Interpretable Federated Transformer Log Learning for Threat Forensics

***Abstract*—Threat detection and forensics have become an imperative objective for any digital forensic triage. Supervised approaches have been proposed for inferring system and network anomalies; including anomaly detection contributions using syslogs. Nevertheless, most works downplay the importance of the interpretability of a model's decision-making process. In this research, we are among the first to propose an interpretable federated transformer log learning model for threat detection supporting explainable cyber forensics. The proposed model is generated by training a local transformer-based threat detection model at each client in an organizational unit. Local models learn the system's normal behavior from the syslogs which keep records of execution flows. Subsequently, a federated learning server aggregates the learned model parameters from local models to generate a global federated learning model. Log time-series capturing normal behavior are expected to differ from those possessing cyber threat activity. We demonstrate this difference through a goodness of fit test based on Karl-Pearson's Chi-square statistic. To provide insights on actions triggering this difference, we integrate an attention-based interpretability module.**

We implement and evaluate our proposed model using `HDFS`, a publicly available log dataset, and an in-house collected and publicly-released dataset named `CTDD`, which consists of more than 8 million syslogs representing cloud collaboration services and systems compromised by different classes of cyber threats. Moreover, through different experiments, we demonstrate the log agnostic capability and applicability of our approach on real-world operational settings such as edge computing systems. Our interpretability module manifests significant attention difference between normal and abnormal logs which provide insightful interpretability of the model's decision-making process. Finally, we deem the obtained results as a validation for the appropriate adoption of our approach in achieving threat forensics in the real world.

I. Introduction

The fast-paced digital transformation compelled various sectors, including business and governments, to gradually shift their activities from conventional manual operations to digitally provisioned modalities to cope with the ever increased volume of demands in services. This shift was enabled by the radical advancement of communication technology, online software services, and increased bandwidth services. Despite the flexible benefits, this revised modality introduces new cyber security threats to computing systems and infrastructures used by corporations and individuals [49].

Most recently, cyber attacks targeting healthcare organizations increased by 45% and by 22% across all other industries around the world [1]. Cyber attacks such as the one launched against Britain's National Health Service [12], Brno University Hospital (Czech Republic) [7], and the University of Düsseldorf (UKD) (Germany) demonstrate the range of employed attack vectors including ransomware, Distributed Denial of Service (DDoS), botnets, and other malicious misdemeanor. In response to these attacks, a joint cyber security advisory was issued by the Cyber Security and Infrastructure Security Agency (CISA), the Federal Bureau of Investigation (FBI), and the Department of Health and Human Services (HHS) advising on an increased and imminent cybercrime threat towards U.S. critical infrastructure and other assets [2].

That said, one important aspect of threat forensics is to acquire attack provenance. For instance, a ransomware attack on a particular organizational machine provokes system behavior changes in which the attack openly notifies the user of the infection [22]. By employing advanced deep learning-based threat detection models on system logs (syslogs), cyber security analysts can better differentiate normal and abnormal system behaviors to develop anomaly detection [14, 33, 47]. Indeed, syslog analysis is highly necessary to understand the inner working and behavior differences of an Operating System (OS). Such logs offer valuable information about significant occurrences that explain how the system operates in terms of software, hardware, system processes, and system components which all can be leveraged to detect a plethora of abnormal activity occurring within a system.

Nevertheless, syslogs contain sensitive information hidden from outside scans and may disclose information about vulnerable targets, login credentials, and security associations between entities [41]. Therefore, exposing syslogs containing sensitive information compromises data privacy. To address this liability, various anonymization techniques such as generalization and suppression have been employed by the industry to safeguard the user's data privacy and system security. A drawback of applying such techniques is the decreased data usefulness for analysis and understanding the system's behavior. Maintaining a balance between data usefulness and privacy preservation remains an important challenge when dealing with syslogs.

In addition, most proposed works fail to include imperative elements for the development of effective threat detection models. These include: (1) producing, collecting, and open-sourcing novel representative datasets, (2) adopting training techniques to ensure user data privacy in accordance with jurisdiction [43] [28], and (3) provide visibility to the model's decision making process for identifying system activity triggering post-incident detection.

To address this gap, we propose the first interpretable federated transformer log learning model for threat forensics. Unlike other works that exclusively focus on anomaly detection in syslogs, our proposed model incorporates the concept of

Federated Learning (FL), a machine learning setting where multiple entities collaborate in solving a machine learning problem under the coordination of a federated server. We leverage the distributed nature of this concept to generate a robust model that offers a privacy preserving solution with a high level of security [4].

