System Log File Reduction and Detection of Malicious Behavior

An Independent Study
Submitted to the Faculty of
The Department of Electrical and Computer Engineering
Villanova University

By

Ralph Ritchey

In Partial Fulfillment
Of the Requirements for the Degree of
Master of Science in Cybersecurity

July 21, 2020

Abstract

Cybersecurity relies heavily on data for the detection of malicious behavior. Historically, intrusion detection systems (IDS) utilized sensors placed strategically within a network to monitor and capture network traffic. Various tools then process the network traffic in real-time or batch mode, generating alerts security analysts review. This methodology worked effectively until the use of encryption became prevalent for network traffic. Encrypted network traffic prevents signature-based IDS tools from inspecting packet payload contents to detect signatures indicating malicious activity or intent. As an alternative data source, system logs, and web server logs capture the indicators at the system level required by cybersecurity tools and analysts to detect possible malicious behavior. Research into log file size reduction while retaining and indicating log entries containing malicious activity is necessary to prevent overwhelming cybersecurity systems, tools, and analysts with too much data.

Contents

List of Figures

Chapter 1
Introduction

Network-based intrusion detection historically formed the backbone for protecting an organization's infrastructure. Strategically placed IDS sensors, as shown in Figure 1.1, monitored all network traffic flowing between the organization and the rest of the world. IDS sensor software, such as Snort [1], examined each network packet, including performing deep packet inspection into the payload of each network packet. Examining the payload allowed the IDS software to detect malicious activity and output an alert for an analyst to review. This sensing capability heavily relied on network traffic being unencrypted. The lack of packet encryption was native to the Internet, which was "…never intended to be secure or open for commercial use." [2]

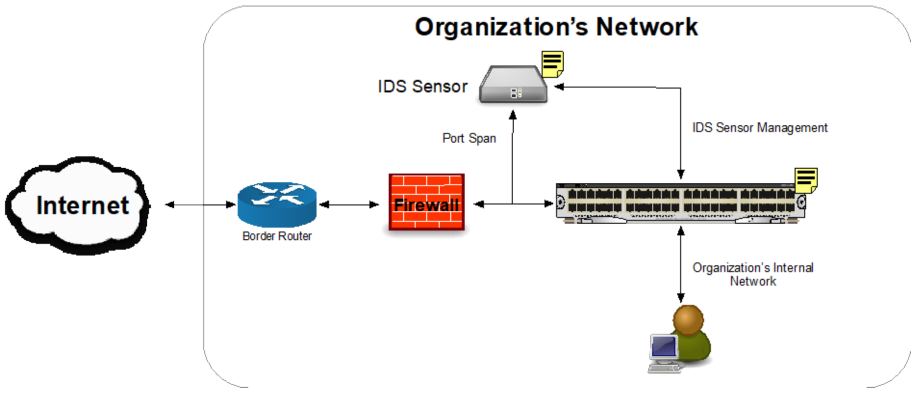

Figure 1.1: IDS Sensor Network Placement.

A study performed by Fortinet, Inc. in 2018 claims, "…encrypted traffic now represents over 72% of all network traffic…". [3] The study also indicates this represents an increase in encrypted traffic by 17% from the previous year. As encryption usage continues to increase, an IDS sensor's ability to perform deep packet inspection decreases, thereby reducing their effectiveness each year. Sensors are unable to inspect encrypted packet payloads unless the sensor has the necessary information to complete the decryption or dynamically break the encryption ("break and inspect"). Dynamically breaking encryption is difficult and computationally expensive, making this an impractical option, especially for high bandwidth environments. Cybersecurity must now look at alternative data sources to fulfill security monitoring.

Alternative data sources are logs generated by an organization's servers, desktops, and other computer-based devices. System logs "…consist of the voluminous intermixing of messages from many software components…" [4] executing on a device. These log files provide insight into the normal execution of software on the device. Anomalies in log entries, while used by systems administrators as indicators of a hardware or software fault, may also indicate defects triggered by malicious activity. For example, Najafabadi et al. [5] used web server logs to train a machine learning algorithm to detect distributed denial of service (DDoS) attacks.

The research project described in this paper proposes using log files, coupled with a machine learning algorithm, to reduce log file sizes and identify malicious log entries. The remainder of this paper contains sections covering a literature search and the planned approach for performing the research.

