PredictDeep: Security Analytics as a Service for Anomaly Detection and Prediction

MARWA A. ELSAYED, (Member, IEEE), AND MOHAMMAD ZULKERNINE, (Senior Member, IEEE)
School of Computing, Queen’s University, Kingston, ON K7L 2N8, Canada
Corresponding author: Marwa A. Elsayed (marwa@cs.queensu.ca)

This research work is partially supported by the Natural Sciences & Engineering Research Council of Canada (NSERC) and Canada Research Chairs (CRC).

ABSTRACT As businesses embrace digitization, the Internet of Everything (IoE) begins to take shape and the Cloud continues to empower new innovations for big data—at the heart, Cloud analytic applications gain increasing momentum. Such applications have remarkable benefits for big data processing, making it easy, fast, scalable, and cost-effective; albeit, they pose many security risks. Security breaches causing anomalous activities due to malicious, vulnerable, or misconfigured analytic applications are considered the top security risks to big “sensitive” data. The risk is further expanded from the coupling of data analytics with the Cloud. Towards maintaining secure and trustworthy applications, effective anomaly detection and prediction become crucial tasks to be offered by Cloud providers. This paper presents, PredictDeep, a novel security analytics framework for anomaly detection and prediction. The proposed framework leverages log data collected from monitoring systems with graph analytics and deep learning techniques to add intelligence for detecting and predicting known and unknown patterns of security anomalies. It represents the collected data and transforms them into a graph model. The graph model captures the analytical activities as well as their interrelation. In this sense, such a model provides informed insight of the monitored application, understanding its behavior, and revealing anomalous patterns. Different from existing traditional rule-based machine learning and statistics-based approaches, our solution takes the benefits of incorporating not only available node attributes but also graph structure and context information to extract rich features that boost the anomaly classification and prediction. We leverage graph embeddings to represent the nodes and relationships in the graph model as feature vectors to learn and predict anomalies in an inductive way utilizing recent advanced deep graph neural network techniques. This design augments our solution with robustness and computational efficiency. Extensive experiments are conducted over an open-source Hadoop log dataset. The evaluation results demonstrate that PredictDeep is a viable solution and very effective.

INDEX TERMS Anomaly detection, deep learning, graph analytics, security service, log analysis, security analytics, big data security, analytic application security.

I. INTRODUCTION

Recent reports reveal that the security front line represents an ever-expanding surface for potential attacks [1]. The proliferation of smart devices creates more endpoints to protect. The Cloud is expanding the security perimeter. In addition, users are, and always will be, a weak link in the security chain. As businesses embrace digitization—and the Internet of Everything (IoE) begins to take shape—real-time anomaly detection and prediction become a crucial task towards maintaining secure and trustworthy systems. Cloud computing is empowering new innovations for big data. At the heart, Cloud analytic applications are playing a core role in several industries including online banking, ecommerce, healthcare, and automotive. They have remarkable benefits for big data processing, making it easy, fast, scalable and cost-effective; albeit, they pose many security risks.

Security breaches causing anomalous activities due to malicious, vulnerable, or misconfigured analytic applications are considered the top security risks to big “sensitive” data [2].

The risk is further expanded from the coupling of data analytics with the Cloud. Furthermore, the unique features of Cloud analytic applications, which run on distributed
large-scale heterogeneous computing environments, render traditional rule-based machine learning and statistics-based solutions ineffective. Effective security measures, delivered by Cloud analytics providers, to detect and predict such malicious and anomalous activities at runtime are still missing.

Hadoop, the most-shiny analytics technology, was not originally designed with security, compliance, and risk management support in mind. Recently, it is evolved to support authentication and encryption mechanisms for protecting data at rest and in transit. Despite the evolving efforts in securing Hadoop, it is still exposed to weak authentication and infrastructure attacks. Such attacks increase the security risk of analytic applications running on Hadoop against data confidentiality and integrity.

Log data is considered a rich source of information for troubleshooting and performance debugging issues. However, the distinct features of executed computations and processed data in distributed large-scale dynamic analytic systems arise several challenges to develop an effective machine learning-based log analysis for anomaly detection and prediction solution. These challenges entail a) handling log data collected across the cluster nodes that is usually characterized by the 4Vs (i.e., volume, velocity, variety, and veracity); b) mining for tangible evidence of security anomalies from log data which typically has unstructured complex confounding format; c) considering the different roles of core daemons responsible for running analytic applications; d) capturing complex data and control flows instigated among the cluster nodes during executing analytic applications; e) overcoming the limitations of existing machine learning techniques in handling complex data and extracting rich features needed for better prediction capability; f) avoiding asymmetric errors that can arise as a result of training the learning model over data which typically exhibits imbalanced distribution of benign and anomalous activities; and g) empowering timely detection and prediction of security breaches to promote real-time decision-making and countermeasure responses.

Nowadays, big data analytics and machine learning are gaining increasing momentum in the security field. As the risk of cyber-attacks grows in sophistication and frequency, security analytics grow in importance and potential even more. In this paper, we are inspired to propose, PredictDeep, a novel security analytics framework for anomaly detection and prediction. We leverage streaming data analytics with graph analysis and deep learning to boost our framework in solving the aforesaid challenges.

The proposed framework integrates with Security Monitoring as a Service (SMaaS) framework, our previously proposed monitoring systems [51]. SMaaS leverages streaming data analytics to automate the collection, management, processing, analysis, and visualization of log data from multiple sources, making it valuable, comprehensive, and cohesive for security inspection [51]. It is mainly responsible for detecting common patterns of security anomalies at real-time. However, PredictDeep complements the security posture of big data systems by fostering the detection and prediction of evolved and uncommon patterns of anomalies. This is achieved by integrating PredictDeep and SMaaS to work together in a harmony.

On one hand, SMaaS consolidates the unstructured heterogeneous log data (i.e., text), emitted from various sources (e.g., log4j and Syslog) of a monitored cluster (e.g., Hadoop), into a semi-structured format (i.e., JSON “JavaScript Object Notation”) representing an information flow profile. This profile is useful to model the normal behavior of a monitored system and detect known patterns of security anomalies, tackling the first three challenges (a-c) [51]. In what follows, we will refer to the log data and the extracted profile as the gleaned data. On the other hand, PredictDeep utilizes the gleaned data and combines graph analytics and advanced deep learning techniques to add intelligence for predicting and detecting evolved and unknown patterns of security anomalies, addressing the remaining challenges (d-g).

