Vol: 2, Issue: 1, 2024
ISSN: 2980-3152
Pages:60-85

CURRENT TRENDS IN COMPUTING
https://dergipark.org.tr/en/pub/ctc
RESEARCH ARTICLE

ANOMALY DETECTION WITH API CALLS BY USING MACHINE LEARNING: SYSTEMATIC LITERATURE REVIEW

VAROL ŞAHİN[1], FERHAT ARAT[2*], SEDAT AKLEYLEK[3]

[1] *Department of Computer Engineering, Ondokuz Mayis University, Samsun, Türkiye*
[2] *Department of Software Engineering, Samsun University, Samsun, Türkiye*
[3] *Department of Computer Engineering, Istinye University, Istanbul, Türkiye*
[3] *Institute of Computer Science, University of Tartu, Tartu, Estonia*

ABSTRACT. **API, in other words system calls are critical data sources for monitoring the operation of systems and applications, and the data obtained from these calls provides a wealth of information for anomaly detection. API calls are the basic building blocks of the interaction between the operating system and user applications, and analysis of these calls provides important data for securing the system. Anomaly detection is crucial for system security and performance. ML models learn normal and abnormal behaviors by processing large amounts of data and use this information to detect anomalies in new data. When anomaly detection using system calls is combined with ML algorithms, it can make more precise and accurate detections. In this paper, we focus on anomaly detection with machine learning methods using API calls. We present a SLR on the topic as well as a SoK by providing basic knowledge. The main goal is to describe, synthesize, and compare security advancements in anomaly detection using API calls with ML algorithms by examining them through the lens of various research questions. More than 30 research papers were retrieved using search phrases identified from common and reputable databases, and those relevant to the topic were included in the SLR using different screening criteria. In addition, the reviewed studies were compared in terms of different metrics such as dataset, platform, success parameter, used ML method, and features.**

1. INTRODUCTION

Smartphones, computers and other electronic devices are involved in every aspect of our daily lives. With the Internet becoming an indispensable element, system security has become an indispensable requirement in both personal and corporate organizations. A system is a set of components that interact with each other and usually form a complex whole. Alternatively, a system can be defined by the functions and behaviors it contains. The interactions between systems, their interconnections, environmental conditions and human governing factors make questions about system safety complex. for instance,

E-mail address: ferhat.arat@samsun.edu.tr (*).
Key words and phrases. API call, system call, anomaly detection, machine learning.

the need for monitoring, measurement and control are critical elements to consider in system interconnections. System security is an important issue not only at the individual and organizational level, but also at the societal level. Keeping up with rapidly growing and evolving technological developments is inevitable in this context.

Today, information security has become even more crucial with the increase in cyber threats. Cyber attacks can cause a wide range of damages at the user and system level, from the theft of personal data to the collapse of corporate systems. Therefore, security measures need to be handled with a proactive approach, not just a reactive one. Advanced monitoring and intrusion detection systems, artificial intelligence and machine learning techniques are used to identify and prevent potential threats in advance. In addition, user training and awareness is also considered an important component of system security. Human error is one of the fundamental causes of many security breaches. Therefore, making users aware of security protocols and teaching them the necessary applications and systems will improve the overall security posture. Anomaly detection is a critical area that aims to detect security threats in advance by identifying unusual behavior of systems. Anomaly detection is also a technique used to detect data samples that do not fit the data model and is an important research area that is being studied for many applications [1], [2]. The main purpose of anomaly detection is to distinguish between normal and abnormal data. This method is very significant in data analysis as it enables the identification of emerging patterns, trends and anomalies in the data [3].

Anomaly detection is critical for various applications such as security, health, network monitoring [4]. In this context, cybersecurity is an emerging and important research domain with applications across various domains such as healthcare, building management, weather forecasting, etc. [4]. Anomalies were considered important because they can point to very important and rare events, enabling critical measures to be taken in a wide range of application areas.

Anomaly detection using system calls, also known as Application Programming Interface (API) calls, provides a detailed analysis of the internal dynamics and behavior of systems. System calls are the basic building blocks that enable the interaction between the operating system and user applications, and analysis of these calls provides important data to ensure the security of the system. An API is an interface that a software program provides to other programs, users, and in the case of web APIs, to the world via the internet [5].

The accuracy and reliability of the methods used in anomaly detection are directly related to the datasets used and the training of the model. The datasets used in the training process are required to adequately represent normal and abnormal behaviors. In this context, the datasets include various types of attacks and system behaviors and provide ideal test environments to evaluate the performance of anomaly detection models. Various features, such as API calls or system calls, registry modification and network activity, constitute the behavior of malware. API calls and various information related to these calls extracted by dynamic analysis are considered as one of the most important features of behavior-based detection systems. Each API call in the sequence is associated with the previous or next API call. These and similar relationships may contain patterns of destructive functions of malware. Many anomaly detection systems, including ML and deep learning models, have been proposed that use various information about API and system calls as features. In particular, ML methods and deep learning algorithms are

used to improve the performance of such anomaly detection systems. for instance, deep learning models can achieve high accuracy rates on complex datasets and can be effectively used in malware detection. In this context, the integration of MLand deep learning methods is of great importance to improve system security and detect malware effectively. There are three anomaly detection techniques: supervised, unsupervised and semi-supervised.

This paper presents a comprehensive Systematic Literature Review (SLR) focusing on anomaly detection with API calls using ML techniques, utilizing the Systematization of Knowledge (SoK) approach. In addition to ML fundamentals and principles, we offer an expansive framework with a specific emphasis on api calls in anomaly detection, contributing to the systematic organization of information. wUnlike existing reviews, our focus is explicitly directed towards using anomaly detection with API calls. We have initiated a comprehensive literature search methodology. The objective is to describe, synthesize, and compare security developments in terms of anomaly detection with API calls.

TABLE 1. Abbreviations and definitions

Abbreviation	**Definition**	**Abbreviation**	**Definition**
API	Application Programming Interface	ML	Machine Learning
KNN	K-Nearest Neighbors	SVM	Support Vector Machine
LSTM	Long Short-Term Memory	CNN	Convolutional Neural Network
ADFA-LD	Australian Defence Force Academy Linux Dataset	DARPA	Defense Advanced Research Projects Agency
UNM	University of New Mexico	IDS	Intrusion Detection System
DDoS	Distributed Denial of Service	TCP	Transmission Control Protocol
URI	Uniform Resource Identifier	HTTP	Hypertext Transfer Protocol
LR	Logistic Regression	NB	Naive Bayes
RBF	Radial Basis Function	WoS	Web of Science
CPS	Cyber Physical System		

1.1. **Motivation and Contribution.**

In this section, we give our motivation as well as summarize the main contributions of the paper. With the rapid advancement of today's technological developments, the use of interconnected systems and applications by individuals and corporate organizations to facilitate operations, increase productivity and provide a seamless experience to users is increasing in parallel.These systems and applications are made possible by APIs that act as a bridge by enabling communication and data exchange between different software components. The increasing use of APIs also increases the need for security measures to protect against potential threats.

The fact that Internet technologies have been adopted and become an integral part of daily life has brought with it disadvantages such as misuse and vulnerability to abuse. The increasing complexity of large amounts of data circulating on the Internet increases the risk of anomalies, unexpected patterns or behaviors.Anomaly identification through API requests is particularly crucial in this context. Especially monitoring an application's API calls to understand its behavior and logic. As they facilitate data transmission and communication between applications, API calls are critical for functionality.

Anomaly detection helps identify unusual behaviors in the system, enabling timely responses before they cause any damage to the application or the system. It detects abnormal requests and intrusion attempts and in this way it helps to protect the system and reduce the costs of deployment and maintenance.

Anomaly detection gives vital information regarding both operations and security. Differences from standard API behavior might indicate problems. Early identification of anomalies allows applications to run more smoothly and efficiently. For example, if an API call is unusually slow, anomaly detection ensures that the issue is identified and resolved before it negatively impact the user experience. Anomaly detection also help protect against financial losses by identifying illegal activities that produces unusual API requests patterns. Detecting such unusual activities early protects businesses from financial harm. As the number of users and systems increase, the volume of API requests also grows. Anomaly detection helps to monitor these large quantities of requests and identify anomalies, reducing the need for manual monitoring and minimize the human intervention. In this regard, our contributions are as follows:

- Our Systematic Literature Review (SLR) offers guidelines enabling researchers to analyze and address specific questions, presenting a Systematization of Knowledge (SoK) approach focused on anomaly detection methods involving API calls from a ML perspective.
- We only consider anomaly detection approaches with API calls using ML techniques.
- We examine ML methods to detect anomalies in the system which use API call sequences as data features.
- We present a detailed comparison for literature based on technique, used data, peformance metric, and other parameters.