The proposed model is segmented into two main stages. In the first stage, a local transformer-based model is trained at each client using the system's local dataset composed of syslog files. Log event sequences are mapped to log key sequences using a parser. The mapped time-series are then embedded and fed to a local model using stochastic gradient descent (SGD) for learning the underlying patterns of each given sequence. In the second stage, the learned parameters from all local models are passed to the FL server, then aggregated to generate a global federated learning model. The global FL model is then shared with all participating clients. Multiple rounds of this cycle are executed for improving the performance of threat detection. In the last cycle, the interpretability module at each client computes the interpretability weights using the latest global FL model. Clients then share the weights with the FL server who aggregates them to generate the federated interpretability weights. All clients receive an updated version of the global FL model for inference and the federated interpretability weights used for comparing it's attention distribution against the one from a specific log sample. In addition, the intrepretability module provides forensic investigators the means to backtrack the log keys triggering a cyber threat detection. To the best of our knowledge, our work is the first to offer this benefit for cyber forensics and digital triage in a federated setting.

In summary, this work makes the following contributions:

- We designed and implemented the first interpretable federated transformer log learning model for cyber threat detection in syslog with the capability of revealing actionable information triggering the model's outcome.
- We validated our model's performance by training it using two datasets while comparing it against State-Of-The-Art (SOTA) works.
- We discovered a strong correlation between log messages carrying indicators of the executed threat sample and the attention given to the corresponding mapped input sequence triggering the threat detection.
- We generated a cyber threat detection dataset consisting of 8,448,715 syslogs collected during the second half of 2020 from a production-level environment comprised of 62 instances running uncompromised cloud collaboration services and 16 cloud instances running threat samples.

The rest of this paper is organized as follows: In Section II, we present the background and related work. Section III covers our proposed model design. Section IV presents our experimental evaluation. Section V explores the capabilities of the model's interpretability module. We present our discussion in Section VI. Finally in Section VII, we offer concluding remarks and opportunities for a few future endeavors.

II. Background and Related Work

In this section, we introduce existing approaches adopted in syslog analysis and their corresponding limitations. In addition, we discuss the concept of federated learning and attention-based interpretability in the field of cyber security. Furthermore, we present a brief overview of the datasets used for threat and anomaly detection.

A. Existing Approaches for Syslog Analysis

During routine system operations, syslogs are generated from various sources in a computer system and consist of textual messages describing the activity throughout the global system. Such messages provide an abundance of information linked to the system's behavior. That said, threat forensics rely on periodic syslog scrutiny to unveil abnormal system behavior originating from suspected attacks. Due to the large number of generated logs, the examination of syslogs is not a trivial task and can be hard (if not impossible) to manually track, analyze, detect, and diagnose system problems. In the following section, we highlight previous and emerging log-based analysis approaches adopted in the literature and discuss their limitations along with the newly adopted approaches.

Rule-based approaches: Early research on log-based analysis proposed rule-based detection techniques. Such approaches rely on predefined rule-sets or dynamic rules which can be created, superseded, or deleted by existing rules at run time to accommodate for system behavior changes and thus limit and avoid unnecessary alert reporting. More precisely, rule-based approaches employ regular expressions to parse and recognize events and trigger predefined actions based on matched rules [17, 39, 40]. For instance, Swatch [17] is a monitoring log file system that filters unwanted data and takes predefined actions based on detected log patterns. Swatch can ignore duplicate entries and perform rule changes based on the time of arrival. Moreover, Logsurfer [39], a log file analysis software, is designed to detect signatures of complex interactions and is capable of updating its rule-set at run-time and act accordingly.

However, two main foreseen pitfalls of rule-based approaches are the need for continuous rule-sets maintenance and their significant specificity to application scenarios. Such ongoing maintenance requires domain expertise to define rules that describe system behaviors. Nonetheless, possible bottlenecks could occur during expert involvement in defining the rules. Accordingly, the literature has taken new directions and proposed further approaches to perform log analysis, which we discuss in the sequel.