In particular, PredictDeep models the gleaned data as a graph that captures the activities conducted in the inspected application as well as their interdependency and linkage relationships. In this sense, such a model can provide informed insight of the monitored application and clear understanding of its behavior. While this tends to bring enormous benefits to improve predicting and revealing hidden anomalies, it imposes significant limitations on existing machine learning techniques. A graph model representing large-scale distributed analytic application normally exhibits complexity. This complexity stems from the graph dynamic size as the nodes evolve and maintain variable number of relations, and nearby neighbors. Such a complexity renders vital learning tasks such as convolution in traditional neural network impractical [29]–[32]. Further, existing techniques assume that the system’s entities are independent of each other. However, the entities (nodes) in the graph model have range of relations that need to be considered. Towards overcoming these limitations, PredictDeep employs graph analytics in a way that allows for extracting rich features essential for the learning analysis. Hence, it leverages graph convolutional networks (GCNs), one of the advanced graph neural network techniques. The GCNs basically extend the deep learning technique to graph data, where the convolution task is performed by aggregating the features of neighboring nodes and applying sampling strategies to conduct the convolution computation over a batch of nodes rather than the whole graph towards improving the practicability and scalability. In this sense, the GCN technique allows detecting and predicting complex non-linear relationships as well as drawing significant insights to uncover hidden anomalies.

Despite the growing importance of effective online anomaly detection for big data systems, there are few studies [13]–[16] reflecting the lack of attention in the research community. Existing approaches span two main directions: statistics-based and machine learning-based analysis. Approaches [13], [14], adopt the former analysis, work in offline settings that hinder their effectiveness to provide...
online protection. As a remedy, other approaches [15], [16] embrace the latter analysis and leverage traditional neural network techniques. These approaches [15], [16] are capable of capturing linear and non-linear relationships considering time-related context to reflect short-term or long-term recurring patterns useful for pinpointing anomalies. However, they fall short to capture interrelation and linkage dependencies between data instances. This, in turn, causes a gap in existing research work to overcome the aforesaid challenges.

Our overall contributions are as follows: 1) We propose PredictDeep, a novel framework of Security Analytics as a Service for anomaly detection and prediction; 2) We introduce an advanced approach that leverages streaming data analytics with graph analysis and deep learning to add intelligence for predicting unseen patterns of anomalies; and 3) We demonstrate the prediction effectiveness and performance efficiency of our framework through a set of experiments over benchmark dataset.

The remainder of this paper is organized as follows: Section II outlines the related work. The operational overview of our proposed framework and the threat model it assumes are introduced in Section III. Section V presents the details of the framework. The framework implementation and experimental evaluation are presented in Section VI. Section VII draws the concluding remarks of the paper and future work.

II. RELATED WORK
Over the last decades, anomaly detection has been extensively studied in diverse research domains ranging from distributed systems, computer networks, social networks, surveillance systems, among others. It has been widely applied for various purposes not only confined for security [13]–[16] but also extended for system performance failure and reliability [18]–[23]. For interested readers, there are various surveys highlighting the state-of-the-art of anomaly detection as data outliers within high-dimensional datasets (e.g., [24], [25]) or as fraudulent activities within IT systems (e.g., [26]).

In this section, we focus the discussion on the most relevant approaches addressing log-based anomaly detection and big data system security.

Log-based Anomaly Detection. It simply revolves around the task of uncovering anomalies that do not conform with the normal/expected patterns within log dataset, where the definition of anomalies is tuned based on the context or the application domain. Accordingly, most researchers have cast this task as data analysis problem. Even though existing approaches actively rely on various techniques of statistics [13], [14], and machine learning (ML) [15], [16] based analysis, graph-based data analysis has not received the same attention.

Statistics and ML based approaches mainly assume that the observed data points are independent and uniformly distributed within the dataset, in contrast, graph-based analysis approaches consider the interrelated correlation between the data points which facilitate deeper and wider coverage for detecting anomalous patterns. Broadly speaking, the workflow of log-based anomaly detection approaches commonly adopts two main tasks: log parsing and log mining.

Log Parsing. Some approaches [13]–[15] exhibit similarity with our approach (PredicDeep) with respect to the log parsing task. After parsing the log entries and identifying the unique log keys (activities), session vectors are extracted to reflect the frequency of the occurrence of these activities within the session, where the order of the activities is sorted according to the normal execution path. Differently, DeepLog [16] uses the order of log keys (activities) to formulate the session vectors representing the inspected execution path. There are other approaches proposed to perform online log parsing. These approaches have been extensively studied and evaluated in a recent study [3]. Among them, Drain [4] and Spell [5] are publicly available parsers that can be used to save implementation efforts and time for the purpose of performing more sophisticated parsing tasks, if needed.

Log Mining. Each anomaly detection approach significantly differs in the subsequent log mining task. Xu et al. [13] require the source code of the inspected system, whose availability is impractical in real settings, in order to directly infer the log messages. They employ one of the popular statistical techniques called PCA (Principal Component Analysis). The PCA is used to extract the repetitive patterns within the parsed “high-dimensional” log data as “low-dimensional” summaries called principle component and hence detect deviations as abnormal activities.

Lou et al. [14] similarly carry out statistics-based approach using IM (Invariant Mining) technique. They identify program invariants from console logs to model the constant linear characteristics of the workflow within the inspected system and pinpoint deviations as execution anomalies.

Both approaches [13], [14] tend to unsupervised learning style; however, they are restricted to a linear dimensionality reduction. They are effective in the context of their problem domains, albeit they work in offline settings that cannot be applied for online protection in security-critical systems. Specifically, they assume that the log data is available in advance before the security inspection, thus, they cannot cope with the dynamic nature of continuously running systems like analytic applications.

Other approaches follow machine learning style for online anomaly detection. Lu et al. [15] transform the parsed data instances into embeddings following a popular approach called word2vec [17] initially proposed by Google. Then, the produced embeddings are fed into CNN (Convolutional Neural Network) to learn the probabilities that correspond to each class and accordingly identify anomalous instances. The CNN technique is mainly capable of discovering short-recurring patterns within a dataset that are presumed to discriminate between normal and anomalous instances.

DeepLog [16] rather employs Long short-term memory (LSTM) as an instance of Recurrent Neural Network
(RNN) technique. This technique attempts to reconstruct its input through the implementation of an autoencoder. The encoder is responsible for transforming the input into compressed representations, while the decoder learns to reconstruct the input back from the representations. This is done by learning the long-term relationships, taking the advantage of time-related context, and automatically extracting the effect of past sequences while optimizing the reconstructing error. Since the LSTM in DeepLog is merely trained on normal data, it fails to reconstruct anomalous instances. Hence, the data instances exhibiting high reconstruction errors are identified as anomalies. Even though these approaches [15], [16] support linear and nonlinear relationships within data instances, they fall short to capture interrelation and linkage dependencies between them. To overcome this limitation, PredictDeep is inspired to embrace graph-based ML-driven approach. This is essential to effectively model the activities taking place in analytic applications which run on distributed large-scale big data systems/cluster. We extended the concept of labeled property graph for modeling the interrelated dependencies including data and control flows between the involved entities in the monitored system. This concept has the advantage to deliver a powerful structure which is intuitive and simple albeit expressive to model such a dynamic system and capture the involved entities, their respective wide-range correlations along with their properties. The property graph has been studied by few works [27], [28], however, they provide multiple definitions and flavors.