1.2. **Research Methodology.**

A Systematic Literature Review (SLR) is a research method that involves systematically collecting, critically assessing, and synthesising existing studies on a clearly defined topic. SLR provides a methodical, repeatable, and reliable framework for reviewing literature. This approach aims to reduce the complexity of research, improve transparency, and offer a thorough understanding of the current state of knowledge in a specific field. In this paper, we aims to extract anomaly detection approaches which utilize API calls in ML concepts to highlight differences, algorithmic design, data features, environmental requirements of the existing studies. Therefore, our primary key is to prepare literature summation as well as presenting compact, collective, and well-constrained SoK. In this Systematic Literature Review (SLR), we employ a set of key terms pertinent to anomaly detection involving API calls. To conduct the research and gather relevant studies, keywords such as ”anomaly detection,” ”API calls,” and ”Machine Learning” are used to create meaningful and consistent search queries. Additionally, an advanced search mode is utilized across five prominent databases: Scopus, Web of Science (WoS), ACM Digital Library, Science Direct, and IEEE Xplore. Then, we apply three-stage research model defined as bellows:

(1) Definition: In this step, we create different combinations of essential keywords to ensure a reliable and consistent search in databases. We also develop research questions that consider our research focus and key factors, preparing for the SLR.
(2) Determination: In this stage, we determine searching filters and used sentences for advanced search.

(3) Elimination: In the last step, we eliminate obtained studies regarding elimination metrics and main purpose of the SLR. Therefore, we create inclusion and exclusion methodologies as shown in Table 2. And we apply the determined manual filters such as ”research article” and ”computer science research area”.

1.3. **Research questions and planning the review.**

To outline the review's future direction, defining the primary objectives is essential. We set these goals to provide a comprehensive and practical viewpoint. Our SLR results aim to deliver conclusions that are applicable, realistic, and easy for researchers to understand. We review the papers using various criteria, including the datasets used, ML methods employed, performance metrics, and data features. To achieve our objectives, we develop research questions that break the research into sub-phases. We outline the generated research questions as follows:

(1) What datasets and data features are most commonly used in ML methods for anomaly detection utilizing API calls?
(2) How is the performance of ML methods for anomaly detection using API calls evaluated, and which metrics are most commonly utilized?
(3) Which ML methods for anomaly detection using API calls are the most effective, and what characteristics make these methods stand out?

We expand our research by considering the research questions defined above. For each research question, we identify the points of description, comparison, and summarizing that we consider important when reviewing the research papers within the scope of our study. In this context, the first question aims to identify the datasets and data features that are frequently employed in ML methods for anomaly detection using API calls. Understanding which data sources and features are preferred and yield successful results is the goal. The second one focuses on the performance evaluation of studies. Analyzing the performance metrics used helps compare different methods and identify the most commonly preferred metrics. The final question aims to determine the effectiveness of different ML methods. Comparing various methods and analyzing which ones yield better results will be one of the key findings of our study. Figure 2 shows research methodology applied throughout paper.

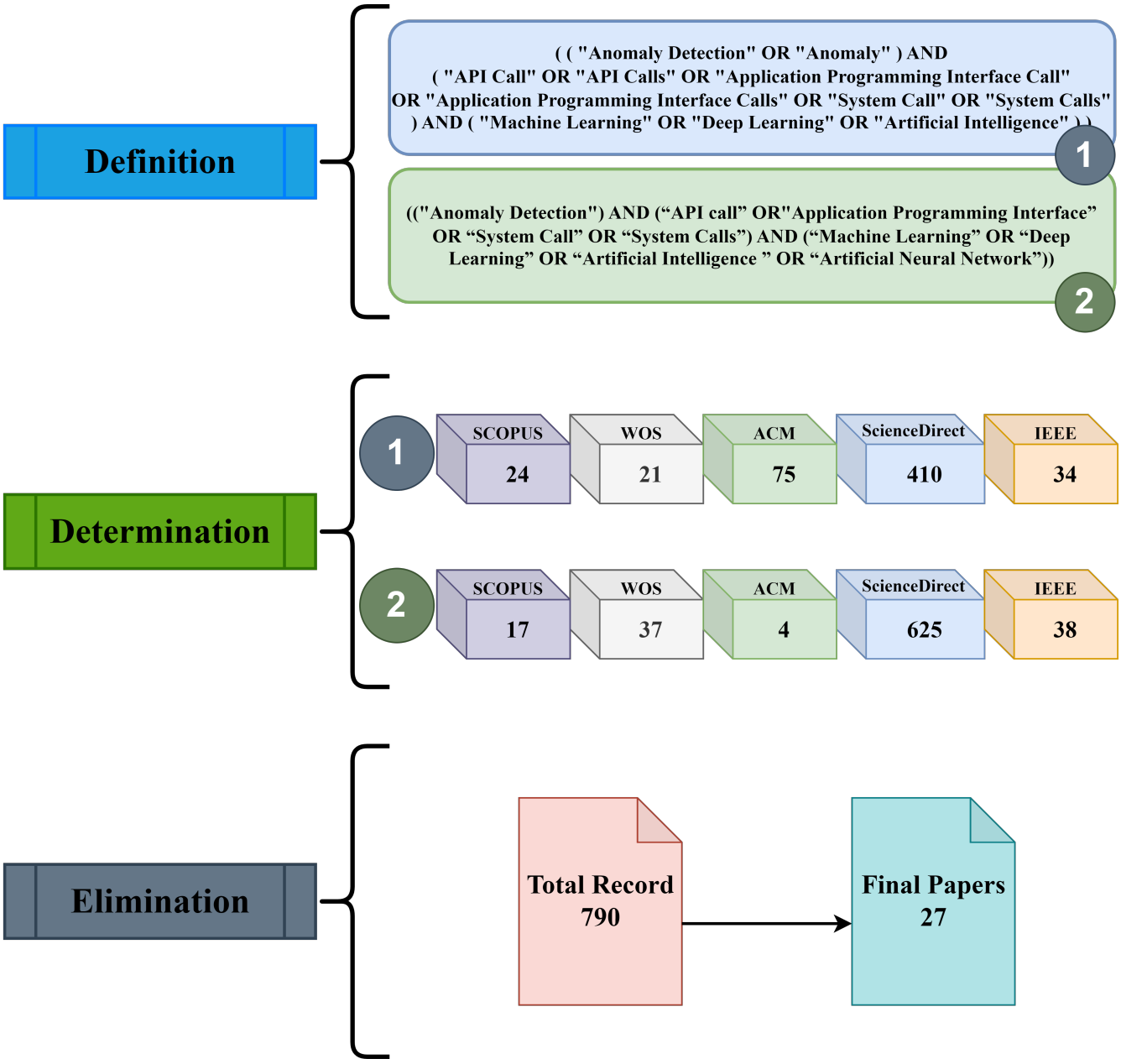

FIGURE 1. Applied research methodology of the paper

1.4. **Determining and performing of the investigation.**

In this section, we address the planning process for selecting studies to be reviewed, following the stages of formulating research questions defined to better specify the focus areas by expanding the SLR

and creating search phrases using keywords. The selection of studies is carried out manually and by using the filtering methods of search engines. In this context, the initial results are subjected to several logical filters to form a study set appropriate for the research. Table 2 shows inclusion and exclusion metrics applied during determining and eliminating the articles.

TABLE 2. Applied research criterion

Exclusion criteria	**Inclusion criteria**
The study must focus on anomaly detection using API calls.	Duplicated papers.
The study must be published in English.	Published in any language other than English.
The study must be related with research fields of computer science or computer engineering.	Review article, conference paper, or another types except research article.
The study should use ML techniques considering API calls for anomaly detection.	Studies using different datasets other than API calls for anomaly detection.

Table 2 outlines the exclusion and inclusion criteria employed in the selection of studies pertinent to anomaly detection using API calls. The exclusion criteria specify that eligible studies must be published in English, ensuring they are accessible to an international audience. Additionally, the studies must be relevant to the fields of computer science or computer engineering, thereby maintaining technical relevance. Methodologically, the studies must utilize ML techniques specifically focusing on API calls for anomaly detection, ensuring consistency with the research objective. Conversely, the inclusion criteria exclude duplicated papers, those published in languages other than English, and studies that are not original research articles, such as review articles or conference papers. Furthermore, studies using datasets other than API calls for anomaly detection are also excluded. These stringent criteria ensure that the selected studies are both methodologically relevant and focused on the specified research area, thereby enhancing the reliability and validity of the research findings.

1.5. **Organization.**

The paper is organized as follows: In Section 1, we explain general definitions of the paper, describing research questions, research structure, and investigation purposes. In Section 2, we give the basics of the topic defining API calls and importance of the anomaly detection using API calls. In Section 3, we present ML concepts giving fundamentals, definitions, and classifications. In Section 4, we highlight and summarize obtained studies in terms of varying research parameters. In addition we compare literature considering specified metrics. In Section 5, we conclude the paper defining open problems and challenges for focused research problems.