Chapter 2

Literature Review

Najafabadi et al. [5] used Principle Component Analysis (PCA) to detect DDoS attacks against a web server. The researchers split logged HTTP requests contained in log files based upon non-overlapping time windows and then sub-grouped the HTTP requests based on client IP addresses. The URLs accessed by the client during the time window processed populated the feature vector for that client record. Only URLs accessed by a minimum of 5 clients formed the feature vector. While successfully detecting DDoS attacks in web server log files, there are many other types of attacks requiring detection by cybersecurity analysts. Additionally, there is a wide range of log files from systems available as potential data sources beyond web server HTTP logs.

Xu's paper [4] focused on detecting anomalies in server system logs using PCA. In their use case, an anomaly is a change in behavior of a system indicated by log entries not conforming to prior learned behavior by a machine learning algorithm. Although the researchers do not apply their technique to detecting reportable cybersecurity events, anomaly detection can be a useful approach for identifying potentially malicious activity. To create feature vectors, the researchers statically analyzed the source code generating log entries, creating templates used to parse log files. The researchers acknowledged static analysis of source code is critical to the "…accuracy of our approach…" While feasible in small environments, access to source code generating log files across all devices deployed in an organization is highly unlikely. Additionally, new software or updates to software may break templates created from prior static analysis, requiring continual static analysis at a rate not sustainable in some environments. In a follow-up paper, their source code static analysis revealed "…20,000 different possible message types…" [6], forcing them to "…heuristically choose the message variables that are likely to indicate problems." Cybersecurity requires accurate detection, using a technique "likely" determining problems will result in too many false positives and overwhelm analysts in large organizations.

Juvonen et al. [7] applied random projection, PCA, and diffusion maps to HTTP log files for cybersecurity anomaly detection. The URL requests contained in log entries were broken into 2-grams to form the feature vector, which, in the worst-case scenario, can result in a vector of 256^2 dimensions. However, in a "…real-world situation, the actual number is much lower." Their results, using a small initial data set, show diffusion mapping, although taking longer to execute, identifying more attacks than PCA and random projection. After performing a similar experiment using more extensive data, the researchers only discussed performance speed. The researchers concluded diffusion maps and random projection work best as part of an ensemble configuration for anomaly detection.

A paper by Lee et al. [8] applied two PCA variations for anomaly detection using the KDD Cup 1999 data set. Although detailed data set transformation specifics did not appear in the paper, the authors claim 94-99.87% accuracy identifying four types of malicious activity. Execution time, critical for application in the real world, was approximately 34 seconds for osPCA and around 3 seconds for Online osPCA using 76,813 samples.

Although not using system log files, Abdulhammed et al. [9] applied PCA to the CICIDS2017 intrusion detection data set as part of an ensemble method to detect and classify malicious network traffic. After transforming the data, PCA reduced the number of features from 81 down to 10. The reduced feature set then fed into several machine learning classifier algorithms (random forest, Bayesian network, linear discriminant analysis, and quadratic discriminant analysis) to perform detection and classification.

Ritchey et al. [10] applied PCA to features extracted from a production Apache HTTP log file. PCA reduced the feature set permitting generation of a 3d graph, as shown in Figure 2.1, indicating possible outliers. The authors encountered difficulties attempting to map graph points back to specific log lines. This challenge caused the authors to switch to a Naïve Bayes based approach where they mapped results back to particular log lines. The Naïve Bayes algorithm trained using the same features used for PCA and generated a model. The trained model was then used to identify unusual log lines in the same log file falling below a probability threshold. Although useful for Apache HTTP logs, the approach was not as practical for outlier detection for Linux syslog files due to their highly repetitive nature.

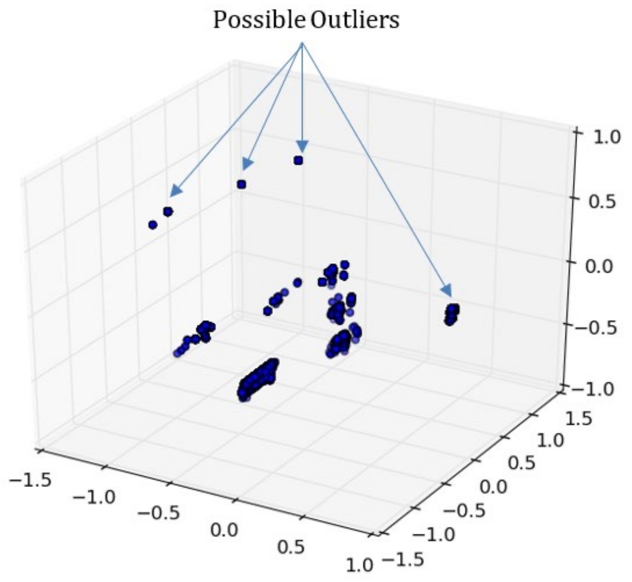

Figure 2.1 [10]: PCA Identification of Outliers (Anomalies).

