IEEE Open Journal of the Computer Society

Received 3 March 2024; revised 22 March 2024; accepted 7 April 2024. Date of publication 9 April 2024;
date of current version 25 April 2024. The review of this paper was arranged by Associate Editor Zhaolong Ning.

Digital Object Identifier 10.1109/OJCS.2024.3386715

Anomaly Detection in the Key-Management Interoperability Protocol Using Metadata

MIR ALI REZAZADEH BAEE [1] (Senior Member, IEEE), LEONIE SIMPSON [1], AND WARREN ARMSTRONG [2]

[1]School of Computer Science, Queensland University of Technology, Brisbane, QLD 4000, Australia
[2]QuintessenceLabs, Canberra, ACT 2609, Australia

CORRESPONDING AUTHOR: MIR ALI REZAZADEH BAEE (e-mail: info@baee.co).

The work was supported by the Cyber Security Research Center Limited funded by Australian Government's Cooperative Research Centers Program.
This article has supplementary downloadable material available at https://doi.org/10.1109/OJCS.2024.3386715, provided by the authors.

ABSTRACT Large scale enterprise networks often use Enterprise Key-Management (EKM) platforms for unified management of cryptographic keys. In such a system, requests and responses commonly use the Key Management Interoperability Protocol (KMIP) format. The KMIP client and server use Transport Layer Security (TLS) to negotiate a mutually-authenti cated connection. Although KMIP traffic is encrypted, monitoring traffic and usage patterns of EKM Systems (EKMS) may enable detection of anomalous (possibly malicious) activity in the enterprise network that is notdetectable by other means. Metadata analysis of enterprise system traffic has been widely studied (for example at the TLS protocol level). However, KMIP metadata in EKMS has not been used for anomaly detection. In this paper, wepresent a framework for automated outlier rejection and anomaly detection. This involves investigati on of KMIP metadata, determining characteristics to extract for dataset generation, and looking for patt erns from which behaviors can be inferred. For automated labeling and detection, a deep learning-based model is applied to thegenerated datasets: Long Short-Term Memory (LSTM) auto-encoder neural networks with specific parameters. As aproof of concept, we simulated an enterprise environment, collected relevant KMIP metadata, and deployed this framework. Although our implementati on used Quintessence Labs EKMS, the framework we proposed is vendorneutral. The experimental results (Precision, Recall, F1 = 1.0) demonstrate that our framework can accurately detectall anomalous enterprise network activities. This approach could be integrated with other enterprise information toenhance detection capabilities. Our proposal can be used as a general-purpose framework for anomaly detecti on and diagnosis.

INDEX TERMS KMIP metadata analysis, deep learning, anomaly detection, enterprise key-management system, framework.

I. INTRODUCTION

Business operations using enterprise information systems produce masses of data, much of this transmitted over communications networks. In response to rising security concerns, encryption is increasingly used to protect stored and transmitted enterprise data [1].

Applying encryption requires the use of cryptographic keys, so key management is also critical in achieving organizational data protection objectives [2], [3], [4]. Many organizations use Enterprise Key Management (EKM) platforms for unified management of the various cryptographic keys needed.

Network traffic inspection devices are unable to inspect encrypted payload data so, although encryption is intended as a security measure, it may be used by attackers to facilitate exfiltration of data from an organization. However, EKM usage provides additional data that can be used for anomaly detection [5]. This paper presents the results of analysis of the metadata associated with the use of an EKM platform, and demonstrates that this can provide useful information to determine whether activity is normal or abnormal.

Consider the workflow of an EKM architecture and communication model as a Client-Server model (shown in Fig. 1). Requests and responses commonly use the Key Management

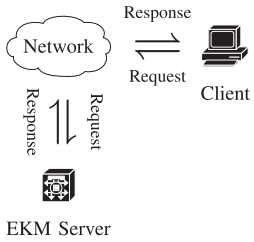

FIGURE 1. The EKM network model.

Interoperability Protocol (KMIP) [6] format. The client makes a request (e.g., a "Create-key" request) through the Application Programming Interface (API), using a client-specific representation. This is encoded in a KMIP format and transmitted to the server. The server decodes the request to form an intermediate representation; the server API uses this to process the request.

One format for KMIP messages (known as TTLV format [7]) includes these elements: the (mandatory) raw Tag (e.g., attribute names), Type (e.g., String, integer), Length (the size of the data), Value (the carried payload). Alternative message encoding formats to TTLV encoding include: Hypertext Transfer Protocol (HTTP), JavaScript Object Notation (JSON), and Extensible Markup Language (XML). In this research paper, we use TTLV formatting. The interested reader can find additional information related to KMIP message encoding formats in [6].

```
REQUEST_MESSAGE:STRUCTURE(96):stru1
 REQUEST_HEADER:STRUCTURE(56):stru2
  PROTOCOL_VERSION:STRUCTURE(32):stru3
   PROTOCOL_VERSION_MAJOR:INTEGER(4):1
   PROTOCOL_VERSION_MINOR:INTEGER(4):4
  BATCH_COUNT:INTEGER(4):1
 REQUEST_BATCH_ITEM:STRUCTURE(24):stru2
  OPERATION:ENUMERATION(4):DISCOVER_VERSIONS
  REQUEST_PAYLOAD:STRUCTURE(0):stru3
RESPONSE_MESSAGE:STRUCTURE(328):stru1
 RESPONSE_HEADER:STRUCTURE(72):stru2
  PROTOCOL_VERSION:STRUCTURE(32):stru3
   PROTOCOL_VERSION_MAJOR:INTEGER(4):1
   PROTOCOL_VERSION_MINOR:INTEGER(4):4
  TIME_STAMP:DATE_TIME(8):Wed Dec 31 19:00:00 1969
  BATCH_COUNT:INTEGER(4):1
 REQUEST_BATCH_ITEM:STRUCTURE(240):stru2
  OPERATION:ENUMERATION(4):DISCOVER_VERSIONS
  RESULT_STATUS:ENUMERATION(4):SUCCESS
```

LISTING 1. KMIP metadata example (discover-version).

In the context of EKMS, metadata includes data about the creation and use of the cryptographic keys, but not the keys themselves. Listings 1 and 2 are examples of the KMIP metadata associated with a client "Create-key" request sent to an EKM server. These are generated using TSF (Trusted Security Foundation) EKM solution provided by QuintessenceLabs [8] and extracted through monitoring, logging, and decrypting the encrypted network traffic using Wireshark [9]. Each KMIP request receives two separate responses. First, the list of protocol versions is returned in a "Discover-Version" response (refer to Listing 1). Then, the response to the actual client request e.g., "Create-Key" is provided by the EKM server (refer to Listing 2). Note that this is specific to QuintessenceLabs implementation, wherein their client-side software always

```
1- REQUEST_MESSAGE:STRUCTURE(376):stru1
2-  REQUEST_HEADER:STRUCTURE(56):stru2
3-   PROTOCOL_VERSION:STRUCTURE(32):stru3
4-    PROTOCOL_VERSION_MAJOR:INTEGER(4):1
5-    PROTOCOL_VERSION_MINOR:INTEGER(4):4
6-   BATCH_COUNT:INTEGER(4):1
7-  REQUEST_BATCH_ITEM:STRUCTURE(304):stru2
8-   OPERATION:ENUMERATION(4):CREATE
9-   REQUEST_PAYLOAD:STRUCTURE(264):stru3
10-    OBJECT_TYPE:ENUMERATION(4):SYMMETRIC_KEY
11-    TEMPLATE_ATTRIBUTE:STRUCTURE(240):stru4
12-     ATTRIBUTE:STRUCTURE(64):stru5
13-      ATTRIBUTE_NAME:TEXT_STRING(4):Name
14-      ATTRIBUTE_VALUE:STRUCTURE(40):stru6
15-       NAME_VALUE:TEXT_STRING(9):New_Key_0
16-      ATTRIBUTE:STRUCTURE(48):stru6
17-       ATTRIBUTE_NAME:TEXT_STRING(20):Length
18-       ATTRIBUTE_VALUE:INTEGER(4):256
19- RESPONSE_MESSAGE:STRUCTURE(208):stru1
20-  RESPONSE_HEADER:STRUCTURE(72):stru2
21-   PROTOCOL_VERSION:STRUCTURE(32):stru3
22-    PROTOCOL_VERSION_MAJOR:INTEGER(4):1
23-    PROTOCOL_VERSION_MINOR:INTEGER(4):4
24-   TIME_STAMP:DATE_TIME(8):Wed Dec 31 19:00:00 1969
25-   BATCH_COUNT:INTEGER(4):1
26-  REQUEST_BATCH_ITEM:STRUCTURE(120):stru2
27-   OPERATION:ENUMERATION(4):$CREATE$
28-   RESULT_STATUS:ENUMERATION(4):SUCCESS
29-   RESPONSE_PAYLOAD:STRUCTURE(64):stru3
30-    OBJECT_TYPE:ENUMERATION(4):SYMMETRIC_KEY
```

LISTING 2. KMIP metadata example (create-key request).

sends an initial Discover-Versions message as a way of testing the network connection. This is not a KMIP requirement.