2. ANOMALY DETECTION

In this section, we address the topic of anomaly detection, providing general information as well as discussing its significance, the role of API calls in anomaly detection, detection techniques, and the use

of ML methods in anomaly detection. By systematically presenting this framework, we establish the foundational background necessary for the subsequent literature review.

A vital technique in the field of data analytics, anomaly detection plays an important role in various applications such as security, health and network monitoring [4]. Anomalies can indicate critical and often rare events that needs immediate attention. In the rapidly evolving field of cybersecurity, where it is vital for maintaining system integrity and preventing hostile activities, anomaly detection is of a great importance. Anomaly detection is a fundamental component of IDS in cyber security. These systems are able to monitor network traffic and system activity to identify unusual actions that may indicate a security breach. For example, an unexpected increase in data traffic might indicate a DDoS attack, where an attacker overwhelms a network with excessive traffic to disrupt services. Similarly, unusual login attempts from unfamiliar locations or devices could signal potential unauthorized access, requiring further investigation to prevent data theft or system vulnerabilities.

Anomaly detection is a highly important technology with diverse applications across various industries. Its ability to identify unusual events allows for the implementation of preventive measures, thereby it enhences safety, reliability, and efficiency in different sectors. As researches in this field enhences, the integration of advanced machine learning methods and accessible artificial intelligence techniques will be essential for improving the accuracy, reliability, and interpretability of anomaly detection systems. This will ensure their continued relevance and effectiveness in a world where data is essential

2.1. **API calls in anomaly detection.**

The API is a critical element of the operating system and encompasses a set of functions contained in specific libraries. Users utilize these functions to communicate with the operating system reflecting the behavior of various files [6]. In the realm of APIs, the term ”application” generally refers to any software that performs a function. An interface, in this context, acts as a service contract that enables two applications to communicate through requests and responses. Essentially, APIs serve as mechanisms that facilitate communication between two software components using specific protocols. API calls are programming interfaces utilized by applications to interact with each other [7]. During an API call, one server sends a request to another server's web interface via the Transmission Control Protocol (TCP). To make a request, three primary components are necessary: the Uniform Resource Identifier (URI), headers, and the request body. While each API may use a distinct combination of these components, communicate in a different format, or require varying data, the request generally follows the Hypertext Transfer Protocol (HTTP) message structure. APIs provide analysts with a critical foundation for examining a program's behavior and functionality, particularly when direct reverse analysis is challenging. This foundation aids in detecting anomalies within the system's behavior. A system call, on the other hand, is a request made by a program to the kernel for a specific service. Analyzing the trace of such calls can reveal the behavior of the process. These traces are instrumental in classifying the process as either normal or malicious [8].

3. MACHINE LEARNING: FUNDAMENTALS, DEFINITIONS AND CATEGORIZATION

Anomaly detection using ML techniques with API calls is of significant importance in various domains, particularly in cybersecurity, network monitoring, and application performance management. API calls are a valuable source of data that offers detailed information into system behavior and interactions. Unusual patterns and deviations in API call data can be identified by using ML techniques, which may indicate potential security breaches, system problems, or performance issues. This approach simplifies proactive measures, enhancing system reliability, security, and efficiency. Various ML techniques are used for anomaly detection with API calls. Some of these techniques are; supervised, unsupervised, and semi-supervised learning methods. Supervised learning models are trained on labeled datasets that allows them to recognize known anomalies. Unlike supervised methods, unsupervised methods uncover hidden patterns in the data to determine anomalies without labeling. Semi-supervised approaches use both labeled and unlabeled data and improve detection performance by combining both methods. Using these ML techniques enables the analysis of API call data, providing strong and reliable anomaly detection. This approach ensures that systems can reduce issues effectively, maintaining optimal performance and security. Figure 2 illustrates classification of ML methods under varying learning techniques.

3.1. **Logistic Legression.**

Logistic regression (LR) employs the sigmoid function to calculate probability values and perform classification tasks. The sigmoid function produces outputs ranging from 0 to 1. Samples with probability values below 0.5 are classified as belonging to the negative class, while those with values of 0.5 or higher are assigned to the positive class. 3.1 [9].

3.2. **Support Vector Machines.**

SVM are supervised learning algorithms frequently utilized for both binary and multiclass classification tasks. SVM operates by mapping input data points into a high-dimensional space and constructing a hyperplane that is one dimension less to distinguish between different groups of data points [9]. The main objective of SVM is to identify a hyperplane to optimally separate the data into two distinct clusters by maximizing the distance between them. When a linear separation is not possible, a technique known as kernel cheating is used (Muhammad and Yan, 2015). Widely utilized kernel functions include Gaussian, radial basis function (RBF) and polynomial kernels. The most significant advantage of SVM is the ability to avoid overfitting and its non-probabilistic nature [10].

3.3. **Naive Bayes.**

Naive Bayes (NB) classifiers are straightforward probabilistic models [10]. The term ”naive” stems from the assumption that all input features are independent and uncorrelated with each other. This algorithm is fundamentally based on Bayes' theorem and computes the probability of each class for a given set of input features [11].

3.4. **Random Forest.**

Random Forest (RF) is an ensemble learning method composed of multiple decision trees. Each tree in the model employs a decision tree algorithm to choose a subset of features. After the forest is formed using the RF technique, new data is classified by passing it through each tree. The trees vote for the class they believe the instance belongs to, and the forest selects the class with the highest number of votes. RF

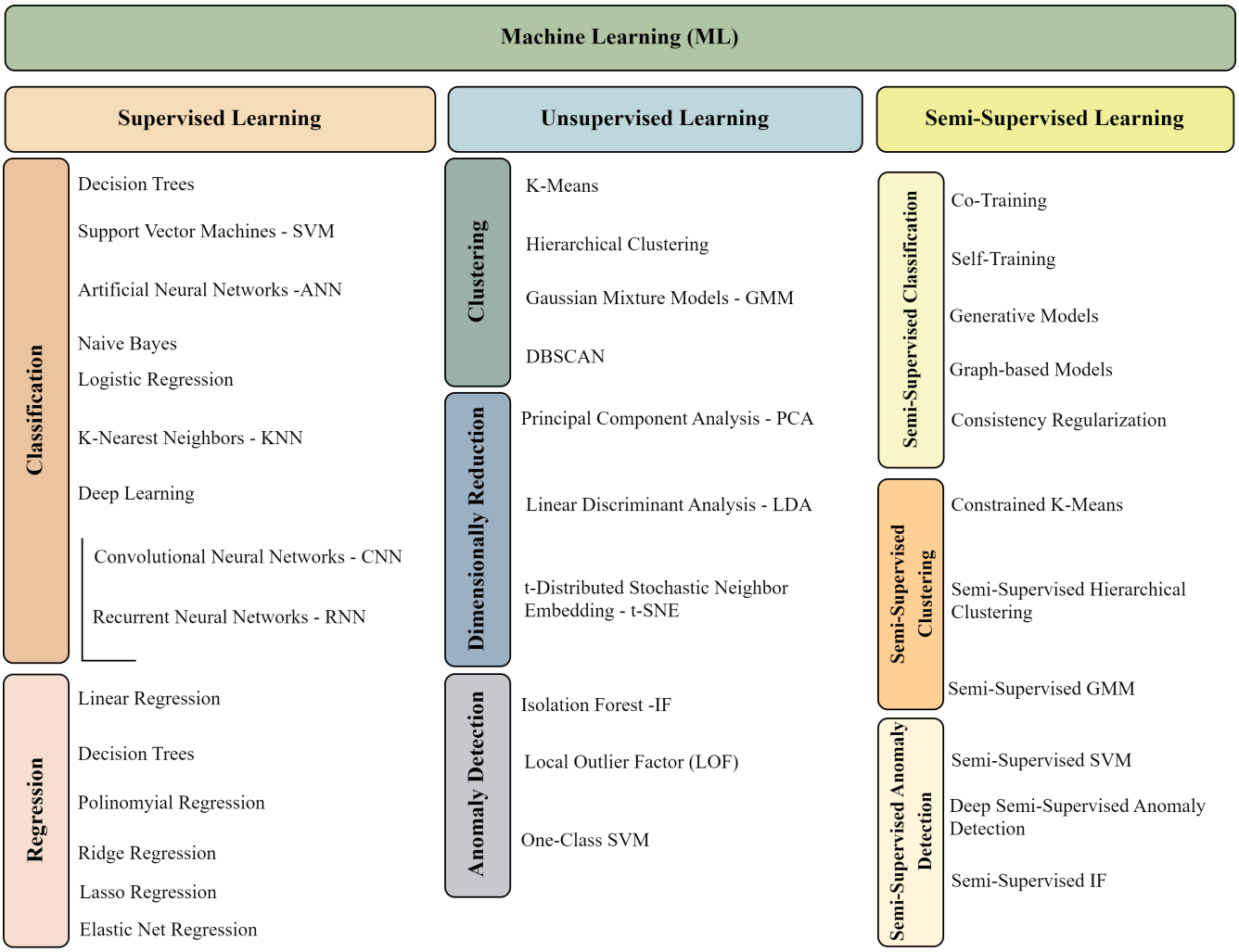

FIGURE 2. Classification of the ML methods

is particularly valued for its noise robustness and lower susceptibility to overfitting compared to other algorithms [12].