Graph Neural Networks. Despite the benefits graph analytics can bring to the anomaly detection problem, the complexity of these graph models renders traditional machine learning techniques impractical [29]–[32]. Thus, several approaches have recently emerged to revolutionize deep learning for graph data resulting on introducing the notion of graph neural networks [29]–[31] and further extending convolution operation to graph neural networks [32]. Early approaches [29]–[32] inclined to handle the whole graph at once and thus struggle to scale for large graphs. Recent approaches improve the scalability by performing the computations on a batch of nodes and transforming the information from neighboring nodes rather than the whole graph to learn what is called “node embeddings”. Most of these approaches focus on learning node embeddings in transductive manner assuming a fixed graph (e.g., [33], [34]). As an improvement, advanced approaches extend learning the node embeddings in an inductive manner to operate on evolving graphs that encounter unseen nodes (e.g., [12], [35]). LGCN [35] relies on standard CNN to extract node embeddings based on a ranking strategy over the neighboring nodes’ features. Differently, GraphSage [12] assembles the embeddings based on an aggregation strategy from the neighboring nodes’ features and further supports flexibility in terms of various flavors of the aggregation function. Driven by the dynamic nature of analytic systems which lead to evolved and dynamic graphs, an inductive-based GCN is more relevant, practical, and effective for our solution. We extend GraphSage and incorporate not only node attributes but also graph structure and context information to enrich the node embeddings. This is essential to boost PredictDeep to generalize its capability for detecting and predicting unknown as well as evolved patterns of anomalies.

Big Data System Security. Further to log-based anomaly detection approaches, there are other research efforts share the same goal for improving the security of big data systems. These approaches span different research directions, ranging from differential privacy [36], integrity verification [37]–[40], policy enforcement [41]–[44], data provenance [45]–[48], honeypot-based [49], to encryption-based [50] mechanisms, Security Monitoring [51], among others.

Despite the fact that differential privacy mechanism recently attracted researchers for specific contexts [36], its effectiveness as a widespread solution is still not testified. Integrity verification [37]–[40] approaches differ in the scope of integrity assurance, the logical layer of operation, and the mode of checking. In general, they require intercepting the computation of MapReduce tasks for verifying result integrity, which comes at the cost of performance penalty. Some other approaches enforce security policies for access control at different granularities that pose modifications for the underlying platform without providing end-to-end protection [41], [42]. As an alternative, accountability mechanisms [43], [44] are proposed to harden access control policies and verify that data access happened after authorization complies with the security policies. Few approaches [45]–[48] embrace data provenance mechanism to keep history about data for the purpose of reproducibility. They face several challenges that may hinder its practicability such as the volume of captured provenance data, the storage and integration required to effectively analyze these data, and the most important factor is the overhead incurred from collecting these data during the execution of distributed analytic applications. Other approach [49] takes on honeypot-based mechanism to detect unauthorized access in MapReduce. A different approach [50] leverages encryption to protect data at the cost of imposing performance burden and reduction of system operations.

Our recent approach (SMaaS) [51] proposes an advanced security monitoring service that leverages streaming big data pipeline and Cloud technologies as key enablers to elude the limitations in the aforementioned solutions. However, SMaaS focuses on detecting common patterns of security anomalies for analytic applications. As an improvement, PredictDeep combines graph analytics and deep learning techniques to add the intelligence needed to detect and predict unseen and evolved patterns of anomalies.

III. OVERVIEW

This section outlines the operational overview of our proposed framework, the threat model it assumes, and a brief preliminary background.
A. THE PredictDeep OPERATIONAL OVERVIEW
The Cloud service model considered for this work consists of three main entities: 1) Cloud analytics provider offering analytics technologies (e.g., Hadoop) which can range from basic services (e.g., IaaS, PaaS) for building analytics clusters, to tailored services (e.g., Data Analytics as a Service) for performing specific analytical tasks; 2) Trusted party offering various security services (e.g., SMaaS and PredictDeep); and 3) Cloud consumers running their analytic applications over the provided cluster in the Cloud. Such applications are designed to collect, analyze, and transform big data for business intelligence (BI).

We propose PredictDeep as an advanced anomaly prediction and detection security feature offered by the analytics provider through a trusted party.

As depicted in Fig. 1, a provider for analytics technology (e.g., Hadoop) offers its consumers the option to subscribe for an advanced anomaly detection and prediction security service (1). For subscribed consumers, the provider enables monitoring the clusters respective these consumers. The provider delegates the trusted party for further analyzing the gleaned data to predict and detect security anomalies (2).

PredictDeep uses log data collected by the monitoring system (e.g., SMaaS). The trusted party, in turn, employs the proposed PredictDeep framework to detect and predict unknown and evolved anomalous and malicious activities indicating security breaches.

To enable consumers plan for effective incident response measures, detailed reporting of the analysis as well as activity statistics and trends are published in the consumers’ dashboard along with email alerts sent upon uncovering security anomalies (3).

B. THREAT MODEL
Analytic applications (e.g., MapReduce jobs, spark jobs, and Hadoop Distributed File System (HDFS) jobs) can be misconfigured, malicious, or vulnerable and may breach the security of the processed data. This can be done throughout their execution via multiple activities (e.g., modifying, copying, or deleting data) at different levels (i.e., input, intermediate results, output) in a way that violates data integrity and confidentiality.

Our solution aims to detect known as well as unknown patterns of security anomalies indicating security violations. We assume the correctness and integrity of log files where attackers cannot inject adversarial inputs to mislead our security analysis and cause inaccurate prediction. We also assume the security of the cluster, the underlying platform, and infrastructure where PredictDeep is deployed on.

C. PRELIMINARIES
A typical big data analytics cluster empowered by Hadoop can be configured to generate auditing and logging data from different disjoint sources (e.g., applications, daemons, audit actions) with various severity levels (e.g., DEBUG, WARN, INFO) for many purposes (e.g., debugging, auditing, operational stats). In this respect, such log data typically has a complex confounding unstructured format which is burdensome
to mine with the goal of fostering the detection and prediction of known and unknown security anomalies.

Formally, the log data is a file that records the execution and access activities which take place during the runtime of a monitored system (e.g., Hadoop daemons and services). These activities are typically logged along with a set of attributes such as the timestamp and the severity level. For example, HDFS logs record all data access activity within Hadoop, while a MapReduce log contains entries for all analytical jobs submitted and executed on the cluster. As illustrated in Fig. 2 (a), a log record represents a single entry $e$ in a log file which mainly consists of two parts: constant and variable. The former, a.k.a. log key, describes the recorded execution or access activity $A$ (e.g., received block) as a fixed text message while the latter contains runtime parameters such as the identifier of the executed or accessed entity $E$ (e.g., data block id).

An entity identifier can be useful to classify multiple log entries into a single session representing the execution and access sequence relative to this entity. In other words, a session $S_E$ is a set of log entries $(e_1, e_2, \ldots, e_n)$ grouped by the same entity $E$ to capture its sequence of activities $A_E=(A_1, A_2, \ldots, A_k)$, where $k$ is the number of unique activities appeared in a log file. The log entries shown in Fig. 2 (b) formulate a session covering the sequence of activities conducted over the data block identified as $blk\_160899687919862906$.