Chapter 3

Proposed Research Approach

Based on the literature search results, there are alternative approaches available for exploring the application of PCA to log files that may be more suitable for cybersecurity purposes. This section provides the overarching goal of the proposed research, the proposed approach for performing the research, possible data sources, and the experimental procedure.

3.1 Research Goals

A research project should expand the body of knowledge and contribute to the scientific understanding of the research topic. Research provides additional value when results are directly applicable to real-world implementation. The goals listed below add to the scientific knowledge gained and increase the real-world applicability of the research project:

- *Deployable without significant manual tuning and maintenance:* A tool requiring substantial amounts of time to install, manually tune, and manually maintain is costly for an organization. Organizations are more likely to install and utilize solutions providing reliable detection results without requiring constant manual labor. The detection technique should self-tune as much as possible, requiring a minimum of manual intervention.

- *Scalable to support large enterprises and large log file sizes:* Organizations range from small network environments with a few servers and desktops to enterprises with hundreds or thousands of systems deployed globally. A tool resulting from the research should work effectively for small and large organizations.

- *Reasonable resource (CPU, memory, time, and disk I/O) utilization:* Systems capable of generating logs used by organizations vary from small, battery-powered portable devices to large, multi-CPU based servers and clusters installed in data centers. A tool applying the research results should impact each device's available resources as little as possible. Systems administrators and users will delete or disable tools utilizing a significant amount of resources that affect the usability of a device.

- *Ability to reduce log file sizes while retaining lines with potential malicious behavior:* System usage may result in log files becoming vast, complicating the ability to store or transport. A tool developed based on the research should facilitate reducing a log file's size while still retaining log lines indicating malicious behavior.

3.2 Data Sources

A challenge for cybersecurity-based research is the availability of suitable, publicly accessible data sets. For example, researchers heavily use the KDD Cup 99 [11] dataset for network-based intrusion detection research. While publicly available and used in published papers, criticism varies from the data set being based on "…previous datasets, which had additional identified flaws…" [12] to the age of the dataset as it "…is not representative of the types, scale or complexity of modern network traffic." [12] Looking more broadly at sharing cybersecurity-related data in general, privacy concerns come to the forefront as "…sharing could expose customer and employee personal data to increased privacy risk." [13]

A search using various tools such as Google revealed many possible data sets. The following are potential data sources for use during the research project:

- Apache HTTP logs from a publicly accessible website the researcher maintains for an iOS iPhone application. Approximately five years of log files are available, with over 420,000 unlabeled lines. There are no privacy concerns as the website is informational only and does not contain any privacy-related data.
- The Cyber Research Center from the West Point Military Academy provides the CDX 2009 data set. [14] This data set contains 6,100 web server log lines collected over 24 hours. The data set is unlabeled; however, the majority of the traffic is malicious because collection occurred during an exercise.
- The Austrian Institute of Technology (AIT) provides the 5.61GB AIT Log Data Set V1.0 [15] spanning six days. The data is a synthetically generated set of log files, providing logs from mail servers, Apache, syslogs, and others from a Linux based environment. Over half a million labeled Apache log lines are provided, along with just under half a million labeled syslog lines.
- The President & Fellows of Harvard College's Dataverse provides Farzin's 3.3GB Nginx web server log [16] containing over 10 million unlabeled log lines for an online shopping store. The authors do not provide information on how the data was generated.
- The HTTP Dataset CSIC 2010 [17] provides 55MB of log files containing 1.3 million labeled lines. The data was synthetically generated, mimicking an e-commerce web application, and includes normal and malicious access attempts.
- LOGPAI Loghub [18] provides a wide range of systems logs. The site offers operating systems logs from Windows, Linux, and macOS, as well as Apache, Hadoop, and others. The log data "…are production data released from previous studies, while some others are collected form real systems in our lab environment."