In Listing 2 the following important questions are answered using the KMIP metadata:

- What was the request message structure? *Line 1*
- What was the requested operation? *Lines 8, 10*
- What was the applied parameter? *Lines 13, 15, 17, 18*
- What was the response message structure? *Line 19*
- When was the response message sent? *Line 24*
- What was the status of the operation? *Lines 28, 30*

Note that the metadata colored in Red contains information needed to understand the creation of a cryptographic key, but does not contain the value of the key itself. Also note that more complex messages are possible. For example, there is an optional additional authentication field, which can be used in the same way as the Unix *'su'* command can be used, so that the identity of the user performing the operation is not tied to the TLS certificate securing the connection. This is used by a computer user to execute commands with the privileges of another user account.

A. RESEARCH MOTIVATION

We expect that metadata associated with KMIP activities may reveal patterns that can be used for automated anomaly detection. This motivated our investigation and attempts to generate heuristics-based rules that describe normal behavior. Once these rules are established, metadata obtained through future monitoring of systems can be analyzed and compared with the established baseline to identify anomalous events. A substantial difference from the baseline data may indicate a cyber security breach in the enterprise network. The investigation of KMIP metadata for use in this way represents a novel case study in metadata analysis in EKM systems.

B. RESEARCH CHALLENGE

The increased use of cryptographic techniques to protect enterprise data has grown rapidly, resulting in increased need for EKM systems. Such systems have been described in existing literature, including standards (See NIST SP800-57 [10]). Metadata analysis techniques have been widely applied in network security to build profiles of normal and anomalous behaviour for use in intrusion detection [11]. However, to the best of our knowledge, there is no public research on the application of KMIP metadata obtained from EKM systems for anomaly detection. This additional EKMS metadata may be useful in enhancing anomaly detection. The research challenge is to determine whether specific KMIP metadata characteristics can be extracted and used for effective anomaly detection, in a range of enterprise contexts.

C. RESEARCH APPROACH

Machine Learning (ML) solutions have been applied in large enterprise systems to establish effective intrusion detection systems [12]. Auto-encoders are commonly applied in scenarios where the process needs to learn to represent a time-series. Such an approach has been successfully applied to operating system log files to develop time-series-based intrusion detection using Long Short Term Memory (LSTM) [13], [14]. Furthermore, when the underlying data is inherently unpredictable, reconstruction-based detection such as auto-encoders have been shown to be more efficient than prediction-based detection [15].

This research applies an LSTM-based approach to model KMIP activities, in two stages. First, we generate KMIP datasets from captured normal protocol behaviors. In the second stage, the generated KMIP metadata datasets are analyzed to establish heuristics indicative of specific enterprise activity states (normal/abnormal). Automating this process permits exploration of the datasets to determine optimal factors and weightings for a given context to form a model capable of detecting anomalous behavior.

D. RESEARCH BRIEF

This research simulated various enterprise contexts and operations involving the use of an EKM system, and captured the associated KMIP metadata. In our simulation, we used the TSF EKM solution provided by QuintessenceLabs [8]. This could have been similarly performed with another EKM system. To enable differentiation between normal and abnormal enterprise behavior using information obtained from the KMIP metadata, datasets are generated for a variety of use conditions. The generated KMIP metadata datasets are analyzed to establish heuristics indicative of specific enterprise activity states (normal/abnormal). This is performed in a Behavioral Security Analysis Engine (BSAE). Automating this process permits extensive exploration of the datasets. This permits identification of indicators of abnormal behavior.

E. RESEARCH CONTRIBUTION

To the best of our knowledge, this is the first paper on enterprise network anomaly detection using KMIP metadata. Our contributions include:

1) A process to generate datasets[1] containing KMIP metadata for enterprise activities.
2) A process to identify those KMIP metadata elements distinctly associated with normal/abnormal behaviors.
3) Application of LSTM auto-encoder neural networks with specific parameters for automated labeling (outlier rejection) and anomaly detection.

We apply our framework in two case studies as a proof of concept. We implemented a small network and included an Enterprise Key and Policy Manager solution provided by QuintessenceLabs [8]. We simulated some enterprise operations, collected KMIP metadata, and performed metadata analysis. Our experiments clearly show that this approach effectively identifies anomalous network activities.

F. ORGANIZATION OF THE PAPER

This paper is organized as follows. Section II gives an overview of existing anomaly detection proposals. Section III introduces the network model, discusses the security objectives and assumptions and defines the capabilities of an adversary. Section IV introduces our framework for anomaly detection using KMIP metadata. Section V describes our case studies where the framework is applied. Section VI explains our experimental settings for implementation and defines the metrics used for evaluating our model. In Section VII, data generation and preparation for automated labeling and detection are discussed. Section VIII reports on our experimental results and discusses the model performance; the detection capability is clearly demonstrated. We conclude the paper in Section IX.

II. EXISTING WORK ON ANOMALY DETECTION

The IEEE Standard 1044-2010 [16] defines the term "anomaly" as *"any abnormality, irregularity, inconsistency, or variance from expectations. It may be used to refer to a condition or an event, to an appearance or a behavior, to a form or a function."* Anomalies can be grouped into three major categories: point anomalies (where a single data sample is anomalous to the rest of the data), collective anomalies (where a collection of data samples occuring together appear to be anomalous although the individual instances may not be), and contextual anomalies (where the data appear to be anomalous in a specific context defined for a particular problem) [17].

Anomaly detection methods based on machine learning can be classified into two broad categories: supervised and unsupervised. Classification is based on the machine learning techniques employed and the type of data involved [5]. In supervised methods, model training requires labeled data points

[1]The open-source datasets produced and shared by this research study to provide a support, accessible via: https://baeeco-my.sharepoint.com/:f:/g/personal/mar_baee_co/EnBVsufPGwtAkrWgG3gUWFwBAwVVWua7V1EyCdwvmw4PhA

that clearly specify the normal instances and the abnormal instances. Then, the trained model can maximize discrimination between normal and abnormal instances. In unsupervised methods, model training does not need labels. Unsupervised training works where abnormal instances are observed to be outlier points that are distant from other instances. Unsupervised learning techniques, such as clustering, can be applied.

A detailed review and evaluation paper by He et al. [18] compares six state-of-the-art log-based anomaly detection methods. These are three unsupervised methods (Principal Component Analysis (PCA) [19], Invariant Mining [20], and Log Clustering [21]) and three supervised methods (Decision Tree [22], Support Vector Machines (SVM) [23], and Logistic Regression [24]). He et al. [18] noted that supervised methods can detect anomalies in a much shorter time (less than one minute) than unsupervised methods. However, supervised methods require labeled datasets, and these are not always available, particularly in network security applications.

Recently, Bitton and Shabtai [25] proposed a network-based intrusion detection system for securing remote desktop connections at the operating system level. Their proposal utilizes machine learning for detecting malicious network packets, which may carry dangerous exploits to the remote desktop server. Their approach comprises multiple anomaly detection models such as k-means clustering and the Cluster-Based Local Outliers Factor (CBLOF) [26]. They conducted an empirical evaluation on an avionic system setup consisting of a commercial tablet connected to a real electronic flight bag server through a remote desktop connection. Their results show that the proposed method can detect malicious packets carrying known exploits. The model is shown to be accurate at anomaly detection. A very low false positive rate and a high true positive rate were obtained, outperforming current state-of-the-art algorithms. In addition, the computational complexity of their proposed model is linear with the size of the packets. Empirical analysis of the model shows that the average processing delay is not noticeable by the user. However, this proposal is based on a supervised learning approach, as the entire training dataset contained labeled data points (for both legitimate and malicious sessions).

In the last few years, deep machine learning techniques have been widely used for log anomaly detection at the operating system level due to their significant improvement in performance over classical machine learning approaches. For instance, Du et al. [13] introduce DeepLog, which first models log entries as a sequence and then trains the normal sequence with LSTM networks. DeepLog is mainly designed for collective anomaly detection but it can also detect point anomalies. It flags anomaly as "True" if there are data elements that do not conform with the normal model. Extensive experimental evaluations over large log data show that DeepLog outperformed other existing log-based anomaly detection methods based on traditional data mining methodologies. DeepLog applies a supervised approach in the training stage; all entries must come from a normal system execution path. However, normal system execution data is not always easily collectable due to large number of entries in a log file that may contain anomalies. Hence, this approach requires an additional strategy to perform normal log collection.