This sequence includes three activities: receiving block, received block, and block updated.

In this sense, log parsing is required as a preprocessing task for our machine learning driven system (PredictDeep) to analyze raw log data collected directly from the monitored system using typical utilities like Log4j and Syslog. The log parsing task is needed to identify the activities and their associated entities in structured format from the unstructured logs. This structured data enables grouping the activities having common entities in separate sessions and associate the start time of the first activity of this sequence as the session’s timestamp. This is crucial to facilitate the subsequent learning task, as outlined later in Section IV. We mainly leverage pattern recognition in the form of regular expressions for parsing the log data based on our domain knowledge of the format of log data.

IV. THE FRAMEWORK

As illustrated in Fig. 3, the proposed framework consists of three main components: Graph Model Designer, Feature Extractor, and Anomaly Predictor. The graph model designer is responsible for modelling the gleaned data as a graph capturing the properties of the cluster’s nodes as well as their dependencies and linkage relationships. Then, the feature extractor component receives the graph model as an input and leverage graph analytics to extract rich features covering not only the node properties but also the graph structure and context information. These features augment the anomaly predictor component to employ a graph convolutional network technique and learn node embeddings in unsupervised manner. These embeddings are the key enabler of applying machine learning techniques for predicting and detecting nodes exhibiting malicious and anomalous activities. These components logically form the prediction engine for predicting and detecting evolved and unknown patterns of security anomalies. The components are further detailed in the following subsections.

A. GRAPH MODEL DESIGNER

The graph model designer is responsible for modeling the gleaned data as a graph. This component receives the log data in unstructured format. This, in turn, requires parsing the raw log data into structured format, as highlighted earlier in Section III. This is achieved by representing the log entries as sessions, where each session reflects the sequence of activities conducted over a respective entity (e.g., data block, Map job, and Reduce job). Given that the number of activities $k$ is constant, each session is encoded as a fixed-length vector $1 \times k$, where each vector dimension matches specific activity and the value of each dimension corresponds to the frequency of the respective activity within the session.

Then, the graph model designer represents the structured data as a dynamic labeled property graph defined according to

![Figure 2. Log snippets demonstrating: (a) example of an activity and an entity; (b) example of a session.](image-url)
Definition 1. The graph consists of nodes representing entities involved in a monitored application and edges indicating the relationships and interactions among these entities.

The graph nodes are marked with labels according to their identifiers (e.g., data block id, Map job id, and Spark job id) and are also associated with their session vectors as node properties.

The relationships between the entities capture different aspects ranging from dependencies representing the control flow and data flow during the execution of the monitored analytic application, to temporal reflecting the timestamp associated with their sessions. This is mainly provided by the consolidated profile from SMaaS to further enrich the graph model and incorporate comprehensive relationships and node properties. In addition, any key-value pair associated with an entity in the profile (i.e., JSON) is combined into the set of node properties of the respective entity in the graph.

In this respect, the graph model acts as a valuable structure offering a new method to uncover anomalous and malicious activities. It can uncover patterns that are difficult to detect using traditional representations of the gleaned data. This can be realized by looking beyond the individual entities, to the connections that relate them.

Definition 1 (Graph): Cleaned data in our settings is modeled as a directed dynamic labeled property graph $G = (N; E; L; P)$, where $N$ is a finite set of nodes representing the entities, $E$ is a finite set of edges representing interactions among the entities, $L$ is an infinite set of labels, and $P$ is an infinite set of properties such that: 1) $\mathcal{F}_E : E \rightarrow (N \times N)$ is a total function that points each $e_{ij} \in E$ from one node $n_i$ to another $n_j$ if the timestamps of their sessions are chronological $T_{A_i} < T_{A_j}$ or there exists an information flow between $n_i$ and $n_j$, where such a flow can be explicit due to data dependency or implicit due to control dependency; 2) $\mathcal{F}_L : N \rightarrow L$ is a partial function which marks each node with a set of labels $l \in L$; and 3) $\mathcal{F}_P : N \rightarrow P$ is a partial function which associates each node with a set of properties $p \in P$.

Definition 2 (Graph Adjacency Matrix): The graph adjacency matrix $A$ is a $n \times n$ matrix with $a_{i,j} > 0$ if and only if $\exists e_{i,j} \in E$, where $a_{i,j} \neq a_{j,i}$.

**B. FEATURE EXTRACTOR**

The feature extractor component receives the graph model as an input and produces the features for all nodes as a matrix $X$ according to Definition 3. Such a matrix is needed for the anomaly predictor component.

Definition 3 (Node Features): For a graph $G = (N; E)$, the node features are extracted as $X \in \mathbb{R}^n$, where $X$ is a feature matrix $n \times X_i$ and $X_i$ is the feature vector for each node $n_i \in N$.

The features are produced by incorporating not only node properties but also graph structures and context information. The graph analytics is essential to improve the machine learning accuracy by extracting rich contextual features useful for better prediction. In specific, we leverage various graph algorithms to evaluate the contextual features. These algorithms fall under two broad categories at the heart of graph analytics: centrality computation and community detection. The details of the steps taken by this component are presented in Algorithm 1 and outlined as follows:

**Centrality Computation Algorithms**

There are various measures for computing centrality in graph analytics. However, each centrality-based feature provides a different perspective about the role of the modeled entities and their impact on the monitored cluster. This is crucial to distinguish the behavior of benign and malicious entities in a monitored cluster. In what follows, these features are further outlined.

**Degree Centrality.** We employ the degree centrality algorithm [6] (line 3) to capture an overview of the connectivity of each node in terms of its incoming and outcoming relationships. The degree centrality reflects the popularity of each node based merely on its direct relationships. This measure captures the activities conducted by each node considering both the communications received by the node and tasks performed by the node. It is defined according to Equation (1), where $ND_i$ is the node degree that represents the number of nodes $(n_i \ldots n_r \in N)$ that are adjacent to node $n_i$ according to the graph adjacency matrix, denoted in Definition 2. The degree centrality can be normalized by dividing the node degree over the maximum possible connections a node can have in a graph.
of $n$ nodes. In this sense, the degree measure is essential to provide a sense of the exposure of the cluster to identify nodes having direct influence and indicating possible targets for collecting information or accessing resources on the cluster. This can help in distinguishing nodes that exhibit malicious or anomalous activity.

$$DC_i = ND_i/(n - 1) = \sum_j a_{i,j}/(n - 1)$$ (1)