3.3 Coding Approach

Coding for the research project will use the Python [19] scripting language. Python, an interpreted scripting-based programming language, facilitates making fast code changes without the need to recompile after each code change. Although compiled languages such as C/C++ perform better under certain situations at runtime, the ability to quickly adjust code is more important for the research-based use case. (For real-world implementation, parts of the Python code may be optimized using Cython [20], which statically compiles Python code into C/C++ compiled objects (libraries).) Using Python allows the focus to remain on experiments and results without the potential time required for lengthy code compilation.

Python provides access to numerous libraries extending the base capabilities a default Python installation natively provides. Machine learning and data manipulation libraries such as scikit-learn [21], Numpy [22], and pandas [23] provide pre-coded access to machine learning algorithms and highly efficient data manipulation and management capabilities. The use of libraries such as these will allow additional time spent performing research to focus on feature extraction, feature engineering, machine learning tuning, and analysis of the results of code changes by significantly shortening the code development cycle.

A lightweight plugin approach will be used for the data processing and feature generation code. Having this code broken out into plugins permits numerous variations to exist and be quickly created to test new ideas. Additionally, this aids in maintaining a historical plugin code set, allowing scripted, re-execution against new data sets. Additional plugin types may be added if other code sections from the main script benefit from being implemented as plugins.

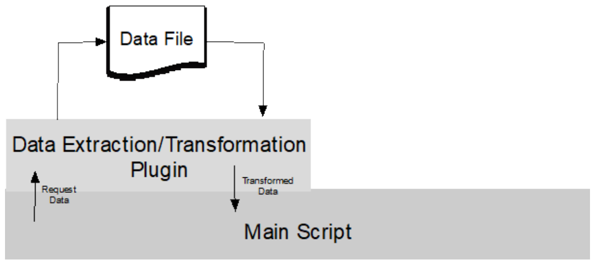

Figure 3.1: General Script Design

3.4 Hardware

The available systems for research experiment execution include:

System 1:
- Mac Pro (Late 2013)
- 3.5GHz 6-Core Intel Xeon E5
- 16GB 1866 MHz DDR3 memory
- MacOS 10.15.5 (Catalina)
- 4TB SSD

System 2:
- Home Built Server
- 3.7GHz 6-Core Intel Core i7 8700

- 32GB 2666MHz DDR4 memory
- Ubuntu 20.04 (Focal Fossa)
- 1TB SSD
- 11GB GDDR5 Nvidia GeForce RTX 2080 Ti

System 3:

- MacBook Pro (Late 2016)
- 2.9GHz Quad-Core Intel Core i7
- 16GB 2133MHz DDR3 memory
- MacOS 10.15.5 (Catalina)
- 2TB SSD

System 4:

- 12 node Villanova University College of Engineering High Performance Computing Cluster
- 128GB memory (per node)
- 20 2.4GHz cores (per node)

For portability reasons, initial coding and testing will use System 3 (laptop). Final experiment execution will use System 1 (Mac Pro) unless a machine learning algorithm benefits from the availability of a GPU, at which point experiment execution will use System 2 (Home Built Server). If the need arises for more memory or will benefit from parallelized processing, the Villanova University College of Engineering High Performance Computing Cluster (System 4) will be used.

3.5 Experimental Procedure

Careful planning and execution of experiments include the incorporation of the following guidelines:

- Reboot the system used for experiment execution just before executing the experiment. Rebooting ensures all experiment executions begin with the system in the same state.
- Capture and store all script experimentation output (logs, etc.) for future re-analysis and verification. In addition to providing necessary data for the paper, this permits careful reviews to ensure no mistakes were made during experiment execution.
- Ensure no extraneous processes or applications are running on the system before executing experiments. Irrelevant processes or applications may impact memory, CPU, and disk utilization as they compete for resources. This may introduce misleading differences in resource utilization and execution performance, which affects the analysis of results.
- Each experiment will be executed three times, gathering the same statistics and information generated from each run for inclusion in charts, graphs, and any calculations provided in the resulting research paper.

Experiments will focus on using PCA as the selected machine learning algorithm and variations on feature extraction from the data set. One of the listed goals of the research is log file reduction and identification of potentially malicious behavior. This equates to anomaly detection, for which others have utilized PCA across a wide variety of data in other research projects. As an unsupervised machine learning algorithm, this satisfies another goal where the implementation of the research is deployable with a minimum level of effort into pre-existing environments. Based on the literature search results, the proposed research has not been exhausted by prior research.