Vinayakumar et al. [14] review the effectiveness of LSTM network models to detect and classify the anomalous events accurately in sensor log files. Unlike DeepLog [13], their model was trained using data labeled as either normal or anomalous. The model is shown to be accurate at anomaly detection with a very low false positive rate and a high true positive rate. However, such a supervised learning approach that requires labeled datasets may not be suitable in network security applications.

Insider threat detection has attracted a considerable attention from both academia and industry. Yuan et al. [27] provide a framework to facilitate the detection of insider threats using LSTM units that are trained in a supervised manner. The LSTM units extract user behavior features from sequences of user actions and generate fixed-size feature matrices, to be classified as normal or anomalous. However, in addition to the limitations imposed by the requirement of labels, they use datasets with only 16 unique labels in model training.

Lu et al. [28] extend Yuan et al.'s work [27] and present an LSTM-based anomaly detection system called Insider Catcher. The LSTM model is trained based on employees' online behavior to predict a user's next possible action. As long as the prediction and the real user action do not have a significant difference, the user follows his or her normal online behavior and the action is normal. After completing the anomaly detection procedure, all records for potential threat event will be stored and passed to the insider threat analysis. These records will be compared with the user's historical records before the identification of insider threat. However, this approach requires an additional strategy to perform normal log collection from employees' online behavior.

Villarreal-Vasquez et al. [29] present LADOHD (LSTM-based Anomaly Detector Over High-dimensional Data), a generic LSTM-based anomaly detection framework to protect against insider threats. The training data contain the behavior of different actors collected by monitoring 30 isolated machines operated in normal situation where no attack was reported during the collection period. The authors highlight that LSTM-based models perform better compared to other models in the detection of anomalies. However, their approach requires an additional strategy to perform normal log collection.

To reduce the amount of noise and outliers in the training data, a well-designed framework to guide collection of such data in an automated manner is needed. Maleki et al. [30] propose a probability criterion based on the classical central limit theorem. This allows evaluation of the likelihood that a data point is drawn from non-anomalous data. Hence, this approach enables labeling of the data on the fly and ensures that no anomalous data is passed for training. However, their work assumes that much of the initial training data is clean, i.e., it requires that temporal datasets have been up and running for a sufficient time and that it is relatively anomaly-free.

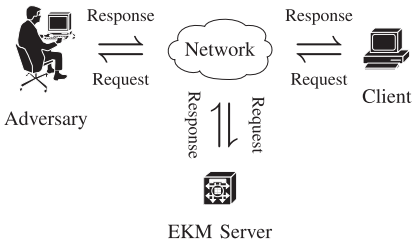

FIGURE 2. The network model including adversary.

Kennedy et al. [31], [32] present a methodology for learning without class labels. Their procedure iteratively cleans the training dataset by removing instances that have an error value above a value computed during the training steps (outliers). As the iterative process executes, the training dataset is incrementally cleaned of the minority instances, while leaving as much of the majority class as possible. The next iteration's learner is trained on a subset of data that contains a higher percentage of the majority class, its performance on unseen test data improves. However, outliers do not necessarily represent abnormal behavior. After the unsupervised outlier rejection stage, detected outliers must be verified to ensure that abnormal activities occurring with high frequency (majority) are not considered as normal. In this case, legitimate activities occurring less frequently (minority) may appear as an outlier.

The anomaly detection proposals discussed in this section were applied to log data particularly at the operating system level. To the best of our knowledge, there has been no public research into the application of encrypted KMIP metadata (under the TLS) obtained from EKM systems for anomaly detection purposes. We fill this gap in the current paper. In addition, most of these proposals require labeled datasets for training. This is not a suitable approach for detecting unknown anomalous events accurately. Further, normal datasets containing non-anomalous event logs are not always easily collectable. A well-designed framework to guide collection of such data in an automated manner is needed.

To address the above-mentioned limitation, this study provides a framework that not only facilitates automated anomaly detection in EKM systems, but can be used as a general-purpose framework to improve the design of past (e.g., DeepLog [13]) and future proposals in anomaly detection.

LSTM seems to be one of the most prominent methods for time-series data modeling and capturing long-range temporal dependencies across time steps [14]. As a proof of concept for our framework, in this study the LSTM networks [33] are used as the auto-encoder.

III. PROBLEM STATEMENT

This section outlines the network model and assumptions, the adversary's capabilities, and the design objectives.

A. NETWORK MODEL AND ASSUMPTIONS

The EKM network model considered in this study consists of a Client, an EKM Server, and an Adversary. This EKM architecture and communication model is illustrated in Fig. 2. We discuss each component below.

Client: sends different cryptographic-key-related requests to the EKM Server in accordance with KMIP. Once the client's credentials are verified (they have been authenticated and authorized), the server provides the requested services.

EKM Server: controls the distribution, use, update, and revocation of cryptographic keys. The server may provide some additional services, such as generating authentication tags or encrypting data on behalf of the clients.

Adversary: is a hostile actor with the following assumptions:

- Their goal is to steal sensitive data.
- They have established a position in the enterprise network.
- They have credentials to access some secret keys on the EKMS.
- The data generated by Adversary is anomalous due to either:
  - the use of invalid credentials to retrieve existing keys (or the previously destroyed keys) which belong to others.
  - the use of valid credentials but originating from an IP address that differs from expected address.

B. ADVERSARY'S CAPABILITIES AND MOTIVATIONS

In this work, we define the capabilities of an adversary in the EKM as monitoring data exchange between the clients and server, stealing sensitive data, delaying their transmission, as well as tampering with messages and replacing the original messages with modified versions. Furthermore, a malicious adversary may deliberately generate large amounts of both legitimate and invalid messages in a relatively short period of time to tie up server resources in processing those requests, denying its services to legitimate clients. In Section V, we outline four important scenarios that specifically address this definition and clarify the security objectives.

C. PROBLEM FORMULATION

We implement a sequence-to-sequence LSTM auto-encoder model to identify KMIP anomalous activities within the EKMS network. The objective is to reconstruct the data samples (associated with captured traffic log) using an encoded representation of the time-series input sequences.

Such a model is capable of learning what constitutes authentic system activities by removing noise during the encoding process, without the need for attack examples. Then, once it learns the non malicious patterns of the data, anything that differentiates from it would be flagged as an anomaly.

Ideally, as long as the input is non-malicious, the ideal model will always output the same sequence that was used as input. However, the reconstructed traffic originated from malicious sequences would present a high-degree of reconstruction error, thus indicating an anomaly in the source

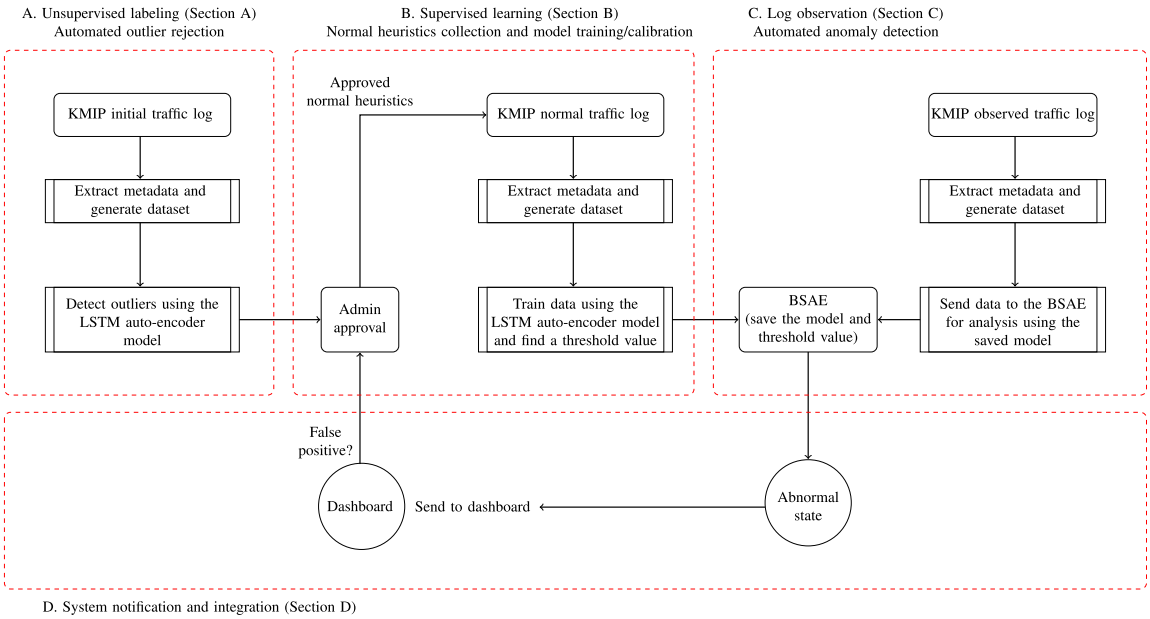

FIGURE 3. Our proposed framework. The framework involves automated outlier rejection (Stage "A"), normal heuristics collection (Stage "B"), automated anomaly detection (Stage "C"), and system notification and integration (Stage "D") with detailed explanations in Sections IV-A, IV-B, IV-C, and IV-D, respectively.

sequence. A post-processing module is thus required to process this information and to generate a configurable anomaly score that might indicate an attack.