Eigen Centrality. We leverage the eigen centrality algorithm [7] (line 4) to gain an insight of the influencing role of each graph node on the monitored cluster. In particular, we calculate the page rank measure [8] as a variant of the eigen centrality. This measure takes into account not only the incoming and outgoing relationships of a node but also the extended transitive connectivity of a node. The page rank measure $PR_i$ of node $n_i$ depends on the page rank $PR_j$ of each node $n_j$ belonging to the nodes $C_j$ connecting to node $n_i$. This measure is denoted according to Equation (2), where $d$ is a damping factor that can be set between 0 and 1 and $n$ is the total number of nodes in the graph. The page rank measure can be calculated as an eigenvector problem using the power iteration method. The algorithm generates random vectors and multiplies them by an adjacency matrix representing the graph until it converges when a maximum iteration number or an error tolerance number is reached, and the corresponding eigenvalue is deemed found. The page rank measure is an essential indicator of direct as well as indirect influence of each graph node over its neighbors (i.e., one hop away) and their neighbors (i.e., more than one hop away) throughout the cluster. This, in turn, is useful to identify unusual connectedness of colluding nodes which tend to have higher centrality in contrast to benign nodes. It also helps in understanding the cluster dynamics in terms of the propagation of malicious activities, the characteristics of each involved entity, and the connectivity between the entities.

$$PR_i = \frac{1 - d}{n} + d \sum_{j \in C_i} PR_j$$ (2)

Betweenness Centrality. We measure the betweenness centrality metric [9] (line 5) to gain a different insight of the influence of each node over the flow of information or resources in a monitored cluster. This metric mainly characterizes the control points which are the most influential bridges between the graph nodes. It is typically measured by counting the frequency each node $n_i \in N$ appears as a bridge on the shortest paths between other graph nodes $n_s, n_t \in N$. This is done by considering the number of all the shortest paths $C(s, t)$ between other nodes ($n_s$ and $n_t$), such that: $s \neq t \neq i$ and enumerating how many times a node $n_i$ lies on one $C(s, t|i)$, as indicated in Equation (3). Thus, the betweenness centrality is helpful in detecting potential points of vulnerability in a cluster.

$$BC_i = \sum_{i \neq s \neq t, i \in N} C(s, t|i)/C(s, t)$$ (3)

Community Detection Algorithms

We go beyond the centrality computation measures and examine features that are more specific to node connectivity to characterize the global graph structure. Such features are essential for evaluating the behavior of connected communities and infer isolated communities. This helps in revealing uncommon phenomena that indicates violation of the normal community boundaries in the monitored system. The following is a more detailed explanation of these features.

Clustering Coefficient. We leverage the clustering coefficient algorithm [10] (line 6) to measure the likelihood of communities as well as the cohesiveness of the overall cluster. The computation of the clustering coefficient typically involves the triangle count and the node degree, as outlined in Equation (4). The triangle count $\tau_i$ is the number of triangles incident to a node $n_i$ and the node degree $ND_i$ is the number of links to a node $n_i$. In this sense, the clustering coefficient measure can exhibit statistical differences which is useful to differentiate between benign and malicious nodes.

$$CC_i = 2\tau_i/ND_i(ND_i - 1)$$ (4)

Connected Components. We apply the connected component algorithm [11] (line 1) to capture the graph’s structure. The connected component partitions common nodes forming communities in the graph, which in turn is useful for the analysis of anomaly detection. This is intuitive as anomalous nodes tend to disregard common patterns of benign communities. In specific, this measure represents distinct connected components (a.k.a. communities), where every node in a community can be reached from other nodes in the same community and cannot be connected to other nodes in a
different community. For each community, we associate the community density (line 9) as a feature for each node $CD_i$ in the respective community.

C. ANOMALY PREDICTOR

The anomaly predictor component is responsible for learning and predicting unknown anomalies in an inductive way utilizing recent advanced deep graph convolutional neural network (GCN) techniques. It accepts as input the graph model $G = (N, E)$ and the extracted features $X$ which comprise of the node properties as well as the graph contextual features. It produces node embeddings that facilitates applying subsequent learning tasks such as classification and prediction of security anomalies.

We employ GraphSage [12] as one of GCN techniques. This technique utilizes the extracted features in order to learn an embedding function that generalizes to unknown nodes. It maps the graph nodes into a low-dimensional vector space (a.k.a. embeddings), while preserving both graph topology structure and node properties. The learning task is performed in an unsupervised manner using batch-training GCN technique. The convolution operation is performed by sampling the neighbors of each graph node $n_i$ into fixed-size batches $\beta(i)$, aggregating $\Psi$ their features $h_b$, and concatenating $\Theta$ them with the node’s own feature $h_0$ to drive the node’s final state $h'_i$, which in turn is optimized using error backpropagation through the hidden layers forming the GCN.

As presented in Equation (5), at each hidden layer $t$, the features $h'^{-1}_i$ form the next layer’s features $h'^{t}$ using the propagation rule $\sigma$ representing a non-linear activation function and $W'^{t}$ representing the weight matrix for layer $t$. In this sense, the GCN boosts our solution to achieve the scalability needed to handle the highly complex and informative structure of our graph model. After learning the node embeddings, the anomaly predictor component becomes ready to automatically operate a machine learning pipeline. The pipeline enables the embeddings to be transformed and correlated together in a model to discover the structures within them as well as predict and detect any deviations indicating security anomalies.

$$h'^{t}_i = \sigma \left( W'^{t} \cdot \Theta \left( h'^{-1}_i, \Psi \left( \left\{ h'^{-1}_b, \forall b \in \beta_i \right\} \right) \right) \right)$$

(5)

V. EXPERIMENTAL EVALUATION

Our evaluation aims at assessing the effectiveness of PredictDeep for predicting and detecting unknown security anomalies in big data systems. Experiments are conducted in order to evaluate various aspects: 1) compare our solution with other existing solutions; 2) study the impact of applying various aggregation methods used in building the GCN with different architectural flavors; and 3) examine whether the graph analytics improve the effectiveness of our solution in predicting unseen anomalies. The following sections present the experimental setup, evaluation method, and the evaluation results of the experiments conducted to assess each aspect.

A. EXPERIMENTAL SETUP

We implemented our proposed solution in Python relying on various set of libraries including a) Pandas\(^1\) and Numpy\(^2\) for the acquisition and processing of the log data; b) Scikit-Learn\(^3\) and TensorFlow\(^4\) for applying the machine and deep learning techniques, and e) NetworkX\(^5\) for performing the graph analysis.

We extended GraphSage [12] for the implementation of inductive-based GCN. We run PredictDeep inside Docker platform which is deployed on a VM with “Ubuntu 16.04.6”, 16 GB RAM, 4 CPUs and 100 GB storage that serves as a private Cloud server.

We used a benchmark dataset for anomaly detection in Hadoop Distributed File System (HDFS). HDFS is a system for high-throughput access to big data analyzed by analytic applications. An analytic application performing MapReduce job breaks the input data into multiple splits, equivalent in size to HDFS block. Then, the application breaks the processing into two main phases: Map and Reduce. The map phase is responsible for mapping the input data into key/value pairs forming intermediate results. Multiple map tasks are initiated on cluster’s nodes to simultaneously process each input split. The reduce phase takes the intermediate key/value pairs and produces the final output. Multiple reduce tasks are commenced to process all relevant intermediate pairs together in parallel to perform the required processing task.