3.6 Machine Learning

The research project will use PCA as the selected machine learning algorithm. PCA is an unsupervised machine learning algorithm, making it an ideal candidate for the research project and potential real-world deployment. Unsupervised machine learning algorithms do not require labeled data. Data in the real-world is unlabeled and requires a significant investment in resources (people and time) for an organization to process, review, and generate labels for training. Experiments, on the other hand, may use both labeled and unlabeled data sets depending on the experiment and data set(s) meeting the experiment requirements. Labeled data allows verification of experiment results against expected results, whereas unlabeled data provides insights into how well PCA identifies potentially malicious log entries under a more realistic scenario.

PCA transforms a given data set by reducing the data set's dimensionality (number of features). The premise is if features in a data set are highly correlated, it "…implies that the 'true' dimension of the dataset is less…" [24], therefore, removal of those highly correlated features (dimensionality reduction) will still "…convey virtually all of the information in the original…" [24] data set. For example, Figure 3.2 [25] shows the graph of a 2-dimensional data set on the left. By moving the axes as shown on the right, the "…potential for dimensionality reduction is in the fact that the *y* dimension does not now demonstrate much variability, and so it might be possible to ignore it…". [25] The reorientation of the axes in this example permits the reduction of the dimensionality of the feature set from 2-dimensions to 1-dimension. Elimination of dimensions reduces noise in the data, which "…can make the results better…". [25]

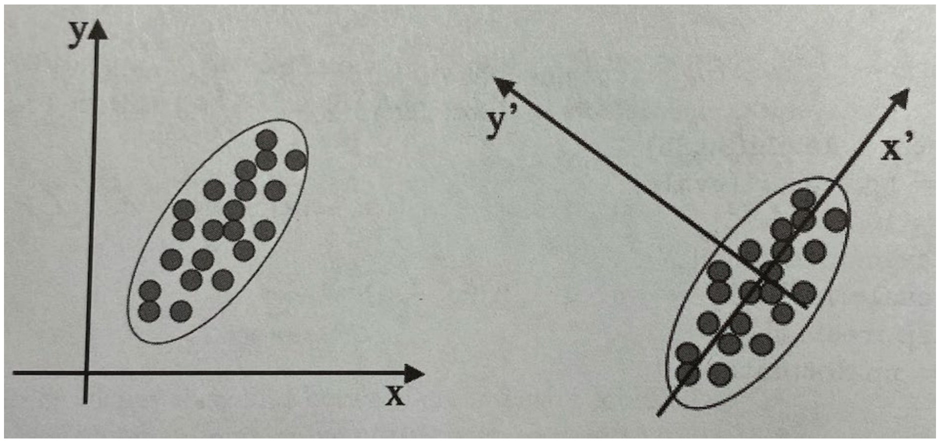

Figure 3.2 [25]: Visual Example of PCA

The first step in performing PCA calculates the mean-deviation form. For a matrix *X* with *m* samples and *n* features (axes), calculate the mean for each axis by summing all the sample values and dividing by the number of samples:

$$M_j = \frac{1}{m}\left(X_{1j} + \cdots + X_{mj}\right)$$

Create a new *m x n* matrix *B* with the normalized values:

$$B_{ij} = X_{ij} - M_j$$

The next step calculates the singular-value decomposition (SVD) of *B* resulting in the *m x n* column-orthonormal matrix *U*, the *n x n* column-orthonormal matrix *V*, and the *n x n* diagonal matrix *S* containing the singular values of *B* (assuming $m > n$).

$$B = U\,S\,V^T$$

The columns of *V* are eigenvectors of the covariance matrix B^TB and the corresponding eigenvalues are the squares of the singular values.

After computing the singular-value decomposition, the *t* smallest singular-values can be ignored, and the remaining $r = n - t$ columns of *U* and *V* form the reduced dimensional model.

References

[1] Cisco, "Snort - Network Intrusion Detection & Prevention System," [Online]. Available: https://www.snort.org. [Accessed 26 June 2020].

[2] Y. Y. Al-Salqan, "Future Trends In Internet Security," in *Proceedings of the Sixth IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, Tunis, 1997.

[3] Fortinet, Inc., "As the Holday Season Draws Near, Mobile Malware Attacks Are Prevalent," 14 November 2018. [Online]. Available: https://www.fortinet.com/blog/industry-trends/as-the-holiday-season-draws-near--mobile-malware-attacks-are-pre. [Accessed 26 June 2020].

[4] W. Xu, L. Huang, A. Fox, D. Patterson and M. I. Jordan, "Detecting Large-Scale System Problems by Mining Console Logs," in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating System Principles*, Big Sky, MT, 2009.