IV. THE PROPOSED FRAMEWORK

In this work, the process of metadata analysis for anomaly detection involves four main stages: automated outlier rejection **(A)**, normal heuristics collection **(B)**, automated anomaly detection **(C)**, and system notification and integration **(D)** with other security tools. We introduce a framework that summarizes relations among different stages of the development of our analysis and detection tool. Our framework can be used to provide guidance in automating metadata analysis to use for anomaly detection.

The design and implementation of an automated metadata analysis tool is not a simple and straightforward process. Fig. 3 shows the proposed framework for anomaly detection in EKM systems. The framework uses specific ordered stages, with input to each stage defined with respect to the previous stage.

A. AUTOMATED OUTLIER REJECTION (STAGE "A")

In this stage, we capture the encrypted KMIP traffic and generate log associated with various enterprise contexts and operations within the EKM system. The log contains two general categories of network activity: normal and abnormal. Note that our final model needs to learn from the non-malicious patterns of the data; anything that differs from it will be flagged as an anomaly.

The initial log must be decrypted and fed into an initial analysis model to detect any observations that are distant from the mean or location of a distribution. These observed data points are called outliers, and do not necessarily represent abnormal behavior. If the training data (final dataset) contains anomalies, our model learns to reconstruct anomalies with a minimal error, resulting in the dismissal of similar types when new data is processed.

Before outlier detection begins, we need to generate datasets of metadata. These metadata must satisfactory answer three important questions: when (operation time), what (the requested operation), and how (message structure and operation success status). This is an unsupervised approach, as we prepare datasets of unknown (unlabeled) behavior associated with both categories of network activity: normal and abnormal. This dataset is then fed into our LSTM auto-encoder model (refer to Fig. 4), to detect any data points with high reconstruction loss value as outliers. Please note that each new dataset requires a new automated data labeling for outlier rejection. However, the trained model for anomaly detection can be saved for future anomaly detection purposes.

The architecture of our applied LSTM auto-encoder model for outlier rejection and anomaly detection is illustrated in Fig. 4. Given a multivariate sequence dataset $\{x^1, x^2, \ldots, x^T\}$, where $x^T \in \mathbb{R}_m$ represents the m-dimensional vector at time step T, the input data is squeezed into a single latent vector with smaller dimension than the original input.

This architecture applies LSTM neural network cells in the auto-encoder model based on the four steps of creating the input sequence, encoding, decoding, and measuring reconstruction loss for detecting anomalies. As a proof of concept, this study utilizes two LSTM hidden layers in both encoder and decoder.

In the encoding stage, the model builds a fixed-length vector that contains all of the information and time-wise

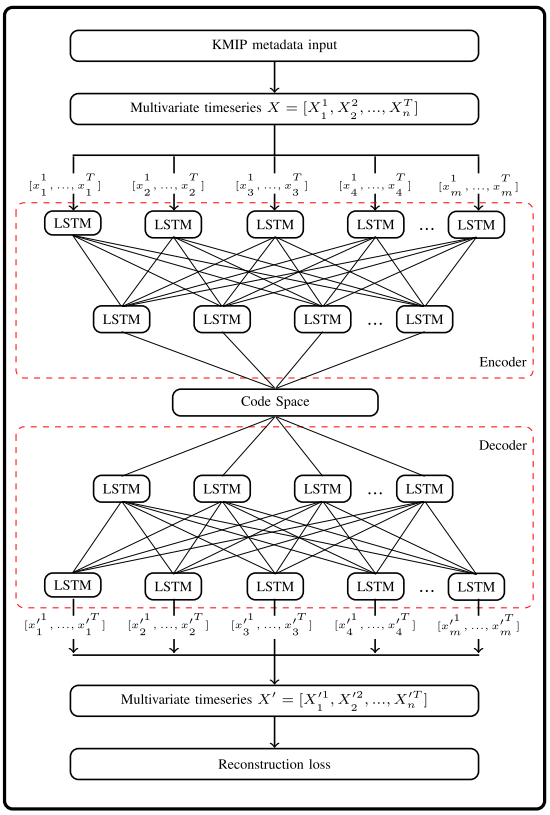

FIGURE 4. The architecture of our applied LSTM auto-encoder model for outlier rejection and anomaly detection.

relationships of the input sequence. This stage provides a compressed representation of the input data. Then, a repeat vector layer (code space) distributes the compressed representational vector across the time steps of the decoder. In the decoding stage, the model expands the compressed vector. The goal is to create an output that is as close as possible to the original input.

B. NORMAL HEURISTICS COLLECTION, MODEL TRAINING, AND CALIBRATION (STAGE "B")

After the unsupervised outlier rejection stage, detected outliers must be verified by a system administrator. This ensures that abnormal activities occurring with high frequency are not considered as normal. In this case, legitimate activities occurring less frequently may appear as an outlier in an automated process. Hence, a system administrator must perform this initial check manually until sufficient data history is acquired. Once sufficient log entries associated with normal activities have been collected, the manual administrator role will be reduced significantly, and most of the outlier rejection can be automated.

The aim is generating datasets of metadata associated with normal behavior. Similar to the previous stage, we generate datasets of metadata from which three questions are answered: when (operation time), what (the requested operation), and how (message structure and operation success status). This is a supervised approach, as we prepared datasets of normal behavior, but these are unlabeled. That is, there is no one-to-one correspondence between samples and the associated input features, and their corresponding output labels.

We use this dataset to train our model which is a combination of LSTM and auto-encoder. This permits determination of optimal factors and weightings to establish heuristics for effective detection of anomalies based on the analysis of time-series data.

Once our model (refer to Fig. 4) is trained, a suitable threshold value for identifying anomalies is determined. The distribution of the calculated loss in the training set can help us to find such a threshold. To calibrate our model, we can set a threshold above the "loss level" so that false positives are not triggered. A false positive corresponds to identification of an anomalous event, when the event is actually legitimate.

C. AUTOMATED ANOMALY DETECTION (STAGE "C")

In this stage, the trained model with a calibrated threshold value is saved into the BSAE. For example, one can save the trained model and its learned weights in the ".h5" format. This can be deployed for future anomaly detection. Using an established (saved) learning process rather than performing a new one permits us to save time in future applications.

When new KMIP traffic is observed, the associated metadata is extracted into a dataset and sent to the BSAE for differentiation between normal and abnormal enterprise behavior. Note that the observed log may contain both "normal" data (within the norm) and "abnormal" data corresponding to anomalies (exceptions to the norm). We aim to detect anything that deviates from the normal pattern "norm". This includes, for example, those data points which have not been seen before by machine.

D. SYSTEM NOTIFICATION AND INTEGRATION (STAGE "D")

The detected anomalous KMIP behavior should be integrated with existing Security Information and Event Management (SIEM) products to sharpen anomaly detection capabilities. For example, requests for key material from processes associated with staff accounts for staff known to be on leave can be indicators of potential enterprise system compromise. In such a case, a notification is sent to the SIEM dashboard to alert the administrator. If "Abnormal State" was falsely called, then the system administrator amends that specific data point into the datasets of normal behavior to prevent these false positive flags in future.

The corrected dataset of normal behavior can be fed into our model on a regular basis. For example, each night (at a fixed time) the model may undergo regular learning, with the old model values being replaced with new values, within the BSAE.

V. CASE STUDIES

This section describes two case studies, each containing two scenarios. These provide a demonstration of the effectiveness of our proposed framework.

A. CASE STUDY 1

This case study focuses on two scenarios for which a change in normal requests is detected by the BSAE. We outline each scenario and describe the associated system symptoms.