3.5. **K-Nearest Neighbor.**

The K-Nearest Neighbor (KNN) algorithm is one of the most basic supervised learning techniques used to classify data into distinct categories [10]. Being non-parametric and probabilistic, KNN is suitable for classification tasks where there is no prior knowledge about the data distribution. The algorithm classifies a new sample based on the majority vote of its k nearest neighbors, determined by a similarity measure (distance). However, its computational complexity increases with data size, often necessitating dimensionality reduction techniques to mitigate the curse of dimensionality before applying KNN [13].

3.6. **Deep Neural Network.**

DNN is one of the most commonly used methods in ML. Unlike traditional ML techniques, it excels when processing large datasets. A notable feature of DNN models is their deep architecture consisting of multiple hidden layers [14]. The design of a DNN reflects the working logic of the human brain and typically consists of an input layer, several hidden layers and output layers. As a result, DNN models encompass many units, making them suitable for classifying non-linear and complex data. However, training DNN models requires more time than other methods due to their complex model structure and size [14].

3.7. **Long Short-Term Memory).**

The LSTM model enhances the capabilities of the RNN (Recurrent Neural Network) model. RNNs, as deep learning models, encounter the vanishing gradient issue when network layers increase, which hampers the network's ability to learn from previous information. LSTMs address this problem with a complex recurrent unit that employs a gating mechanism to regulate information flow. These models incorporate memory cells with fixed weights and self-contained recurrent nodes, enabling the retention of state values over long periods and allowing gradients to pass through numerous time steps without diminishing [15].

Traditional RNNs process sequential data inputs with forward recursive computation, chaining neurons together to integrate past information with current inputs, effectively functioning as a memory to handle time sequences. However, due to their limited memory capacity, RNNs retain less historical information while preserving more recent data. During training, information distortion occurs as it passes through numerous iterative loops, leading to gradient fading.

In contrast, the LSTM model enhances traditional neurons with memory cells, significantly improving the network's information transfer and processing capacity. These memory cells effectively store historical data, and the input gates within the cells autonomously manage the retention time of values, enabling better prediction of crucial information [16]

3.8. **Convolutional Neural Network.**

CNNs have become a cornerstone of deep learning, especially in visual tasks, due to their outstanding performance in computer vision applications such as object recognition, detection, and segmentation. These networks have not only achieved top results in various tasks but also matched human-level performance in recognizing visual objects and in critical medical applications [17].

CNNs are a favored deep learning technique because of their ability to model complex, non-linear relationships. They are more efficient than traditional DNNs and excel at learning abstract image features, which makes them particularly suitable for image processing. A CNN with sparse connections and shared weights has significantly fewer parameters compared to a fully connected network of similar size. The architecture of CNNs consists of three main layers: convolutional layers, pooling layers, and fully connected layers. Unlike standard ANNs where each hidden layer has distinct weights, inputs, and outputs, CNN neurons operate on two-dimensional planes for inputs and outputs, using feature maps (kernels) as weights. Convolutional layers extract features from the input images, organizing the outputs into two-dimensional planes known as feature maps. Each layer's plane is formed by combining outputs from the previous layer. As features are passed to higher layers, their size decreases in proportion to

the filter size used in the convolution and pooling layers, while the number of feature maps increases to improve feature extraction and classification accuracy. Pooling layers typically follow convolution layers. After convolution and pooling, the extracted features are converted into a vector for classification via fully connected layers, which are recognized for their high performance [18].

CNNs, inspired by the visual processing mechanisms of the human brain, are a type of multi-layer perceptron within the feed-forward neural networks category. They are designed to automatically and adaptively learn spatial feature hierarchies through backpropagation using key components such as convolution layers, pooling layers, and fully connected layers [19]. The structure of a CNN includes an input layer, an output layer, and several hidden layers, which can be convolutional, pooling, or fully connected [20].

4. ANOMALY DETECTION USING API CALLS WITH ML MODELS

In this section, we systematically summarize and analyze the studies obtained in response to the research questions posed earlier. This systematic review forms the core of our investigation, building on the general concepts, fundamentals, and definitions provided in the preceding sections. By examining the datasets and data features most commonly used in ML methods for anomaly detection utilizing API calls, we aim to identify the prevalent data sources and attributes that drive effective anomaly detection. Furthermore, we evaluate how the performance of these ML methods is assessed, focusing on the metrics that are most commonly employed in the literature. Lastly, we identify the most effective ML methods for anomaly detection using API calls and explore the characteristics that make these methods particularly successful. Through this structured approach, we not only synthesize existing research but also provide a comprehensive comparison and analysis, offering valuable insights into the state-of-the-art techniques in this domain. Table 4 highlights general contributions of the investigated papers.

TABLE 4. System Call Anomaly Detection Techniques Overview

Ref.	Contribution	Method
[21]	A deep learning approach for detecting anomalies in power grid systems Detection of anomalies in power measurement sets Generation of power system data using a customized module Addressing imbalanced dataset issues in anomaly detection.	Customized real-time cluster formation Anomaly detection and identification using DNN
[22]	Automated system call collector for anomaly detection Large-scale dataset of system calls in Linux kernels Deep learning-based anomaly detection using various models like CNN, LSTM, etc.	Real-time collection of system call logs from Linux systems Anomaly detection using CNN/RNN, LSTM, WaveNet, and ECOD
[23]	Open-source anomaly detection and alert framework for applications Anomaly detection considering different parameters in system calls IDS framework applied to container platforms using machine learning methods	Data reading and writing using Elasticsearch API Anomaly detection using Kubernetes, CNN, and cuDNN
[24]	Frequency and LSTM-focused anomaly detection framework for cloud systems Anomaly detection using LSTM architecture with system calls Attack-based classification based on the frequency of past call sequences	LSTM-based architecture for attack classification Feature selection using TF-IDF and attack classification using NN
[25]	Anomaly detection by analyzing behavioral units Threat analysis at the system call and API levels focusing on critical behaviors	Anomaly detection using a multi-level DL model Attack prevention using a multi-level transformer-based model
[8]	Deep learning-based anomaly detection Improved training time efficiency with an enhanced neural architecture	Anomaly detection using LSTM and CNN with hyperparameter optimization
[26]	Analysis of program behaviors based on characteristic values (CV) Modeling program behavior by altering function parameters with CV sequences Creating a neural network vector using word embedding method and prototype design	Learning method based on CV values with LSTM-RNN
[27]	Real-time dataset analysis using APIs Anomaly detection using machine learning considering traffic flow through APIs	Anomaly detection and classification using STL and iForest
[28]	Design of anomaly detection algorithm for cloud systems Custom classification method for detecting faulty states Scenario design for various subsystems based on fault scenarios	Markov model for state probability estimation Modeling using PPM-C and VMM
[29]	Graph model for modeling state data GCN modeling with network traffic and system state data	Intrusion and anomaly detection using hybrid tGCN-KNN method
[30]	Anomaly detection using system calls for Linux environment End-to-end anomaly classification using machine learning with system logs	Feature vector transformation using Doc2Vec, RNN-AE, and RNN-DAE Anomaly detection using IF, LOF, and 1-SVM
[31]	Design of an ML-based detection system for anomalous pods in Kubernetes clusters Modeling the system using Linux kernel calls	Classification explanation using SHAM and LIME Machine learning implementation using decision trees, ANN, and EL
[32]	Feature-based anomaly detection and classification method	Anomaly detection using hierarchical clustering algorithms
[33]	Multi-anomaly detection system based on application behaviors Weighted graph representation based on system call numbers and frequencies IDS modeling using DNN with system calls	Three-component feature vector method for neural networks with D-MLP Anomaly detection using STIDE, text classification, and system call graph
[34]	Anomaly detection by reducing troubleshooting time for developers Collection of system calls during execution in a Linux environment	Collection of system calls using Toolkit Next Generation Classification using multi-class SVM Clustering using K-means and Dbscan
[35]	Anomaly detection for IoT networks with a customized NIDS method named Panop Feature extraction based on network-related device behaviors Real-time scenario representation using Raspberry Pi devices	Anomaly detection using Kitsune and ANN
[36]	NLP implementation for program behavior analysis using BoSC Anomaly detection based on the sequence of system calls at a specific point in time	Anomaly detection using Cosine Similarity Algorithm (Co-Sim) based on NLP
[37]	ML-based methods for anomaly detection accuracy using normal and attack data	N-gram technique for creating feature vector Classification and prediction using SVM, LR, and KNN methods
[38]	Feature extraction method using system call names Low-cost feature extraction method applicable across different platforms	N-gram technique to convert system calls to frequency sequences Classification using IF, LOC, OCSVM, and KNN
[39]	Anomaly detection using LSTM framework Hybrid detection model using LSTM and unsupervised learning framework	Classification using OC-SVM, LSTM, and SVDD
[40]	Flexible anomaly detection system using system security logs Semantic feature extraction and threat behavior modeling for internal threats	Model using LSTM and GRU

4.1. **Prevalent Datasets and Key Features in API Call Anomaly Detection.**

In this section, we will identify and describe the most commonly used datasets and the key features that are leveraged in ML methods for anomaly detection utilizing API calls. We will provide an overview of the sources of these datasets, the nature of the data they contain, and the specific features that are extracted and used for training anomaly detection models. This analysis will help in understanding the data foundation upon which current research is built and highlight any gaps or opportunities for future dataset development.