The dataset represents a benchmark used in several related studies [13]–[16]. Xu et al. [13] generated the dataset by collecting HDFS logs during running MapReduce analytic applications over a cluster consisting of 203 nodes hosted on Amazon Elastic Compute Cloud (EC2). The collected HDFS raw log data is 1.56 GB in size and contains over 11 million (11,197,705) log entries, where roughly 3% reflects anomalous activities and the rest reflects normal activities. Recall that the log parsing converts the gleaned data into structured format, the structured data is modeled into a graph of 575,059 nodes representing the entity sessions. The sessions are labeled by Hadoop domain experts with 558,221 normal sessions and 16,838 anomalous sessions [13]. Each session is encoded as a sequence of 29 unique activities that can be conducted over HDFS’s data blocks during a job execution. After employing our solution (PredictDeep), the feature vector of each entity (data block) is extended to 34 features covering the session feature vector as well as the contextual and graph structure features. This results in node feature matrix X having a dimension of 575,059 $\times$ 34.

B. ANOMALY PREDICTION EVALUATION METHOD

PredictDeep learns node embeddings in an unsupervised manner that facilitate applying subsequent learning tasks.

\(^1\)Pandas API. https://pandas.pydata.org/

\(^2\)Numpy API. https://numpy.org/

\(^3\)Scikit-Learn API. https://scikit-learn.org/

\(^4\)TensorFlow API. https://www.tensorflow.org/

\(^5\)NetworkX API. https://pypi.org/project/networkx/
In these experiments, we provide the learned node embeddings and their labels to logistic regression classifier optimized by the stochastic gradient descent (SGD) algorithm. We then leverage the data labels that exist within the dataset to validate our testing results. We undertake cross-validation approach for the out-of-sample evaluation of PredictDeep.

We evaluate our system in terms of baseline predictive metrics including recall, precision, F-measure, and false alarm rate. These metrics are defined according to Definition 4-7 as follows:

**Definition 4 (Recall):** Recall is the fraction of anomalies that are correctly predicted out of the total number of actual anomalies. This metric is also known as True Positive Rate (TPR). It is calculated as \( \text{Recall} = \frac{TP}{TP + FN} \).

**Definition 5 (Precision):** Precision is the fraction of correctly predicted anomalies out of the number of the predicted anomalies. It is measured as \( \text{Precision} = \frac{TP}{TP + FP} \).

**Definition 6 (F-Measure):** F-measure is computed as the weighted harmonic mean of precision and recall. \( \text{F-measure} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \).

**Definition 7 (False Alarm Rate):** The fraction of incorrectly predicted anomalies out of the total number of actual false anomalies. This metric is also known as False Positive Rate (FPR) and measured as \( \text{FPR} = \frac{FP}{FP + TN} \).

The terms TP, FP, TN, and FN point to true positive, false positive, true negative, and false negative, respectively. They are defined as follows: a) true positive (TP): the number of anomalies correctly identified; b) false positive (FP): the number of anomalies incorrectly identified; c) true negative (TN): the number of anomalies correctly rejected; and d) false negative (FN): the number of anomalies incorrectly rejected.

**C. CONTRAST PredictDeep WITH OTHER SOLUTIONS**

We assess the effectiveness of PredictDeep and contrast the results with other relevant existing solutions [13]–[16]. The
FIGURE 6. The confusion matrices of training PredictDeep over various GCN architectures based on the following aggregation functions: (a) mean; (b) convolutional; (c) max-pooling; and (d) LSTM.

FIGURE 7. The confusion matrices of training PredictDeep without incorporating contextual features in the best two performing GCN architectures: (a) mean and (b) LSTM.
source of the results of the existing solutions come from publicly published data [13]–[16]. Following the same settings undertaken by these solutions, we split the dataset into roughly 1% for training and 99% for testing. Out of the testing set, we exclude small percentage (9%) used for validating and choosing the hyperparameter values, which include the learning rate,\(^6\) the batch size,\(^7\) the epoch parameter,\(^8\) and the model dimension, needed to configure the GCN. This results in a subgraph with 6,381 nodes for the training, a subgraph with 50,092 nodes for the validation, and a subgraph with 518,586 nodes for testing, chosen in a completely random way. In such a case, we do not control the percentage of benign or anomalous classes in each set. This is crucial for two main reasons: a) reflect the real-world settings, where naturally there is imbalance distribution between benign and anomalous activities in a monitored system; and b) achieve online learning to cope with the dynamic nature of a monitored system, where its graph continuously evolves with new nodes (data instances).

We select the mean aggregation function for the GCN architecture and configure the learning rate to be 0.00001, the batch size to be 512, the epoch parameter to be 10, and the model dimension to be 128. We then apply stochastic gradient descent (SGD) algorithm to optimize a classifier to predict the anomalous data instances based on the extracted node embeddings. Unlike gradient descent algorithm which considers the whole training data, the SGD estimates the gradient loss each data sample at a time and allows minibatch learning by considering only random sample of data points while changing the weights and updating the model along the way with a decreasing learning rate. As such the SGD supports the scalability and speed needed to handle large graphs that model big data systems. As depicted in Fig. 4, PredictDeep outperforms existing solutions [13]–[16] achieving F-measure of 99% with both recall and precision rate of 98.7%.

D. THE IMPACT OF VARIOUS AGGREGATION ARCHITECTURES

This experiment studies if the aggregation method used to learn the node embeddings has an impact on the effectiveness of PredictDeep. Each aggregation method differs in the search depth or the number of hops considered when aggregating the features from a node’s local neighbors to learn the node embeddings. The employed GCN technique (GraphSage [12]) supports various aggregation methods which adopt different artificial neural network architectures including mean, convolutional, pooling, and Long short-term memory (LSTM). We follow the same hyperparameter configurations used in the first experiment and utilize the same training and testing datasets. We here employ each GCN’s architecture separately and gauge the effectiveness of PredictDeep in terms of the metrics in question after predicting anomalous activities using the SGD classifier outlined in the first experiment.

As illustrated in Fig. 5, the mean and LSTM based GCN architectures achieve the best result in comparison to other GCN architectures. The results are further broken down in Fig. 6 showing the confusion matrix obtained at each case. The mean-based GCN shows strong prediction capability obtaining F1-measure of 99%, following LSTM-based GCN scoring F-measure of 94% at the second place.

E. INVESTIGATE THE EFFECT OF GRAPH ANALYTICS

This experiment examines whether the graph analysis improves the effectiveness of our solution in predicting unseen anomalies to validate our intuition. We assess the sensitivity of PredictDeep with respect to the contextual features obtained from the graph analytics, we follow the same configuration settings in Experiment 1.

We conduct PredictDeep analysis without incorporating the contextual features leveraging only two convolutional architectures: mean and LSTM. We choose these architectures as they achieve the best results in Experiment 2.