Scenario 1: Sensitive data is encrypted by keys stored within the EKMS, so the hostile actor decides to try and retrieve the encryption keys from their place of storage. They attempt to find a vulnerability in the EKMS server software to bypass authentication requirements and leak data. Their first step in this goal is to employ fuzzing tools against the EKM, trying to find an input that causes a buffer overrun, a use-after-free issue, or similar.

Symptom: KMIP messages, low volume, with abnormal contents. Abnormality may be violations of encoding (e.g., within TTLV, the L says 40 bytes but V only contains 20 bytes) or violations of protocol (e.g., required attribute missing, or irrelevant attribute included, or attribute combinations lacking internal consistency).

Scenario 2: A hostile actor has managed to obtain an encryption key used to protect network traffic on a link encryptor, and is decrypting and monitoring that traffic. The actor is aware that the organization implements date-based key rotation, and does not want the supply of intelligence to dry up. Using a credential stolen from an administrative work station, the actor sends a request to the EKMS server to modify the "Deactivation Date" and "Protect Stop Date" attributes of the currently used key, ensuring automated key rotation is delayed.

Symptom: Valid KMIP "Modify" requests which are not part of the usual network traffic.

B. CASE STUDY 2

This case study focuses on two scenarios for which a spike of KMIP requests is received by the EKMS. We outline each scenario and describe the associated system symptoms.

Scenario 3: A hostile actor has managed to log in to the computer of a trusted employee. The credentials needed to retrieve encryption keys are saved on this computer. The hostile actor retrieves the keys required to locally decrypt data stored in a remote database for local processing. The actor starts bulk downloading and decrypting the data, intending to exfiltrate it and sell it on the black market.

Symptom: The frequency of KMIP "Get-key" requests for this particular user spikes; for example, going from one request per minute to one request per second.

Scenario 4: A hostile actor has exploited a vulnerability in a web-based management screen and gained administrative access to the corporate EKMS. They have the ability to create new credentials, and they become aware that the EKMS is a virtual system and lacks a high-volume source of entropy. The actor attempts to exhaust the entropy pool of the server during a small time window, with the goal that keys created during this window will be weak (easily guessed). Using their administrative access, they create multiple new EKMS clients to avoid large spikes in the traffic of any particular user, or existing users.

Symptom: A spike in new user credentials being created/observed in use; a spike in KMIP "Create-Key/Key-Pair" requests, aggregated across all requests from all users.

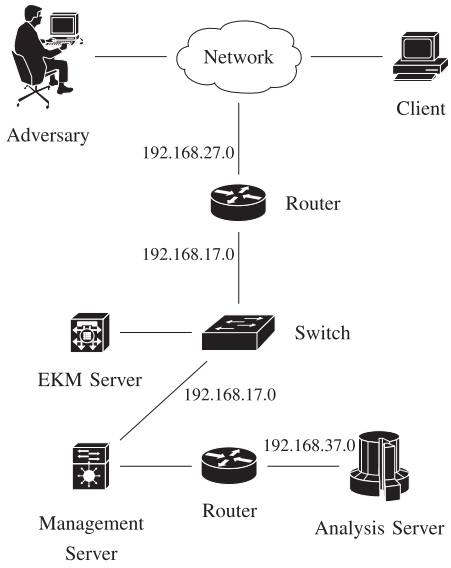

FIGURE 5. The network topology, actors, and nodes.

VI. IMPLEMENTATION AND EVALUATION

In this section, we describe the experimental settings applied in our implementation, and define the metrics used for evaluating our model.

A. EXPERIMENTAL SETTINGS

We extend the network model shown in Fig. 2 to include a Management Server, an Analysis Server, and the communication hardware between these entities. We deploy an environment to simulate this extended network model. Fig. 5 shows our implemented network topology, actors, and nodes. There are three networks, namely:

- network 192.168.27.0 that includes a client and an adversary,
- network 192.168.17.0 that includes an EKM Server and a Management Server, and
- network 192.168.37.0 that includes an Analysis Server.

This enables the collected metadata to be used in the subsequent analysis phase. The Management Server monitors the encrypted KMIP traffic and saves the captured data into a log file for decryption. After that the Management Server pushes the decryted log file to the isolated Analysis Server, which is responsible for parsing log files, training our model, and automating the outlier/anomaly detection. The result of analysis is then pushed into the Management Server for notification and integration.

We have configured our simulation environment as illustrated in Fig. 6. Three Virtual Machines (VMs) run on a VMWare Workstation. These include a TSF VM (the EKM

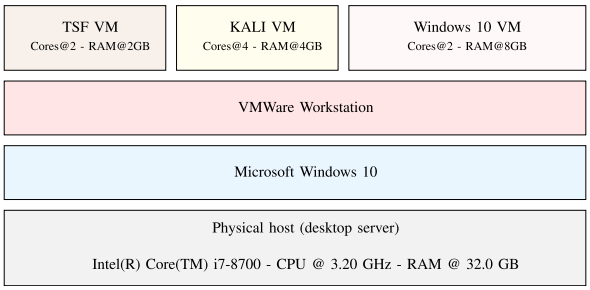

FIGURE 6. The configuration of our simulation environment.

solution provided by QuintessenceLabs), a KALI Linux VM (the Analysis Server), and a Windows 10 VM (the Management Server). The VMWare Workstation runs on a Windows 10 operating system, hosted by an Intel Core Desktop Machine.

Our Deep Learning model implementation makes use of Python 2.7 and libraries including: Numpy 1.16.1, Scipy 1.2.3, Scikit_learn 0.20.4, Theano 1.0.5, TensorFlow 2.1.0, Keras 2.7.0, Pandas 0.24.2, and Matplotlib 2.2.5 installed on a KALI GNU/Linux 2021.2.

B. EVALUATION METRICS

This section defines the metrics used in our evaluation. These are precision, recall, $F1$, and accuracy.

Precision is the ratio of the number of true positives to the sum of true and false positives. Recall focuses on measuring the ability of the methods to detect anomalous events. We can calculate precision by:

$$\text{precision} = \frac{TP}{TP + FP} \tag{1}$$

and calculate recall by:

$$\text{recall} = \frac{TP}{TP + FN}, \tag{2}$$

and use Equations 1 and 2 to calculate $F1$:

$$F1 = \frac{2 \times precision \times recall}{precision + recall}, \tag{3}$$

where TP is true positives, FP is false positives, and FN is false negatives. The accuracy score is calculated by dividing the number of correct predictions by the total prediction number.

VII. DATA GENERATION AND PREPARATION

For the set of predefined scenarios outlined in Section V, we have generated the Client to Server KMIP requests, including: Get-key, Create-key, Activate-key, Add-attribute, Modify-attribute, Revoke-key, and Destroy-key, for both normal and adversarial activities.

Through the Management Server, the KMIP traffic is logged and pushed to the Analysis Server for processing. We parsed [34] the KMIP metadata and generated datasets for training and testing. This includes the removal of characters that are common across all files (e.g., ".", "_", and blank characters). Also, empty cells are replaced with "0". The Analysis Server is to perform the tasks discussed in this stage.

VIII. PROCESS, RESULTS, DISCUSSION, AND COMPARISON

In this section, we first report on the process and our experimental results for the two case studies, in line with the four stages of our framework: automated outlier rejection **(A)**, normal heuristics collection **(B)**, automated anomaly detection **(C)**, and system notification and integration **(D)**. Then, we discuss our framework and model performance, and compared it to the other approaches published in literature.

A. PROCESS AND RESULTS

This section reports on the process and our experimental results.

Stage "A" (Identical for Case Studies 1 and 2): We capture the KMIP metadata and generate a dataset associated with various enterprise contexts and operations on the EKM Server (as outlined in Section VII). The initial dataset in Stage "A" of our framework contains 2900 data points (for 725 KMIP requests), representing mostly normal activities, with some outliers which need to be detected.

To enable our model to output a large reconstruction loss for values with lower occurrence, each parsed and normalized data cell is converted into an 8-digit hash value. The hash function property ensures that any change in the cell information (before conversion) makes a huge change after conversion. Listing 3 shows an example of KMIP metadata and the hash value associated with an "Activate-key" request from a user.

```
Metadata:
REQUESTMESSAGESTRUCTURE312stru1,STRUCTURE312stru1,STRUCTURE88stru2,
STRUCTURE32stru3,INTEGER41,INTEGER44,ENUMERATION4STOP,BOOLEAN8True,
INTEGER42,STRUCTURE160stru2,ENUMERATION4LOCATE,STRUCTURE120stru3,
STRUCTURE40stru4,ENUMERATION4ObjectType,STRUCTURE64stru4,
STRUCTURE40stru5,STRUCTURE40stru5,ENUMERATION4ACTIVATE,
STRUCTURE0stru6,0,0,0,0,0,0,0

Hashed value of Metadata:
43104290,43104150,44319384,23232140,49091028,36069548,59661531,
97101956,74212242,41298217,12779962,91136435,75049752,93923845,
80815691,91608005,91608005,50423314,66283429,41564428,41564428,
41564428,41564428,41564428,41564428,41564428
```

LISTING 3. Data preparation for Case study 1.