In [8] a hybrid anomaly detection system based on deep learning techniques, aiming to enhance both accuracy and efficiency was proposed. The proposed system combines CNN and LSTM to achieve improved detection capabilities. The initial step involves inputting the raw sequence of system call traces into the CNN network to decrease the dimensionality of the traces. Subsequently, the reduced trace vector was passed to the LSTM network to understand the sequence of system calls and generate the final detection result.The hybrid model was implemented and trained using TensorFlow-GPU, and its performance was evaluated on the ADFA-LD dataset. The ADFA-LD host-based intrusion detection dataset was generated by the ADFA. This dataset records Linux system calls, which facilitate communication between user and kernel modes through standard interfaces provided by the Linux kernel. Every system call on the sequence trace has a unique identifier number. The host was designed to profile a recent Linux server that logs system call traces during a specific time period. ADFA-LD dataset was used in the first phase. In the second phase, stratified sampling was applied and the dataset was divided into training, validation, and test. In the second phase, a hybrid deep learning model utilizing CNN and the LSTM algorithm was trained. The CNN consists of two layers: the convolution layer and the pooling layer. The convolution layer applies convolution procedures to the input picture to extract significant features, while the pooling layer decreases image dimensionality and deals with data nonlinearity. ReLu activation function was employed. Finally, a hybrid DL-based CNN with LSTM was presented to detect abnormalities in sequential system calls. The CNN extracted significant features, and the LSTM learned the sequence patterns from the reduced data. Experiments reveal that the suggested technique displayed reduced training time and better anomaly detection rates, hence lowering false alarm rates.

In [41], a novel anomaly recognition and detection framework named AnRAD inspired by biological systems, which utilizes probabilistic inferences was proposed. It investigates feature dependencies and introduces a self-structuring approach that learns an efficient confabulation network from unlabeled data. This network enables fast incremental learning, continuously refining its knowledge base with streaming data. Comparative analysis with existing anomaly detection methods demonstrates competitive detection quality. Moreover, the AnRAD framework leverages parallel processing capabilities, yielding significant speed enhancements when implemented on graphic processing units and Xeon Phi coprocessors compared to sequential execution on standard microprocessors. Versatility of the framework enables real-time processing of concurrent data streams across various knowledge domains, making it applicable to large-scale problems characterized by multiple local patterns. The proposed methodology incorporates the principles of the confabulation theory within a hierarchical cognitive architecture, enabling flexible network configurations tailored to specific applications. Leveraging the computational power of GPUs

and Xeon Phi processors, the notable speed enhancements through both fine-grained and coarse-grained parallelization methods were achieved.

In [24], a novel intrusion detection framework was introduced, which can identify both known and unknown attacks through system call sequence analysis. This framework examines the system call sequences of VMs using a hybrid model that combines LSTM networks with anomaly detection techniques based on system call frequency. The effectiveness of this framework was validated using the ADFA-LD. System call traces are collected and stored as a dataset for offline training, necessitating preprocessing to remove extraneous information and retain only system call sequences. These sequences are then encoded with unique identifiers and labeled as normal or malicious. Frequency-based methods like Bags of n-grams are employed to generate a Feature Vector Matrix (FVM) from the processed traces. These vectors, representing the frequency of distinct n-grams, are stored in a feature-vector database file. Experimental results showed that this framework outperforms existing models in accuracy and has a lower false positive rate.

In [38], a new feature extraction method designed to derive features that are independent of system call names, making the samples directly applicable to cross-platform scenarios. The method converts system call sequences into frequency sequences of n-grams and then extracts a fixed number of statistical features from these sequences. These features are calculated based on the frequency sequences rather than the direct system call sequences, and are used to train a one-class classification model for anomaly detection. The study utilized the ADFA-LD, ADFA-WD, and NGIDS-DS datasets, employing anomaly detection algorithms like Isolation Forest, LOF, OCSVM, and kNN. The method was compared with other feature extraction techniques, such as Bag of System Calls, tf-idf, and subsequence frequency. Even though the proposed method did not always achieve the highest AUC on the same platform, it generally outperformed other methods, especially in cross-platform scenarios.

Lv et al. developed a system-call sequence-to-sequence prediction model by semantically modeling system calls [42]. This model predicts future system calls to monitor system states and detect attack behaviors. An end-to-end system call prediction model was built to predict subsequent system calls based on traces generated during malicious process execution. The RNN model was used to ensure the generation of semantically reasonable sequences. The model was evaluated using the ADFA-LD dataset, which contains traces from both normal and intrusion attempts. Performance was assessed using the BLEU score and Euclidean distance between encoded semantic vectors, with TF-IDF used for sequence similarity evaluation. The predicted sequences, when combined with known system call traces, significantly enhanced intrusion detection performance across various classifiers.

System calls are the primary means for applications to communicate with the Operating System (OS), making them vital for Host-based Intrusion Detection Systems (HIDS). In [22], several existing datasets are outdated, prompting the introduction of a large-scale dataset specifically for anomaly detection in the Linux kernel. The dataset, named DongTing, comprises 85 GB of data, including 18,966 system call sequences labeled as normal or anomalous. It encompasses over 200 kernel versions and 3600 bug-triggering programs from the past five years. Cross-dataset evaluation demonstrated that models trained on this dataset exhibited superior generalization capabilities. The dataset was divided into training, validation, and testing subsets for training deep learning models to detect anomalies in Linux kernels. Four

deep learning models—CNN/RNN, LSTM, WaveNet, and ECOD—were evaluated, showing that models trained on this dataset achieved the highest generalization scores and better performance when trained on abnormal data. This framework significantly reduces the time required to produce the dataset.

In [30], an end-to-end strategy was presented for identifying abnormal activities, merging sequential information preservation through log embedding techniques with anomaly detection algorithms based on ML. Unlike current ML methods for system anomaly detection, which rely on domain experts to extract relevant features from log data, the proposed method converts raw log data into fixed-size continuous vectors regardless of length. These vectors are then utilized to train anomaly detection algorithms. This paper proposes a strong intrusion detection model designed specifically for Linux settings, combining sequential information-preserving log embedding techniques with anomaly detection algorithms. Doc2vec was used to convert system call traces of different durations into fixed-dimensional real-valued vectors, as are recurrent neural network-based auto-encoder (RNN-AE) and recurrent neural network-based denoising auto-encoder (RNN-DAE) approaches that keep sequential information. To validate the detection model, an experiment was carried out using the ADFA-LD dataset, which contains Linux system call traces. Three assessment measures were utilized to assess the performance of anomaly detection algorithms. The ROC was used to assess the performance of each model. In the studies, the performance of anomaly detection systems based on unsupervised learning was assessed across several attack types. After gathering a significant quantity of labeled data, a supervised classification model was trained and its performance was compared against anomaly detection techniques. Finally, the paper provides an unsupervised ML-based system anomaly detection framework that does not require labeled data for model training.

In [30] an end-to-end approach was proposed for detecting abnormal behaviors, integrating sequential information preservation through log embedding algorithms with anomaly detection algorithms based on ML. Unlike other ML models for system anomaly detection that rely on domain experts to extract meaningful features from log data, the proposed approach transforms raw log data into fixed-size continuous vectors regardless of their original length. In this work, a robust intrusion detection model was developed which was specially created for Linux environments, leveraging sequential information-preserving log embedding techniques alongside anomaly detection algorithms. To convert system call traces of varying lengths into fixed-dimensional real-valued vectors, Doc2vec was used, as well as recurrent neural network-based auto-encoder (RNN-AE) and recurrent neural network-based denoising auto-encoder (RNN-DAE) methods, which retain the sequential information. To validate the detection model, an experiment was conducted using the ADFA-LD dataset, which comprises Linux system call traces. In the experiments, firstly the performance of anomaly detection algorithms based on unsupervised learning was evaluated across different attack types. After gathering a significant quantity of labeled data, a supervised classification model was trained and its performance was compared against anomaly detection techniques.