As inferred from Fig. 7, both cases show decrease in their effectiveness. In the case of mean-based GCN, the F1-measure sharply diminishes to 93% in contrast to the 99% achieved when incorporating the contextual features. On the other hand, the LSTM-based GCN exhibit slight drop in the F1-measure from 94% to 92% when trained without the contextual features. The results proof the importance of graph structure and contextual features to boost our solution to achieve strong prediction. This is aligned with our intuition, where it is well known that relationships are often the strongest predictors of behavior. It is worth mentioning that throughout the three experiments, PredictDeep achieve a very negligible false alarm rate that reaches 0.003 at most, further demonstrating the high effectiveness.

VI. CONCLUSION

In this paper, we introduce, PredictDeep, a novel framework for anomaly prediction and detection in big data systems. The PredictDeep framework is offered as an advanced security analytics service from a trusted party to the consumers of Cloud analytics providers.

PredictDeep leverages the advantages of streaming data analytics combined with graph analytics and deep learning techniques to mine log data collected from monitoring systems looking for hidden anomalous and suspicious activities.

The contributions of our approach revolve around overcoming the major challenges of developing an effective machine learning-based log analysis for anomaly detection and prediction for big data systems, outlined as follows:
1) Handle the Complexity of big data systems. This is achieved by extending the concept of labeled property graph to model not only linear and non-linear relationships but also the interrelated correlations covering data and control dependencies among the entities processed throughout the runtime of the monitored application. This model allows capturing semantic features locally from the node properties as well as graph structure and contextual features which together represent a rich source of information.

2) Tackle the Limitations of Traditional Machine Learning Techniques. Such techniques may stand incapable in front of mining such a complex graph model and be prone to the naturally inherent imbalance distribution of anomaly and benign observations within the data. PredictDeep leverages graph convolutional networks (GCNs) to extend the capabilities of mining such a complex graph model and be prone to benign observations within the data. PredictDeep is augmented by the streaming capability of graph data with additional focus on the temporal evolution and induce the embeddings of unseen nodes. This, in turn, allows the model to generate embeddings within the data.

These contributions are validated throughout experimental evaluation demonstrating the effectiveness of our framework in predicting anomalous activities. The experiments are conducted over open-source benchmark Hadoop log dataset for anomaly detection in HDFS. The experimental results reflect the benefits gained by PredictDeep in handling the complexity of graph data, standing immune against the asymmetric errors in imbalanced data, and providing online scalable capabilities. The results demonstrate that our system not only attains high prediction capability but also outperforms existing solutions.

PredictDeep is built on the streaming capability of SMaaS in collecting and managing heterogenous data from various sources (e.g., Hadoop and system logs). We plan to investigate different deep learning techniques for graph data with additional focus on the temporal evolution and parallelization to further mine such heterogeneous data. Future work may also include extending our solution to predict and classify anomalies into multiple categories based on the change in the system behavior they cause.

REFERENCES

[1] Check Point Software Technology. 2019 Cyber Security Report. Accessed: 2019. [Online]. Available: https://pages.checkpoint.com/cyber-security-report-2019.html

[2] Big Data Security and Privacy Handbook: 100 Best Practices in Big Data Security and Privacy. Cloud Security Alliance, Seattle, WA, USA, 2016.

[3] J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, “Tools and benchmarks for automated log parsing,” in Proc. IEEE/ACM 41st Conf. Softw. Eng., Softw. Eng. Pract. (ICSE-SEIP), May 2019, pp. 121–130.

[4] C. Freeman, “Centrality in social networks conceptual clarification,” Social Netw., vol. 1, no. 3, pp. 215–239, Jan. 1978.

[5] M. Du and F. Li, “Spell: Streaming parsing of system event logs,” in Proc. IEEE 16th Int. Conf. Data Mining (ICDM), Dec. 2016, pp. 859–864.

[6] L. C. Freeman, “Centrality in social networks conceptual clarification,” Social Netw., vol. 1, no. 3, pp. 215–239, Jan. 1978.

[7] P. Bonacich, “Power and centrality: A family of measures,” Amer. J. Sociol., vol. 92, no. 5, pp. 1170–1182, Mar. 1987.

[8] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 1024–1034.

[9] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, “Detecting large-scale system problems by mining console logs,” in Proc. ACM Symp. Operating Syst. Princip. (SOSP), 2009, pp. 117–132.

[10] J. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, “Mining invariants from console logs for system problem detection,” in Proc. USENIX Annu. Tech. Conf. (ATC), 2010, pp. 231–244.

[11] B. A. Galler and M. J. Fisher, “An improved equivalence algorithm,” Commun. ACM, vol. 7, no. 5, pp. 301–303, May 1964.

[12] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 1024–1034.

[13] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in Proc. Adv. Neural Inf. Process. Syst., 2013, pp. 3111–3119.

[14] K. Zhang, J. Xu, M. R. Min, G. Jiang, K. Pelechrinis, and H. Zhang, “Automated IT system failure prediction: A deep learning approach,” in Proc. IEEE Int. Conf. Big Data (Big Data), Dec. 2016, pp. 1291–1300.

[15] X. Yu, P. Joshi, J. Xu, G. Jin, H. Zhang, and G. Jiang, “CloudSeer: Work-flow monitoring of cloud infrastructures via interleaved logs,” in Proc. ACM Int. Conf. Architectural Support Program. Lang. Operating Syst. (ASPLOS), 2016, pp. 489–502.

[16] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, “Log clustering based problem identification for online service systems,” in Proc. 38th Int. Conf. Softw. Eng. Companion (ICSE), 2016, pp. 102–111.

[17] A. Pecchia, D. Cotroneo, Z. Kalbarchczyk, and R. K. Iyer, “Improving log-based field failure data analysis of multi-node computing systems,” in Proc. IEEE/IFIP 41st Int. Conf. Dependable Syst. Netw. (DSN), Jun. 2011, pp. 97–108.

[18] S. Banerjee, H. Srikanth, and B. Cukic, “Log-based reliability analysis of software as a service (SaaS),” in Proc. IEEE 21st Int. Symp. Softw. Rel. Eng., Nov. 2010, pp. 239–248.

[19] M. Farshchi, J.-G. Schneider, L. Weber, and J. Grundy, “Experience report: Anomaly detection of cloud application operations using log and cloud metric correlation analysis,” in Proc. IEEE 26th Int. Symp. Softw. Rel. Eng. (ISSRE), Nov. 2015, pp. 24–34.

[20] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection for discrete sequences: A survey,” IEEE Trans. Knowl. Data Eng., vol. 24, no. 5, pp. 823–839, May 2012.

[21] A. Taha and A. S. Hadji, “Anomaly detection methods for categorical data: A review,” ACM Comput. Surv., vol. 52, no. 2, pp. 1–35, May 2019.
[26] U. Flegel, J. Vayssiere, and G. Bitz, “A state of the art survey of fraud detection technology,” in Proc. Insider Threats Cyber Secur., vol. 49. Berlin, Germany: Springer, 2010, pp. 73–84.