The initial dataset is then fed into our LSTM auto-encoder model (refer to Fig. 4). Stage "A" utilizes two LSTM hidden layers in both encoder and decoder, with 40 and 10 LSTM units, respectively. Note that our model incorporates the Rectified Linear Activation Unit or "ReLU" which is the most common choice of activation function in multi-layer neural networks or deep neural networks. We set "Adam" as our neural network optimizer algorithm. Adam is a learning rate optimization algorithm for training deep learning models, designed to accelerate the optimization process. This decreases the number of function evaluations required to reach good results. We set Mean Squared Error (MSE) for calculating our loss function. For regression tasks, MSE is a popular choice

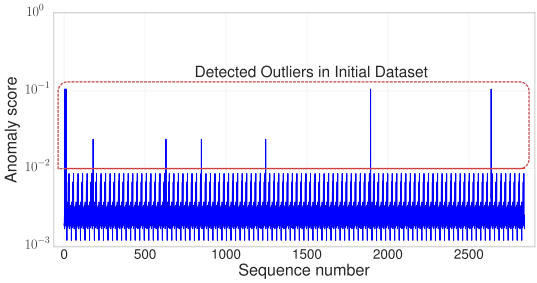

FIGURE 7. Automated outlier rejection in Stage "A".

of loss function [35]. Compared to the Mean Absolute Error (MAE), MSE is more sensitive to outliers. The number of epochs and batch size are 256 and 64, respectively.

Fig. 7 shows the detected outliers in Stage "A" using our LSTM auto-encoder model. These minority instances (within the highlighted rectangle) are distant from the mean or location of the distribution, with a reconstruction loss value greater that 10^{-2}. These observed outliers represent abnormal behavior. If the final training dataset contains anomalies, our model learns to reconstruct anomalies with a minimal error, resulting in the dismissal of similar types when new data is processed. Hence, this stage is necessary, to identify only those KMIP metadata elements distinctly associated with normal behaviors.

After removing these outliers, we have a dataset of behaviors under normal operating conditions. This dataset enables us to proceed to the Stage "B" in which the trained model for anomaly detection can be saved for future anomaly detection purposes. Please note that a newly generated dataset in Stage "A" of our framework requires a new outlier rejection.

Stages "B", "C", and "D" (for Case Study 1): The dataset used in Scenarios 1 and 2 contains 2828 data points (for 707 KMIP requests), representing normal operating conditions. This is achieved after passing the "Stage A" of our framework. The Testing dataset contains 200 data points (for 50 KMIP requests), which represents a mix of normal (144 samples) and abnormal (56 samples) operating conditions.

Each parsed and normalized data cell is converted into an 8-digit hash value. This helps our trained model to output a large reconstruction loss value for any new input data that has never seen before.

Stage "B" in Case Study 1 utilizes two LSTM hidden layers in both encoder and decoder, with 40 and 10 LSTM units, respectively. Our model incorporates the "ReLU" activation function, the "Adam" neural network optimizer algorithm, and MSE for calculating our loss function.

The dataset is split for training and validation. We instantiate and train our model on 2686 samples, and validate our model on 142 samples, where the number of epochs and batch size are 256 and 64, respectively.

Fig. 8 shows the training losses, and is used for evaluating our model's performance. We need to determine a suitable threshold value for identifying an anomaly. The distribution of the calculated loss in the Training set can help us to find such

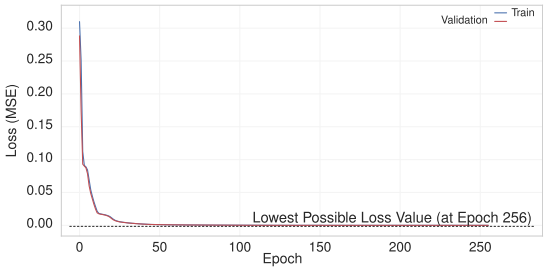

FIGURE 8. The training losses.

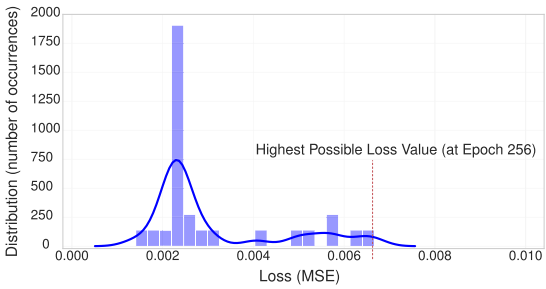

FIGURE 9. The distribution of the calculated loss in Training set.

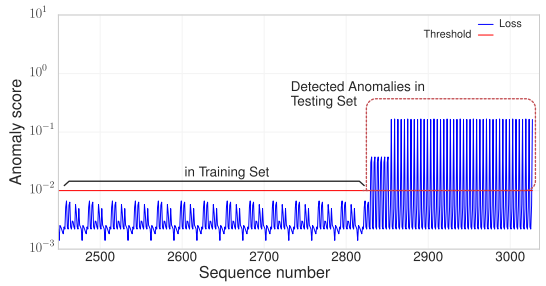

FIGURE 10. The results of anomaly detection over time.

a threshold (refer to Fig. 9). Hence, we can set a threshold above the "loss level" so that false positives are not triggered.

From the above loss distribution (Fig. 9), we see that the maximum loss value is ≈ 0.01. Therefore, we set an anomaly threshold value of 0.01, a threshold above the "loss level" so that false positives are not triggered. This flags an anomaly when the reconstruction loss in our Test set is greater than 0.01. Fig. 10 visualizes the results over time, where the red line indicates our threshold value of 0.01.

Our LSTM auto-encoder model for anomaly detection is able to flag the anomalous activities within an observed log (Stage "C"), with 100% accuracy. Listing 4 shows three rows in the final report generated for Testing set in Stage "D". Observing these three rows, two rows contain anomalous events (flagged as True) where:

1) a violations of KMIP message format is detected where an irrelevant attribute "STRUCTURE95stru1" observed,
2) 'the 'Modify Attribute" request is not part of the usual KMIP activities.

Hence, the loss value for each of the above two items goes over the Threshold value 0.01. However, the second request in the list is a part of usual KMIP activities, and as a result, the loss value for this item goes under the Threshold value 0.01.

```
REQUESTMESSAGESTRUCTURE96stru1,STRUCTURE95stru1,STRUCTURE56stru2,
STRUCTURE32stru3,INTEGER41,INTEGER44,INTEGER41,STRUCTURE24stru2,
ENUMERATION4DISCOVERVERSIONS,STRUCTURE0stru3,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0

REQUESTMESSAGESTRUCTURE96stru1,STRUCTURE96stru1,STRUCTURE56stru2,
STRUCTURE32stru3,INTEGER41,INTEGER44,INTEGER41,STRUCTURE24stru2,
ENUMERATION4DISCOVERVERSIONS,STRUCTURE0stru3,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0

REQUESTMESSAGESTRUCTURE216stru1,STRUCTURE216stru1,STRUCTURE56stru2,
STRUCTURE32stru3,INTEGER41,INTEGER44,INTEGER41,STRUCTURE144stru2,
ENUMERATION4MODIFYATTRIBUTE,STRUCTURE104stru3,STRUCTURE48stru4,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0

Loss (MSE)              Threshold       Anomaly
0.08715903994155796     0.01            True
0.002395049293256259    0.01            False
0.03028861369665133     0.01            True
```

LISTING 4. Three samples along with their loss values.

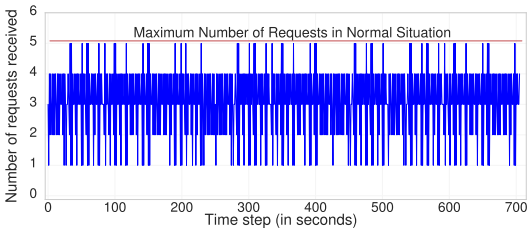

FIGURE 11. The request frequency, each one second (Training set).

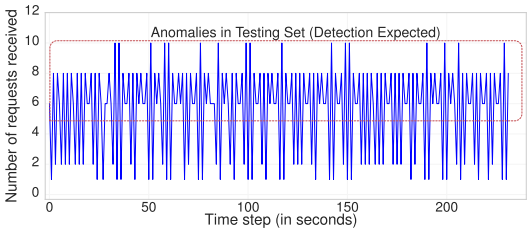

FIGURE 12. The request frequency, each one second (Testing set).