4.2. **Performance Evaluation Metrics for Machine Learning-Based API Anomaly Detection.**

This section will focus on how the performance of ML methods for anomaly detection using API calls is evaluated. We will examine the most commonly utilized metrics, such as accuracy, precision, recall, F1 score, and area under the ROC curve (AUC). By analyzing these metrics, we aim to provide insights into

how researchers measure the effectiveness of their models, the challenges associated with each metric, and the contexts in which certain metrics are preferred over others. This will offer a comprehensive understanding of the evaluation landscape in this field. Figure 3 illustrates dataset and platform summary of the examined studies.

In [34], system calls based anomaly highlighting and detecting framework was proposed to guide developers regarding performance problems in data. The LTTng was used to collect data from the Linux kernel, applications, and libraries. A supervised learning method was utilized in order to manage large amounts of labeled data. In addition, the learning method was improved and modified as semi-supervised. Then, the feature vector was constructed considering the duration of the most significant call sequences. In the detection phase, an automated anomaly detection method was implemented as module-by-module. It was indicated that the proposed method ensures high accuracy and efficiency in large scale systems and distinguishes CPU and memory shortages as well as detecting normal behavior.

In [39], an unsupervised anomaly detection and innovative algorithms based on LSTM neural networks were developed. The proposed structure was started by processing variable length data sequences through an LSTM-based structure, resulting in fixed-length sequences. Decision functions for anomaly detection were then derived using One-Class Support Vector Machines (OC-SVMs) or Support Vector Data Description (SVDD) algorithms.The main contribution of the proposed work was indicated that the simultaneous training and optimization of LSTM architectural parameters as well as OC-SVM or SVDD parameters, facilitated by highly effective gradient and quadratic programming-based training methods.This study extends the unsupervised framework to semi-supervised and fully supervised settings. The resulting anomaly detection algorithms excel in processing variable length data sequences, particularly time series data, offering superior performance compared to conventional methods.

In [43], an anomaly detection model utilizing LSTM networks as well as intra- and inter-trace context vectors was proposed to overcome the challenge of online anomaly detection in CPSs. This dynamic approach allows both identified and unseen anomalies to be addressed while improving the understanding of kernel event execution contexts both horizontally and vertically. A deep context-aware architecture was introduced for anomaly detection in semi-structured sequences specifically focused on system calls or kernel events. Furthermore, the importance of using a context-based attention layer to extract rich semantics that help identify non-linear high-dimensional relationships present in syslog sequences was emphasized. In the simulation, a custom dataset generated by existing work was used. Two parameters were relevant to analyze the complexity of the model. Finally, the proposed approach characterizes the behavior of the system through online execution trace analysis using recurrent neural networks. Experiments show that the proposed model provides effective and robust results in anomaly detection using system call sequences.

In [44], a state summarization and and nested-arc hidden semi-Markov model (NAHSMM) model was proposed to model dynamic usage behavior and identifying anomalies for cloud servers. The model was designed to control the propagation of system call sequences and less usage transitions. In addition, the proposed detection algorithm was generated by integration of NAHSMM and state summarization. The system calls in varying length were summarized using these methods and the NAHMM was utilized to fit time sequences. As fundamental, the proposed system was constructed as a mathematical model. The

FIGURE 3. Comparison of the studies in terms of dataset and platform

Reference	Dataset	Number of Instances	Platform
[34]	Synthetic	946,000 - 3,800,000	Not provided
[25]	Firefox-SD, ADFA-LD, PLAID, Synthetic	18,966	Intel Xeon E5-2640 V4, DDR4 ECC 2400 MHz, 128G memory, 4 GPU
[35]	Synthetic	1,551,006	Linux OS, GeForce RTX 3060
[22]	ADFA-LD	Not provided	Ubuntu 18.04 OS, i5 processor, 8 GB RAM, 500 GB HDD, Xen 4.6
[36]	AndroCT	35,974+	Windows 10 OS, AMD Ryzen 7 5800 8-Core Processor, 3.40 GHz, NVIDIA 3060, 32.0 GB RAM
[8]	ADFA-LD	Not provided	Not provided
[37]	Synthetic	Not provided	Not provided
[38]	Synthetic	5,748 - 15,000,000+	Not provided
[33]	Not provided	Not provided	Intel Xeon server (E5-2630L v3 @ 1.80 GHz), 16 GB RAM, 150 GB, Linux CentOS v7.0
[32]	CSE-CIC-IDS2018, UNSW-NB15, Synthetic	257,673	Intel Xeon (Cascade Lake) Platinum 8269 2.5GHz/3.2GHz 4-core CPU, 8GB RAM
[39]	Synthetic	Not provided	Not provided
[40]	ADFA-LD	5,206	Intel (R) Core (TM) i7-7700K CPU 4.20GHz, NVIDIA GTX 1080Ti 11GB RAM, 32G RAM
[41]	Synthetic	Not provided	Intel Core i5-6200U CPU 2.3 GHz and 2.4 GHz, 8GB RAM
[42]	DARPA, UNM, ADFA-LD	2,766 for UNM, others session-based variable	Intel Core i7-4790 CPUs, 16 GB RAM
[27]	Synthetic	Not provided	Intel Core i7 4 GHz, 8 CPU, 32 GB
[26]	Synthetic	50,000	Not provided
[43]	ADFA-LD	Not provided	Not provided
[23]	ADFA-LD, ADFA-WD, NGIDS-DS	5,951 for ADFA-LD, 7,725 for ADFA-WD, 3,070 for NGIDS-DS	Not provided
[28]	Occupancy, HHE, Http, Alcoa	Not provided	i5-6400, 2.7 GHz CPU, 16 GB RAM
[31]	CMU CERT v6.2	200,000	Not provided

main concept was to use structured numerical models that include summarization of states to better understand the behavior of sequences of system call identifiers. In training and testing phases, IXIA Perfect Storm was used to collect data. As final, the effectiveness of the proposed model in accurately detecting anomalies within machine operating systems was highlighted . By leveraging descriptive features and a streamlined structural framework, the model achieves this with fewer parameters, leading to significant reductions in computational complexity and storage needs. Although the focus is on modeling system call identifier sequences from servers in cloud environments, the method shows promise for application in classifying network traffic and identifying anomalous human behavior.

In [40], a threat detection model was proposed implementing the Word2vec-based approach. The possibility of suspicious behavior was assessed by leveraging Word2vec model trained on a corpus of various security logs.The method consists of three main components: log2text, text2corpus, and anomaly detection. The log2text component standardizes events from diverse security logs into uniform text format. These texts are then merged, sorted chronologically, and processed into a corpus by the text2corpus component. Finally, the anomaly detection component trains a Word2vec model on this corpus to compute the probability of a specific behavior given an event, denoted as p(behavior—user). Events exceeding a certain threshold are flagged as suspicious, potentially indicating malicious insider activity if multiple suspicious events are associated with a user. The dataset was divided into smaller data and selected specified security logs. The TPR and FPR were used as success metrics to determine performance of the anomaly detection. The proposed study was compared with existing ones in terms of cost and complexity.

In [29], a novel anomaly and intrusion detection models were designed. The network data was represented as a graph structure to identify relation features between samples. The graph structure was constructed as a triplet graph CNN and it was used to detect anomalies in the system. In addition Graph Convolutional Network (GCN) was modeled and CSE-CIC-IDS2018 and UNSW-NB15 datasets were used to monitor performance of the model. IThe dataset includes varying attack types and subtypes. In the training phase, the value of the neighborhood number K is modified to achieve optimal detection accuracy and the KNN model was utilized to complete learning. A small traffic data sample was used in the integration phase of the proposed tGCN-KNN. The experiments were performed for a varying number of samples under tGCN and tCNN learning models. As final, according to the comparison of three methods, it was indicated that tGCN-KNN outperforms tCNN and CNN in terms of accuracy.

In [28], a novel approach named fault injection analytics was introduced for analyzing data from fault injection experiments. This approach integrates distributed tracing to gather raw failure data and employs unsupervised ML to identify failure modes within the injected system. The primary objective is to aid human analysts in identifying failure modes more efficiently, especially when managing large volumes of data from fault-injection experiments. A new anomaly detection algorithm was proposed within this framework, designed to pinpoint unusual events in fault injection experiments. This algorithm is resilient to noise inherent in cloud systems, which can stem from non-deterministic timing and event ordering. It is also efficiently trainable with a small set of fault-free executions of the distributed system, leveraging a variable-order Markov Model. The proposed method treats the cloud-computing system as a collection of black-box communicating components, eliminating the need for prior knowledge about their internal workings. Unsupervised ML is applied to execution traces to uncover patterns of failure.

The method detects shared symbols among sequences by calculating the Longest Common Subsequence (LCS) of the sequences. The LCS represents a subset of symbols present in both sequences in the same order, obtained by minimally eliminating symbols from the original sequences. Using this probabilistic model, the method effectively detects anomalies in noisy execution traces, reducing false alarms while maintaining the detection of true anomalies. Results indicate that clustering achieves high accuracy under various conditions.