[27] R. Angles, “The property graph database model,” in Proc. AMW. CEUR Workshop, 2018, p. 2100.

[28] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaeker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, and A. Taylor, “Cypher: An evolving query language for property graphs,” in Proc. SIGMOD, 2018, pp. 1433–1445.

[29] M. Gori, G. Monfardini, and F. Scarselli, “A new model for learning in graph domains,” in Proc. IEEE Int. Joint Conf. Neural Netw., vol. 2, Aug. 2005, pp. 729–734.

[30] A. Micheli, “Neural network for graphs: A constructive contextual approach,” IEEE Trans. Neural Netw., vol. 20, no. 3, pp. 498–511, Mar. 2009.

[31] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” IEEE Trans. Neural Netw., vol. 20, no. 1, pp. 61–80, Jan. 2009.

[32] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” in Proc. Int. Conf. Learn. Representations, 2014, pp. 1–14.

[33] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, “Community preserving network embedding,” in Proc. 31st AAAI Conf. Artif. Intell., 2017, pp. 203–209.

[34] D. Wang, P. Cui, and W. Zhu, “Structural deep network embedding,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), 2016, pp. 1225–1234.

[35] H. Gao, Z. Wang, and S. Ji, “Large-scale learnable graph convolutional networks,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2018, pp. 1416–1424.

[36] I. Roy, S. Setty, A. Kilzer, V. Shmatikov, and E. Witchel, “Airavat: Security and privacy for MapReduce,” in Proc. 7th USENIX Conf. Netw. Syst. Des. Implement. (NSDI), Berkeley, CA, USA: USENIX Associates, 2010, pp. 297–312.

[37] W. Wei, J. Du, T. Yu, and X. Gu, “SecureMR: A service integrity assurance framework for MapReduce,” in Proc. Annu. Comput. Secur. Appl. Conf., Dec. 2009, pp. 73–82.

[38] H. Ulusoy, M. Kantarcioglu, and E. Pattuk, “TrustMR: Computation integrity assurance system for MapReduce,” in Proc. IEEE Int. Conf. Big Data (Big Data), Oct. 2015, pp. 441–450.

[39] Y. Wang and J. Wei, “VIAF: Verification-based integrity assurance framework for MapReduce,” in Proc. IEEE 4th Int. Conf. Cloud Comput., Jul. 2011, pp. 300–307.

[40] Y. Wang, J. Wei, M. Srivatsa, Y. Duan, and W. Du, “IntegrityMR: Integrity assurance framework for big data analytics and management applications,” in Proc. IEEE Int. Conf. Big Data, Oct. 2013, pp. 33–40.

[41] H. Ulusoy, P. Colombo, E. Ferrari, M. Kantarcioglu, and E. Pattuk, “GuardMR: Fine-grained security policy enforcement for MapReduce systems,” in Proc. 10th ACM Symp. Inf., Comput. Commun. Secur. (ASIA CCS), 2015, pp. 285–296.

[42] H. Ulusoy, M. Kantarcioglu, E. Pattuk, and K. Hamlen, “Vigiles: Fine-grained access control for MapReduce systems,” in Proc. IEEE Int. Congr. Big Data (Big Data), Jun. 2014, pp. 40–47.

[43] N. Solanki, W. Zhu, I.-L. Yen, F. Bastani, and E. Rezvani, “Multi-tenant access and information flow control for SaaS,” in Proc. IEEE Int. Conf. Web Services (ICWS), Jun. 2016, pp. 99–106.

[44] H. Ulusoy, M. Kantarcioglu, E. Pattuk, and L. Kagal, “AccountableMR: Toward accountable MapReduce systems,” in Proc. IEEE Int. Conf. Big Data (Big Data), Oct. 2015, pp. 451–460.

[45] R. Ikeda, H. Park, and J. Widom, “Provenance for generalized map and reduce workflows,” in Proc. 5th Biennial Conf. Innov. Data Syst. Res. (CIDR), Asilomar, CA, USA, 2011, pp. 273–283.

[46] E. Yoon and A. Squicciarini, “Toward detecting compromised MapReduce workers through log analysis,” in Proc. 14th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput., May 2014, pp. 41–50.

[47] C. Liao and A. Squicciarini, “Towards provenance-based anomaly detection in MapReduce,” in Proc. 15th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGrid), May 2015, pp. 647–656.

[48] O. Alabi, J. Beckman, M. Dark, and J. Springer, “Toward a data spillage prevention process in Hadoop using data provenance,” in Proc. Workshop Changing Landscapes HPC Secur. (CLHCS), 2015, pp. 9–13.

[49] H. Ulusoy, M. Kantarcioglu, B. Thuraisingham, and L. Khan, “Honeypot based unauthorized data access detection in MapReduce systems,” in Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI), May 2015, pp. 126–131.

[50] S. Y. Shah, B. Paulovicks, and P. Zerfos, “Data-at-rest security for spark,” in Proc. IEEE Int. Conf. Big Data (Big Data), Dec. 2016, pp. 1464–1473.

[51] M. A. Elsayed and M. Zulkernine, “Towards security monitoring for cloud analytic applications,” Proc. IEEE 4th Int. Conf. Big Data Secur. Cloud. (BigDataSecurity), May 2018, pp. 69–78.

MARWA A. ELSAYED (Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science from the School of Computer and Information Sciences, Ain Shams University, Cairo, Egypt, and the Ph.D. degree in computing from Queen's University, in September 2018. She has more than ten years of interdisciplinary academic teaching and research experience. She is currently a Postdoctoral Fellow with the School of Computing, Queen’s University, Canada, where she is also a member of the Queen’s Reliable Software Technology (QRST) research group. Her research interests span various areas in cybersecurity of cloud computing, autonomous connected vehicles, the Internet of Things, 5G wireless networks, and big data systems. She has received several recognitions and two best paper awards at top international conferences. More information about her publications is available at http://www.cs.queensu.ca/home/marwa/

MOHAMMAD ZULKERNINE (Senior Member, IEEE) received the Ph.D. degree from the University of Waterloo, Canada. He joined Queen’s, in 2003, and spent his sabbatical as a Visiting Professor at the University of Trento, Italy, and a Research Scientist at Irdeto, Canada. He is currently a Professor and the Canada Research Chair with the School of Computing, Queen’s University, Canada, where he leads the Queen’s Reliable Software Technology (QRST) research group. He has led major research projects supported by a number of provincial and federal agencies and industry partners. His current research focuses on building reliable and secure software systems and he has extensive publications in this area. He is a Senior Member of ACM and a licensed Professional Engineer in the province of Ontario, Canada. He has been holding leadership positions, such as the General Chair, the Organizing Chair, and the Program Chair for many major conferences and workshops. More information about Dr. Zulkernine is available at http://research.cs.queensu.ca/home/mzulkern/.