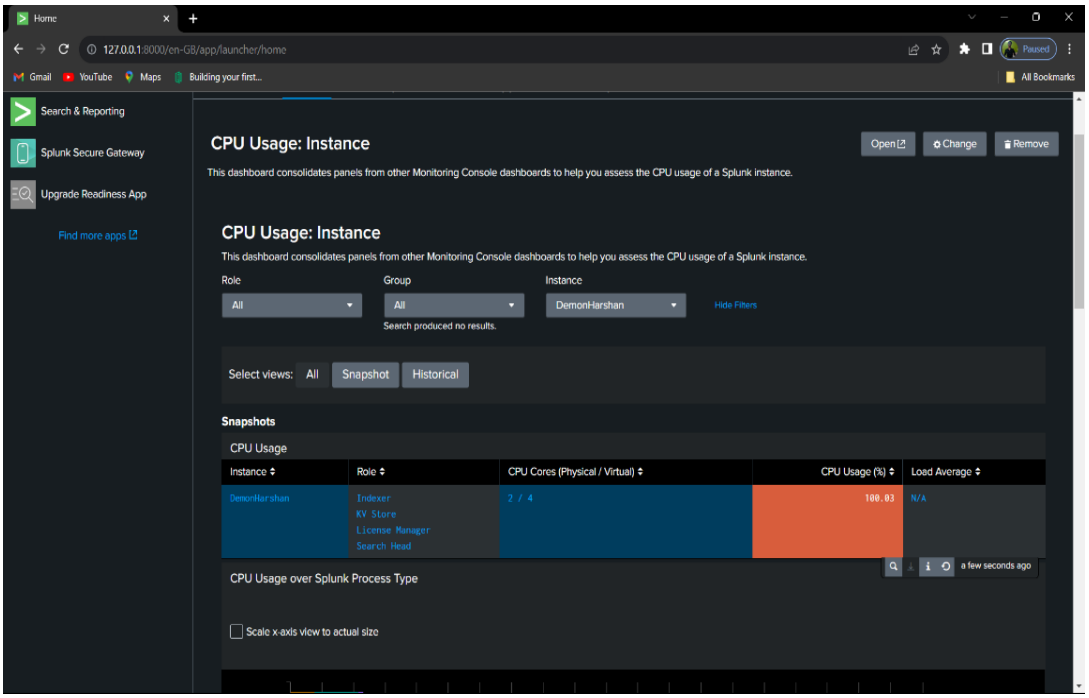

Fig 4

In the above output we have generated a system's log after this we are going to add this to a stable web application to show the different system's log in one place.

5. CONCLUSION

Recent years have witnessed the blossom of log analysis. Given the importance of software logs in reliability engineering, extensive efforts have been contributed to efficient and effective log analysis. This survey mainly explores the four main steps in automated log analysis framework: logging, log compression, log parsing, and log mining. Additionally, we introduce the available open-source toolkits and datasets. Our article enables the outsiders to step in this promising and practical field, and allows the experts to fill in the gaps of their knowledge background. Based on the investigation of these recent advances, we proposed new insights and discussed several future directions, including how to make automated log analysis more feasible under the modern agile and distributed development style, and the concept of a next-generation log analysis framework

6. REFERENCE

[1] Marcos K Aguilera, Jeffrey C Mogul, Janet L Wiener, Patrick Reynolds, and Athicha Muthitacharoen. 2003. Performance debugging for distributed systems of black boxes. ACM SIGOPS Operating Systems Review 37, 5 (2003), 74–89.

[2] AspectJ. 2020. Eclipse AspectJ. Retrieved September 1, 2020 [1].

[3] CFDR. 2020. Computer failure data repository. Retrieved September 1, 2020 [2].

[4] Daikon. 2020. A dynamic invariant detector. Retrieved September 1, 2020 [3].

[5] Failure dataset. 2020. Error logs produced by OpenStack. Retrieved September 1, 2020 [4].

[6] EDGAR. 2020. Apache log files. Retrieved September 1, 2020 [5].

[7] ELK. 2012. ElasticSearch. Retrieved September 1, 2020 [6].

[8] Facebook. 2019. Downtime, outages and failures - understanding their true costs [7].

[9] Facebook. 2019. Facebook loses $24,420 a minute during outages [8].

[10] Fluentd. 2020. An open source data collector for unified logging layer. Retrieved September 1, 2020 [9].

[11] GoAccess. 2020. A fast, terminal-based log analyzer. Retrieved September 1, 2020 [10].

[12] GrayLog. 2020. A leading centralized log management solution. Retrieved September 1, 2020 [11].

[13] Logalyze. 2020. An open source log management and network monitoring software. Retrieved September 1, 2020 [12].

[14] Loggly. 2009. Automated parsing log types. Retrieved September 1, 2020 [13].

[15] Loghub. 2020. A large collection of log datasets from various systems. Retrieved September 1, 2020 [14].

[16] LogPAI. 2020. A platform for log analytics powered by AI. Retrieved September 1, 2020 [15].

[17] Logstash. 2020. A server-side processor for log data. Retrieved September 1, 2020 [16].

[18] logz. 2014. Log parsing - automated, easy to use, and efficient. Retrieved September 1, 2020 [17].

[19] Microsoft. 2018. Event logging. Retrieved September 1, 2020 [18].

[20] Prometheus. 2020. A systems and service monitoring system. Retrieved September 1, 2020 [19].