4.3. **Top Performing Machine Learning Methods for API Call Anomaly Detection: Characteristics and Effectiveness.**

In this section, we will identify the ML methods that have proven to be the most effective for anomaly detection using API calls. We will explore the characteristics that make these methods stand out, such as their ability to handle high-dimensional data, robustness to noise, computational efficiency, and interpretability. By comparing and contrasting these methods, we will highlight their strengths and weaknesses, providing a clear picture of the current best practices and innovative approaches in the field. This analysis will also suggest directions for future research and potential improvements. Table 5 summarizes method and platform based summary of the summarized studies.

In [36], an anomaly detection approach was developed utilizing NLP. The Bags of System Calls (BoSC) was used to analyze application activity on Windows virtual machines operating under the Xen hypervisor. System call traces were retrieved from both regular (benign) and malware-affected (malicious) apps utilizing virtual memory introspection. The retrieved system call sequences were preprocessed to produce valid sequences by filtering and arranging duplicate system calls. The behavior of these sequences was then investigated using NLP-based anomaly detection algorithms. The Cosine Similarity Algorithm (Co-Sim) was used to identify malicious processes on the Virtual Machine (VM). Furthermore, the Point Detection Algorithm was employed to identify the point of breach in the system call sequence. Virtual Machine Introspection (VMI) was identified as the most flexible approach for detecting, monitoring, and evaluating malware threats at the hypervisor level. In the feature extraction step, a technique called angle similarity, which is comparable to text classification for anomaly detection, was applied. In this method, a sequence of system calls was treated as a document, but individual system calls were treated as words. According to the findings, the suggested algorithms have a high detection accuracy for spotting abnormalities.

In [25] an innovative method for detecting anomalies with adversarial robustness was proposed to address vulnerabilities in existing systems against perturbation attacks. The focus of the proposed approach was on analyzing behavior units, which encapsulate representative semantic information of local behaviors to enhance the resilience of behavior analysis. A multilevel deep learning model was leveraged to understand overall semantics and contextual relationships among behavior units, effectively mitigating perturbation attacks targeting both local and large-scale behaviors. Moreover, versatility was demonstrated across different types of behavior logs, including low-level (e.g., API) and high-level (e.g., syscall) logs. The approach assumed limited attacker knowledge and incorporated threat modeling to address potential modifications to behaviors by attackers. Initially, key behavioral actions were identified from behavior sequences, followed by the use of the Longest Common Subsequence (LCS) algorithm to extract related segments that bolstered model robustness. Finally, multilayer transformer models were

implemented for feature extraction from behaviors, enabling behavior classification to determine whether a system sequence was abnormal or normal.

In [26] a novel approach was proposed where program behavior, considered as a sequence of computational steps, was represented by a single Characteristic Value (CV) rather than individual input values. This CV sequence was used as input for neural networks, resulting in improved efficiency in modeling program behavior. Multiple LSTM-RNN models were employed to reduce the neural network's input space, marking a significant advancement in program behavior modeling. The primary focus of the proposed model was on modeling program behavior using CV sequences. These sequences were utilized to represent program behavior after execution steps and were integral to the anomaly detection phase based on the constructed CV models. A custom dataset was employed to evaluate the proposed model, comparing its performance in terms of detection accuracy against existing methods.

In [33], a model for an intrusion detection system was developed that integrated various detection techniques into a single system, aiming to achieve a comprehensive view of application behaviors. The paper proposed a novel modified system calls graph that was designed to integrate and consolidate information from different techniques within a unified data structure. A deep neural network was employed to combine the results from different detection techniques used in the global model. The effectiveness of this approach was validated using three datasets of varying complexity levels. The key contribution of this study was the integration of multiple intrusion detection techniques into a unified system, leading to improved detection accuracy. The architecture of the proposed system was characterized by two main stages: detection and integration. In the detection stage, multiple intrusion detection techniques were utilized concurrently. The study employed several datasets, including DARPA, UNM, and ADFA-LD, each chosen for its distinct complexity levels. Results demonstrated significant improvements in detection accuracy compared to using individual techniques, with higher detection rates and reduced false positives achieved by the proposed model.

TABLE 5. Method and platform based summary of the methods

Reference	ML Method	Applied platform	Success Metrics
[21]	DNN	General	Accuracy, loss rate, TNR, precision, F1-score, FPR, sensitivity, FNR, G-mean
[22]	CNN/RNN, LSTM, WaveNet, ECOD	Linux Kernel	FPR, F1-score, time efficiency, AUC, TPR
[23]	DNN, CNN	Container Platforms (Kubernetes cluster)	Accuracy, NPV, coverage, sensitivity, FPR, F1-score, ROC
[24]	LSTM	Linux, Cloud	Accuracy, loss rate, FPR, F1-score, sensitivity
[25]	Multi-layer DL	Android	Precision, F1-score, ROC
[8]	LSTM, CNN	Linux	Accuracy, loss rate, detection rate, FAR, training time
[26]	LSTM-RNN	General	Accuracy, detection rate, AUROC, AUPR, CPU cycle count, complexity, memory usage
[27]	iForest	Smart Traffic System with Sensor Devices	Detection rate
[28]	Not provided	Cloud Systems	FAR, time efficiency, TPR, computational cost
[29]	tGCN-KNN	General	Accuracy, precision
[31]	DT, ANN	Container Clouds	Loss rate, F1-score, precision, recall
[30]	1-SVM, LOF, iForest	Linux	AUROC, AUC
[32]	Hierarchical Clustering	Kernel Events, Operating System (OS)	Precision, FPR, complexity, FNR
[33]	D-MLP	Linux	Detection rate, FPR, ROC
[34]	SVM, K-means, Dbscan	Linux	Accuracy, time efficiency
[35]	ANN	General, IoT	Accuracy, FPR, TPR
[37]	SVM, LR, KNN	Linux	Accuracy, AUC, ROC
[38]	IF, LOF, OC-SVM, KNN	Linux, Windows	FPR, computational cost, TPR
[39]	OC-SVM, LSTM, SVDD	General	AUC, ROC
[40]	Word2vec	General	FPR, TPR, computational cost

5. OPEN PROBLEMS AND CHALLENGES

While anomaly detection using API calls is very significant and quite functional, there are some open issues that have not been resolved by researchers and application developers. Some of these challenges are as follows:

- **Imbalanced datasets:** The performance of ML methods is significantly affected by the imbalanced datasets used for anomaly detection. Typically, the volume of data representing anomalous behavior is significantly smaller than that representing normal system behavior. This imbalance can lead to inadequate performance of ML methods in both the training and testing phases, resulting in inadequate success metrics. Furthermore, labeling the available dataset is often time-consuming and resource-intensive, especially when dealing with large and diverse data volumes.
- **High data volume in processing:** ML models applied to high-volume datasets may struggle to perform efficiently under heavy loads. The scalability of these models is directly affected by the increasing number of API calls. The literature shows that performance issues arising from high data volumes in distributed and cloud systems represent a critical area for improvement.
- **Open source datasets:** Datasets available in open sources often contain sensitive information within API calls, raising concerns about privacy violations. Therefore, when creating datasets related to application and system behavior, it is crucial to prioritize privacy and data security.
- **Real-time Processing:** The ability of ML models to detect anomalies in real-time remains a significant challenge. Real-time processing of data requires advanced algorithms and infrastructure that can handle large-scale, high-speed data streams without compromising accuracy or speed.
- **Adaptability to Emerging Threats:** Anomaly detection systems must constantly adapt to new and evolving threats. Static models can quickly become obsolete, requiring the development of adaptive learning techniques that can update and evolve in response to new data and threat patterns.
- **Explainability and Interpretability:** The black box nature of many ML models poses a problem for understanding and interpreting results. Developing methods to make anomaly detection models more transparent and interpretable is crucial for their practical application and reliability.
- **Integration with Existing Systems:** It is often difficult to seamlessly integrate anomaly detection systems with existing IT infrastructure and workflows. Ensuring compatibility and minimal disruption to existing processes requires advanced integration strategies and tools.

6. CONCLUSIONS

In this paper, a systematic literature review on anomaly detection using ML methods with API calls is presented by providing a systematization of information. A research methodology is established by selecting appropriate search keywords and the searching sentences are generated with these keywords. Common databases are used in advanced mode to use generated searching terms. Research questions are determined and inclusion and exclusion criteria are defined to filter the studies according to the focus of the literature. Over 30 research papers are summarized and compared based on different criteria. Fundamental concepts related to API calls, machine learning fundamentals, and the scope of our review

are summarized to provide a foundation. Through systematic research and analysis, it is obtained that models such as KNN, SVM, LSTM, and CNN are frequently used for anomaly detection with API calls. Additionally, open-source datasets like ADFA-LD, DARPA, and UNM are generally preferred for classification and detection. It is also observed that custom datasets are often created using various tools from operating systems like Linux. Metrics such as accuracy, F1-score, recall, and precision are used to measure the performance of the models in the studies.

DECLARATIONS

- **Conflict of Interest:** The authors are not affiliated with any entity that has a direct or indirect in the subject matter covered in this paper.