Stages "B", "C", and "D" (for Case Study 2): The dataset used in Scenarios 3 and 4 contains 2148 data points, representing normal operating conditions within 706 seconds. This is achieved after passing the "Stage A" of our framework. The Testing dataset contains 1308 data points within 232 seconds, which represents a mix of 70 normal seconds (104 data points) and 162 abnormal seconds (1204 data points). In each one second, a different number of KMIP request-response (data points) was recorded. For instance, at "01:06:08" on "2021.09.05" four client-server message exchanges were recorded. This format for recording the number of KMIP request-response per second is used for both training and testing.

Fig. 11 shows the frequency of KMIP message exchange in each one second in our Training dataset. The highest number of KMIP request-response per second is 5. Hence, we consider the upper bound to be 5 message exchanges per second.

Fig. 12 show the frequency of KMIP message exchange in each one second in our Testing dataset. The highest number of KMIP request-response per second is 10. Therefore, we expect our automated model to detect any one-second time period in which the number of message exchange is higher than our upper bound that is 5.

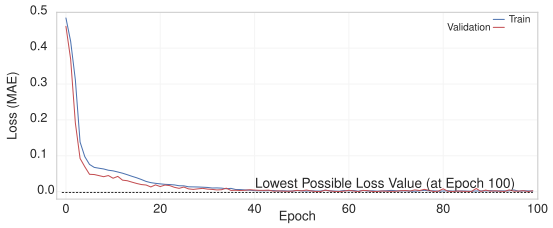

FIGURE 13. The training losses.

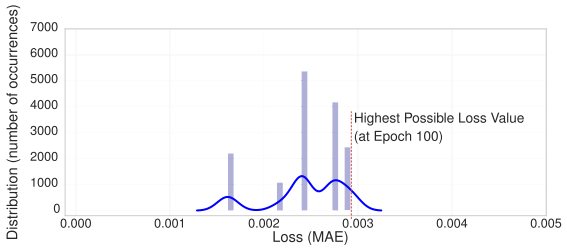

FIGURE 14. The distribution of the calculated loss in Training set.

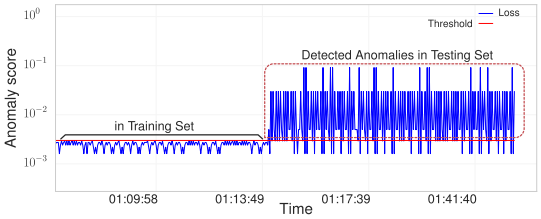

FIGURE 15. The results of anomaly detection over time.

Stage "B" in Case Study 2 utilizes two LSTM hidden layers in both encoder and decoder, with 16 and 4 LSTM units, respectively. Our model incorporates the "ReLU" activation function, the "Adam" neural network optimizer algorithm, and MAE for calculating our loss function. Please note that both MSE and MAE can be used in this Case Study. As we reduced the number of features to only the number of KMIP message exchange per seconds, MAE can produce enough sensitivity to outliers.

The dataset is split for training and validation. Then, we instantiate and train our model on 670 samples, and validate our model on 36 samples, where the number of epochs and batch size are 100 and 10, respectively.

Fig. 13 shows the training losses, and is used for evaluating our model's performance. We need to determine a suitable threshold value for identifying an anomaly. The distribution of the calculated loss in the Training set can help us to find such a threshold (refer to Fig. 14). Hence, we can set a threshold above the "loss level" so that false positives are not triggered.

From the above loss distribution (Fig. 14), we see that the maximum loss value is ≈ 0.003. Therefore, we set an anomaly threshold value of 0.003, a threshold above the "loss level" so that false positives are not triggered. This flags an anomaly when the reconstruction loss in our Test set is greater than 0.003. Fig. 15 visualizes the results over time, where the red line indicates our threshold value of 0.003.

Our LSTM auto-encoder model for anomaly detection is able to flag the anomalous activities within an observed log

TABLE 1. Comparison With Existing Frameworks

Research	Properties			
	Technique	Purpose	Unsupervised Dataset Labeling	Strong Outlier Rejection
Du et al. [13]	LSTM	anomaly detection/enterprise system	×	×
Vinayakumar et al. [14]	LSTM	anomaly detection/sensor network	×	×
Yuan et al. [27]	LSTM	anomaly detection/enterprise system	×	×
Lu et al. [28]	LSTM	anomaly detection/enterprise system	×	×
Villarreal-Vasquez et al. [29]	LSTM	anomaly detection/commercial network	×	×
Maleki et al. [30]	LSTM auto-encoder	anomaly detection/abrupt change/incident	✓	×
Kennedy et al. [31]	auto-encoder	fraud detection/credit card system	✓	×
Kennedy et al. [32]	auto-encoder	anomaly detection/cognition system	✓	×
Baee et al. [this work]	LSTM auto-encoder	anomaly detection/key-management system	✓	✓

—Note: The property is satisfied [✓]. The property is not satisfied [×].

(Stage "C"), with 100% accuracy. Listing 5 shows four rows in the final report generated for Testing set in Stage "D". Observing these four rows, two rows contain anomalous events (flagged as True) where the number of requests per second goes over 5 which means the loss value goes over the Threshold value 0.003.

```
2021-09-04 01:40:06,6
2021-09-04 01:40:07,1
2021-09-04 01:40:08,8
2021-09-04 01:40:09,2

Loss (MAE)              Threshold       Anomaly
0.0049970149993896484   0.003           True
0.0016150027513504028   0.003           False
0.029864788055419922    0.003           True
0.002921879291534424    0.003           False
```

LISTING 5. Four samples along with their loss values.

B. DISCUSSION

According to our investigations, the amount of data required for training to achieve the highest level of accuracy can be reduced from the volume we used in our investigation. In Case Study 1, we used 2828 samples under "normal" condition. We can get equivalent accuracy with 1600 samples without changing the Threshold. However, for further reductions (from 1600 to 400 samples) to retain this accuracy we needed to increase the Threshold value to ≈ 0.1.

While we did not use a GPU with TensorRT, the processing time for the model was short. The recorded training time in Case Studies 1 and 2 are ≈ 36 and ≈ 18 seconds, respectively. Also, the saved model could perform detection in a dataset of 2424 samples in ≈ 100 milliseconds.

In some scenarios there may be a level of false-positive detection. For example, if new staff come on or take on new responsibilities, "normal" access patterns may change. This change in pattern could be detected as abnormal. This detection is important in resilient systems, where system must continue to carry out their mission in the face of adversity. This issue can be solved by excluding the new staff member from the datasets during data preparation. The new staff member's activity can be separately logged (e.g., for one week) before its amendment into the main dataset of normal behavior.

TABLE 2. Comparison With Existing LSTM Auto-Encoder

Method	Results			
	Precision	Recall	F1	Dataset
Maleki et al. [30]	0.90	1.0	0.95	AWS[1]
Maleki et al. [30]	0.99	0.95	0.97	Siemens[2]
This work	1.0	1.0	1.0	QLabs EKMS[3]

1- The Amazon Web Services (AWS) monitoring CPU usage data.
2- The Siemens Industrial Turbomachinery real-time data.
3- The QuintessenceLabs (QLabs) EKM solution data (refer to Note 1).

C. COMPARISON

Table 1 presents a summary of the focus and details found in the prior work [13], [14], [27], [28], [29], [30], [31], [32] discussed in Section II, for comparison purposes with our research.

Compared to the other approaches published in literature (e.g., [13] and [14]), our framework guarantees 0% false negative detection. Also, in both Case Studies, the results prove that our framework can perform automated detection with 100% accuracy. Table 2 shows the performance comparison of our work to the detection method presented by Maleki et al. [30] which is an LSTM auto-encoder based solution.

IX. CONCLUSION

This article presented a novel framework for enterprise network anomaly detection using KMIP metadata. We applied our framework in two case studies as a proof of concept. We implemented a small network and included an Enterprise Key and Policy Manager solution provided by QuintessenceLabs, and collected KMIP metadata, generated datasets containing KMIP metadata for enterprise activities, identified those KMIP metadata elements distinctly associated with normal/abnormal behaviors, and performed metadata analysis. For automated outlier rejection (data labeling) and anomaly detection, an LSTM auto-encoder model with specific parameters was applied to the generated datasets.

Our experiments proved that monitoring traffic and usage patterns of EKM systems can enable detection of anomalous

(possibly malicious) activity in the enterprise network that is not detectable by other means. The experimental results (Precision, Recall, and F1 = 1.0) demonstrated that our framework can accurately detect all anomalous enterprise network activities.