[5] M. M. Najafabadi, T. M. Khoshgoftaar, C. Calvert and C. Kemp, "User Behavior Anomaly Detection for Application Layer DDoS Attacks," in *2017 IEEE International Conference on Information Reuse and Integration*, San Diego, CA, 2017.

[6] W. Xu, L. Huang, A. Fox, D. Patterson and M. Jordan, "Experience Mining Google's Production Console Logs," in *SLAML'10: Proceedings of the 2010 Workshop on Managing Systems via Log Analysis and Machine Learning Techniques*, Vancouver, BC, 2010.

[7] A. Juvonen, T. Sipola and T. Hamalainen, "Online Anomaly Detection Using Dimensionality Reduction Techniques for HTTP Log Analysis," *Computer Networks,* vol. 91, pp. 45-56, November 2015.

[8] Y.-J. Lee, Y.-R. Yeh and Y.-C. F. Wang, "Anomaly Detection via Online Oversampling Principal Component Analysis," *IEEE Transactions on Knowledge and Data Engineering,* vol. 25, no. 7, pp. 1460-1470, 2013.

[9] R. Abdulhammed, M. Faezipour, H. Musafer and A. Abuzneid, "Efficient Network Intrusion Detection Using PCA-Based Dimensionality Reduction of Features," in *2019 International Symposium on Networks, Computers and Communications (ISNCC)*, Istanbul, Turkey, 2019.

[10] R. P. Ritchey, G. G. Shearer and K. D. Renard, "Naive Bayes Log File Reduction and Analysis," US Army Research Laboratory, Adelphi, MD, 2019.

[11] T. U. K. Archive, "KDD Cup 1999 Data," 28 October 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. [Accessed 2 July 2020].

[12] L. F. Sikos and K.-K. R. Choo, Data Science in Cybersecurity and Cyberthreat Intelligence, Springer, 2020, p. 118.

[13] J. Bhatia, T. D. Breaux, L. Friedberg, H. Hibshi and D. Smullen, "Privacy Risk in Cybersecurity Data Sharing," in *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*, Vienna, Austria, 2016.

[14] United States Military Academy, West Point, "Cyber Research Center - Data Sets," [Online]. Available: https://www.westpoint.edu/centers-and-research/cyber-research-center/data-sets. [Accessed 2 July 2020].

[15] L. Max, S. Florian, W. Markus, H. Wolfgang and R. Andreas, "AIT Log Data Set v1.0 | Zenodo," 21 March 2020. [Online]. Available: https://zenodo.org/record/3723083#.Xv4HVy2z3UK. [Accessed 2 July 2020].

[16] Z. Farzin, "Online Shopping Store - Web Server Logs - Harvard Dataverse," 10 December 2018. [Online]. Available: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/3QBYB5. [Accessed 2 July 2020].

[17] C. T. Gimenez, A. P. Villegas and G. A. Maranon, "HTTP Dataset CSIC 2010," 20 January 2012. [Online]. Available: https://www.isi.csic.es/dataset/. [Accessed 2 July 2020].

[18] J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng and M. R. Lyu, "logpai/loghub: A large collection of system log datasets for AI-powered log analytics," 2019. [Online]. Available: https://github.com/logpai/loghub. [Accessed 2 July 2020].

[19] Python Software Foundation, "Welcome to Python," [Online]. Available: https://www.python.org. [Accessed 6 July 2020].

[20] S. Behnel, R. Bradshaw, L. Dalcin, M. Florisson, V. Makarov and D. S. Seljebotn, "Cython: C-Extensions for Python," [Online]. Available: https://cython.org. [Accessed 6 July 2020].

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau and Brucher, "scikit-learn: Machine Learning in Python," [Online]. Available: https://scikit-learn.org/stable/index.html. [Accessed 06 July 2020].

[22] NumPy, "NumPy," [Online]. Available: https://numpy.org. [Accessed 6 July 2020].

[23] NumFOCUS, "pandas - Python Data Analysis Library," [Online]. Available: https://pandas.pydata.org. [Accessed 6 July 2020].

[24] I. Jolliffe, "Principal Component Analysis: A Beginner's Guide - I. Introduction and Application," *Weather,* vol. 45, pp. 375-382, 1990.

[25] S. Marsland, "6.2 Principal Component Analysis (PCA)," in *Machine Learning An Algorithmic Perspective*, Boca Raton, FL, CRC Press, 2015, pp. 133-137.