REFERENCES

[1] S. Garg, S. Batra, A novel ensembled technique for anomaly detection, International Journal of Communication Systems 30 (11) (2017) e3248.

[2] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, N. F. Samatova, Anomaly detection in dynamic networks: a survey, Wiley Interdisciplinary Reviews: Computational Statistics 7 (3) (2015) 223–247.

[3] M. Ahmed, A. N. Mahmood, M. R. Islam, A survey of anomaly detection techniques in financial domain, Future Generation Computer Systems 55 (2016) 278–288.

[4] D. Alsalman, A comparative study of anomaly detection techniques for iot security using amot (adaptive machine learning for iot threats), IEEE Access (2024).

[5] B. Jin, S. Sahni, A. Shevat, Designing Web APIs: Building APIs That Developers Love, " O'Reilly Media, Inc.", 2018.

[6] A. Almaleh, R. Almushabb, R. Ogran, Malware api calls detection using hybrid logistic regression and rnn model, Applied Sciences 13 (9) (2023) 5439.

[7] Y. Li, F. Kang, H. Shu, X. Xiong, Y. Zhao, R. Sun, Apiaso: A novel api call obfuscation technique based on address space obscurity, Applied Sciences 13 (16) (2023) 9056.

[8] F. Osamor, B. Wellman, Deep learning-based hybrid model for efficient anomaly detection, International Journal of Advanced Computer Science and Applications 13 (4) (2022).

[9] U. S. Shanthamallu, A. Spanias, C. Tepedelenlioglu, M. Stanley, A brief survey of machine learning methods and their sensor and iot applications, in: 2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA), IEEE, 2017, pp. 1–8.

[10] I. Muhammad, Z. Yan, Supervised machine learning approaches: A survey, ICTACT Journal on Soft Computing 5 (3) (2015).

[11] I. Rish, et al., An empirical study of the naive bayes classifier, in: IJCAI 2001 workshop on empirical methods in artificial intelligence, Vol. 3, Citeseer, 2001, pp. 41–46.

[12] E. Min, J. Long, Q. Liu, J. Cui, W. Chen, Tr-ids: Anomaly-based intrusion detection through text-convolutional neural network and random forest, Security and Communication Networks 2018 (1) (2018) 4943509.

[13] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is "nearest neighbor" meaningful?, in: Database Theory—ICDT'99: 7th International Conference Jerusalem, Israel, January 10–12, 1999 Proceedings 7, Springer, 1999, pp. 217–235.

[14] H. Liu, B. Lang, Machine learning and deep learning methods for intrusion detection systems: A survey, applied sciences 9 (20) (2019) 4396.

[15] Y. Liu, X. Hao, B. Zhang, Y. Zhang, Simplified long short-term memory model for robust and fast prediction, Pattern Recognition Letters 136 (2020) 81–86.

[16] S. Yang, A. Jin, W. Nie, C. Liu, Y. Li, Research on ssa-lstm-based slope monitoring and early warning model, Sustainability 14 (16) (2022) 10246.
[17] J. Bernal, K. Kushibar, D. S. Asfaw, S. Valverde, A. Oliver, R. Martí, X. Lladó, Deep convolutional neural networks for brain image analysis on magnetic resonance imaging: a review, Artificial intelligence in medicine 95 (2019) 64–81.
[18] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.
[19] R. Yamashita, M. Nishio, R. K. G. Do, K. Togashi, Convolutional neural networks: an overview and application in radiology, Insights into imaging 9 (2018) 611–629.
[20] G. Yao, T. Lei, J. Zhong, A review of convolutional-neural-network-based action recognition, Pattern Recognition Letters 118 (2019) 14–22.
[21] A. Akagic, I. Džafić, Enhancing smart grid resilience with deep learning anomaly detection prior to state estimation, Engineering Applications of Artificial Intelligence 127 (2024) 107368.
[22] G. Duan, Y. Fu, M. Cai, H. Chen, J. Sun, Dongting: A large-scale dataset for anomaly detection of the linux kernel, Journal of Systems and Software 203 (2023) 111745.
[23] S. L. Rocha, F. L. L. de Mendonca, R. S. Puttini, R. R. Nunes, G. D. A. Nze, Dcids—distributed container ids, Applied Sciences 13 (9301) (2023) 9301.
[24] A. Chaudhari, B. Gohil, U. P. Rao, A novel hybrid framework for cloud intrusion detection system using system call sequence analysis, Cluster Computing (2023) 1–17.
[25] D. Zhan, K. Tan, L. Ye, X. Yu, H. Zhang, Z. He, An adversarial robust behavior sequence anomaly detection approach based on critical behavior unit learning, IEEE Transactions on Computers (2023).
[26] S. Ahn, H. Yi, H. Bae, S. Yoon, Y. Paek, Data embedding scheme for efficient program behavior modeling with neural networks, IEEE Transactions on Emerging Topics in Computational Intelligence 6 (4) (2022) 982–993.
[27] A. Karamanou, P. Brimos, E. Kalampokis, K. Tarabanis, Exploring the quality of dynamic open government data using statistical and machine learning methods, Sensors 22 (24) (2022) 9684.
[28] D. Cotroneo, L. De Simone, P. Liguori, R. Natella, Fault injection analytics: A novel approach to discover failure modes in cloud-computing systems, IEEE transactions on dependable and secure computing 19 (3) (2020) 1476–1491.
[29] Y. Wang, Y. Jiang, J. Lan, Intrusion detection using few-shot learning based on triplet graph convolutional network, Journal of Web Engineering 20 (5) (2021) 1527–1552.
[30] C. Kim, M. Jang, S. Seo, K. Park, P. Kang, Intrusion detection based on sequential information preserving log embedding methods and anomaly detection algorithms, IEEE Access 9 (2021) 58088–58101.
[31] R. R. Karn, P. Kudva, H. Huang, S. Suneja, I. M. Elfadel, Cryptomining detection in container clouds using system calls and explainable machine learning, IEEE transactions on parallel and distributed systems 32 (3) (2020) 674–691.
[32] O. M. Ezeme, A. Azim, Q. H. Mahmoud, Peskea: Anomaly detection framework for profiling kernel event attributes in embedded systems, IEEE Transactions on Emerging Topics in Computing 9 (2) (2020) 957–971.
[33] F. J. Mora-Gimeno, H. Mora-Mora, B. Volckaert, A. Atrey, Intrusion detection system based on integrated system calls graph and neural networks, IEEE Access 9 (2021) 9822–9833.
[34] I. Kohyarnejadfard, D. Aloise, M. R. Dagenais, M. Shakeri, A framework for detecting system performance anomalies using tracing data analysis, Entropy 23 (8) (2021) 1011.
[35] H. Kim, S. Ahn, W. R. Ha, H. Kang, D. S. Kim, H. K. Kim, Y. Paek, Panop: Mimicry-resistant ann-based distributed nids for iot networks, IEEE Access 9 (2021) 111853–111864.
[36] S. K. Peddoju, H. Upadhyay, J. Soni, N. Prabakar, Natural language processing based anomalous system call sequences detection with virtual memory introspection, International Journal of Advanced Computer Science and Applications 11 (5) (2020).
[37] Y. Shin, K. Kim, Comparison of anomaly detection accuracy of host-based intrusion detection systems based on different machine learning algorithms, International Journal of Advanced Computer Science and Applications 11 (2) (2020).

[38] Z. Liu, N. Japkowicz, R. Wang, Y. Cai, D. Tang, X. Cai, A statistical pattern based feature extraction method on system call traces for anomaly detection, Information and Software Technology 126 (2020) 106348.
[39] T. Ergen, S. S. Kozat, Unsupervised anomaly detection with lstm neural networks, IEEE transactions on neural networks and learning systems 31 (8) (2019) 3127–3141.
[40] L. Liu, C. Chen, J. Zhang, O. De Vel, Y. Xiang, Insider threat identification using the simultaneous neural learning of multi-source logs, IEEE Access 7 (2019) 183162–183176.
[41] Q. Chen, R. Luley, Q. Wu, M. Bishop, R. W. Linderman, Q. Qiu, Anrad: A neuromorphic anomaly detection framework for massive concurrent data streams, IEEE transactions on neural networks and learning systems 29 (5) (2017) 1622–1636.
[42] S. Lv, J. Wang, Y. Yang, J. Liu, Intrusion prediction with system-call sequence-to-sequence model, IEEE Access 6 (2018) 71413–71421.
[43] O. M. Ezeme, Q. H. Mahmoud, A. Azim, Dream: deep recursive attentive model for anomaly detection in kernel events, IEEE Access 7 (2019) 18860–18870.
[44] W. Haider, J. Hu, Y. Xie, X. Yu, Q. Wu, Detecting anomalous behavior in cloud servers by nested-arc hidden semi-markov model with state summarization, IEEE Transactions on Big Data 5 (3) (2018) 305–316.