ACKNOWLEDGMENT

All authors would like to acknowledge that they are participants in the Cyber Security Cooperative Research Center.

REFERENCES

[1] J. Hou and X. Jia, "Research on enterprise network security system," in *Proc. 2nd Int. Conf. Comput. Sci. Manage. Technol.*, 2021, pp. 216–219.
[2] M. A. R. Baee, "Privacy-preserving authentication and key management for cooperative intelligent transportation systems," Ph.D. dissertation, Dept. School Comput. Sci., Queensland Univ. Technol., Brisbane City QLD, Australia, 2021.
[3] M. A. R. Baee, L. Simpson, X. Boyen, E. Foo, and J. Pieprzyk, "A provably secure and efficient cryptographic-key update protocol for connected vehicles," *IEEE Trans. Dependable Secure Comput.*, early access, 2023, doi: 10.1109/TDSC.2023.3345406.
[4] S. Rana, F. K. Parast, B. Kelly, Y. Wang, and K. B. Kent, "A comprehensive survey of cryptography key management systems," *J. Inf. Secur. Appl.*, vol. 78, 2023, Art. no. 103607.
[5] S. Wang, J. F. Balarezo, S. Kandeepan, A. Al-Hourani, K. G. Chavez, and B. Rubinstein, "Machine learning in network anomaly detection: A survey," *IEEE Access*, vol. 9, pp. 152379–152396, 2021.
[6] T. Cox and C. White, "OASIS key management interoperability protocol (KMIP) specification version 2.0," Oct. 2019.
[7] J. Wang, X. Yang, J. Yu, L. Lu, and D. Guo, "A protocol conformance testing method based on the FSM for KMIP," in *Proc. 2nd Int. Conf. Electron., Commun. Inf. Technol.*, 2021, pp. 128–135.
[8] QuintessenceLabs; "Trusted security foundation enterprise key and policy manager," 2023. Accessed: Mar., 05, 2023. [Online]. Available: https://www.quintessencelabs.com
[9] V. Jain, *Getting Familiar With Wireshark*. Berkeley, CA, USA: Apress, 2022, pp. 35–78.
[10] E. Barker, "Recommendation for key management: Part 1–General (Nist Special Publication 800-57 Part 1 Revision 5)," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, 2020. [Online]. Available: https://csrc.nist.gov/pubs/sp/800/57/pt1/r5/final
[11] F. Skopik, M. Landauer, and M. Wurzenberger, "Online log data analysis with efficient machine learning: A review," *IEEE Secur. Privacy*, vol. 20, no. 3, pp. 80–90, May/Jun. 2022.
[12] F. Skopik, M. Landauer, and M. Wurzenberger, "Online log data analysis with efficient machine learning: A review," *IEEE Secur. Privacy*, vol. 20, no. 3, pp. 80–90, May/Jun. 2022.
[13] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*,, 2017, pp. 1285–1298.
[14] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Long short-term memory based operation log anomaly detection," in *Proc. Int. Conf. Adv. Comput., Commun. Inform.*, 2017, pp. 236–242.
[15] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based encoder-decoder for multi-sensor anomaly detection," *CoRR*, vol. abs/1607.00148, 2016, [Online]. Available: http://arxiv.org/abs/1607.00148
[16] IEEE, *IEEE Standard Classification for Softw. Anomalies*, IEEE Standard 1044-2009 (Revision of IEEE Standard 1044-1993), 2010.
[17] K. Choi, J. Yi, C. Park, and S. Yoon, "Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines," *IEEE Access*, vol. 9, pp. 120043–120065, 2021.
[18] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *Proc. IEEE 27th Int. Symp. Softw. Rel. Eng.*, 2016, pp. 207–218.
[19] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proc. ACM SIGOPS 22nd Symp. Operating Syst. Princ.*, 2009, pp. 117–132.
[20] J.-G. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, "Mining invariants from console logs for system problem detection," in *Proc. USENIX Conf. USENIX Annu. Tech. Conf.*, 2010, pp. 231–244.
[21] 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*, 2016, pp. 102–111.
[22] M. Chen, A. Zheng, J. Lloyd, M. Jordan, and E. Brewer, "Failure diagnosis using decision trees," in *Proc. Int. Conf. Autonomic Comput.*, 2004, pp. 36–43.
[23] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo, "Failure prediction in IBM BlueGene/L event logs," in *Proc. IEEE 7th Int. Conf. Data Mining*, 2007, pp. 583–588.
[24] P. Bodik, M. Goldszmidt, A. Fox, D. B. Woodard, and H. Andersen, "Fingerprinting the datacenter: Automated classification of performance crises," in *Proc. 5th Eur. Conf. Comput. Syst.*, 2010, pp. 111–124.
[25] R. Bitton and A. Shabtai, "A machine learning-based intrusion detection system for securing remote desktop connections to electronic flight bag servers," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1164–1181, May/Jun. 2021.
[26] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," *Pattern Recognit. Lett.*, vol. 24, no. 9, pp. 1641–1650, 2003.
[27] F. Yuan et al., "Insider threat detection with deep neural network," in *Proc. 18th Int. Conf. Computat. Sci.*, 2018, pp. 43–54.
[28] J. Lu and R. K. Wong, "Insider threat detection with long short-term memory," in *Proc. Australas. Comput. Sci. Week Multiconference*,, 2019, pp. 1–10.
[29] M. Villarreal-Vasquez, G. Modelo-Howard, S. Dube, and B. Bhargava, "Hunting for insider threats using LSTM-based anomaly detection," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 451–462, Jan./Feb. 2023.
[30] S. Maleki, S. Maleki, and N. R. Jennings, "Unsupervised anomaly detection with LSTM autoencoders using statistical data-filtering," *Appl. Soft Comput.*, vol. 108, 2021, Art. no. 107443.
[31] R. K. L. Kennedy, Z. Salekshahrezaee, and T. M. Khoshgoftaar, "A novel approach for unsupervised learning of highly-imbalanced data," in *Proc. IEEE 4th Int. Conf. Cogn. Mach. Intell.*, 2022, pp. 52–58.
[32] R. K. Kennedy, Z. Salekshahrezaee, and T. M. Khoshgoftaar, "Unsupervised anomaly detection of class imbalanced cognition data using an iterative cleaning method," in *Proc. IEEE 24th Int. Conf. Inf. Reuse Integration Data Sci.*, 2023, pp. 303–308.
[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[34] T. Zhang, H. Qiu, G. Castellano, M. Rifai, C. S. Chen, and F. Pianese, "System log parsing: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 8, pp. 8596–8614, Aug. 2023.
[35] A. Painsky, "Quality assessment and evaluation criteria in supervised learning," in *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook*. Cham, Switzerland: Springer, 2023, pp. 171–195.

MIR ALI REZAZADEH BAEE (Senior Member, IEEE) received the Ph.D. degree in information security from the Queensland University of Technology, Brisbane, QLD, Australia. He is currently an Information Security Researcher with the Queensland University of Technology, collaborating with the Cyber Security Cooperative Research Center (CSCRC), Australia, to solve pressing real-world cyber security challenges. He has more than 15 years of experience working in computer science, and he has been involved in computer science research with a strong focus on applied cryptography and information security since 2012. His contributions have led to novel and important scientific outcomes. He has actively been a Reviewer for flagship journals such as IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, *Internet of Things*, and conferences including the IACR's *EUROCRYPT and ASIACRYPT*. He is a member of the International Association for Cryptologic Research (IACR).

LEONIE SIMPSON received the Ph.D. degree from the Queensland University of Technology, Brisbane, QLD, Australia, in 2000. She is currently an Associate Professor and Information Security Researcher with the Queensland University of Technology. She is also researching efficient authenticated encryption methods for use in securing data transmissions between small, low-power devices in the rapidly growing Internet of Things. She has been involved in information security research for more than 25 years. She has extensive experience analyzing cryptographic algorithms and finding weaknesses that reduce the security provided. She has applied her knowledge of design flaws in algorithms to help develop more secure ciphers, working in teams with Australian and international researchers. Her research interests mainly include symmetric cryptology, widely used for data protection. She is a member of the International Association for Cryptologic Research (IACR) and the Australian Mathematical Society (AustMS).

WARREN ARMSTRONG received the Ph.D. degree from Australian National University, Canberra ACT, Australia, in 2011. He is currently the Director of Engineering with QuintessenceLabs Pty Ltd., Canberra, Australia. He is the main point of engagement between QuintessenceLabs and Cyber Security Cooperative Research Centre. He has 15 years of industry experience building cybersecurity products.