Causality-based approaches: Causality analysis on logs gained attention for comprehending system activities and detecting potential risks. These approaches leverage backward and forward causal graphs to identify multi-hop attacks [26]. In addition, causality analysis utilizes dependency graphs [25] and action history graphs [24], which provide a detailed graph describing the system execution sequences that occur during an intrusion. Moreover, provenance-aware system approaches are introduced in [36] and [38] to facilitate the integration of provenance across multiple levels of abstraction. For example, such approaches can help determine malware existence and find the source of anomalies. Moreover, whole system simulation, like TaintBochs [11, 16], is being used to analyze sensitive data handling at a holistic level while helping recover legitimate file system data after an attack.

Despite the many credits of causality analysis in time series problems, many concerns were related to the non-trivial lifetime and iterative input and output processing, which can cause the well-known problem of dependency explosion [29]. Many researchers addressed the problem of dependency explosion by adopting data reduction techniques [16, 26, 29]. These approaches include binary-based execution partitioning and TaintBochs simulation. However, it is not certain that reducing the data volume of irrelevant dependencies can lead to a decrease in attack investigation time [32]. In addition, such data reduction [5] can unintentionally eliminate important data and hinder the efficacy of the analysis.

AI-based approaches: Current literature presents two main approaches for log analysis that make use of AI models namely, (1) log event indices-based approaches and (2) log template semantics-based approaches. In this paper, we follow the log event indices-based approach. Both approaches first employ a log parser to identify log templates from a time-series of log events. Each log templates is assigned a unique numerical identifier (whole number) called log key which substitutes each matching log from the original time-series. In contrast, semantics-based approaches employ word embeddings to convert the identified log templates into vectors.The sequence of vectors representing the original time-seriesis then used for training supervised and unsupervised AI-based models. These models learn the underlying patterns within the vector time-series using different methods. Such methods include Principal Component Analysis (PCA) [51], Support Vector Machine (SVM) [18], Bi-LSTM [54], LSTM [14, 35], and Transformers [20, 48]. These works incorporate methods to control previously unseen log events during the model's training phase via human intervention or by applying semantic vectorization techniques such as Term Frequency-Inverse Document Frequency (TF-IDF).

Nonetheless, SOTA methods fail to preserve user data privacy, as they are based on centralized training approaches, and lack methods for revealing the level of influence that each element in the time-series had over the model's prediction.

B. Federated Learning and Attention-based Interpretability

Federated learning [34] addresses the fundamental problems of privacy, ownership, and data locality. Federated Stochastic Gradient Descent (FedSGD) and Federated averaging (FedAVG) [34] are two methods used for aggregating the parameters of local models trained at each client. Previous works on federated learning has been applied in the cyber security domain. These include (1) a multi-task deep neural network in federated learning (MT-DNN-FL) for network anomaly detection, traffic recognition, and traffic classification [55], (2) an autonomous self-learning distributed system for detecting anomalies in IoT devices [37], and (3) DeepFeed [30], which applies federated deep learning to detect cyber threats against industrial cyber physical systems. To the best of our knowledge, no works in the cyber forensic space have integrated federated learning and interpretability for threat inference using syslogs.

Model interpretability in cyber forensic applications has been explored using LSTM-based models [6] by calculating the weighted sum over the attention value vectors. Other LSTM works, in which no attention mechanisms are used, have employed anomaly score decomposition [45] to provide visibility on the model's decision making process. Such works do not integrate interpretability in a federated learning setting.

C. Datasets for Threat and Anomaly Detection

Most cyber security datasets focus on network traffic [15, 23] while the few public datasets which consist of syslogs [52] are mostly collected from distributed systems [50] or high performance computing infrastructure with a relatively low number of unique log messages and data structure, in contrast to those seen in OS logs. On the other hand, datasets for cyber threat analysis have been created using synthetic data generators [31] or using real systems but are limited to authentication events in Windows-based systems [21].

Different from these works, our Cyber Threat Detection Dataset (`CTDD`) [3] is the first publicly available dataset consisting of syslogs produced by real user activity and synthetic threat scenarios in a modern cloud-based operational system. The syslogs were captured from 62 Virtual Machines (VMs) running uncompromised cloud collaboration services. Synthetic threat scenarios were simulated in 16 additional VMs. Our approach facilitate the incorporation of additional real or synthetic threat samples using the same cloud image which is publicly available in Jetstream Cloud [42, 44]. Our framework's environment deployment was secured by enabling IP-based access for ingress traffic, restricted port access, and log data sanitization throughout our pipeline to ensure a negligible uncertainty in the collected logs from uncompromised systems presented in the `CTDD` dataset.

III. Proposed Model Design

In this section, we present the proposed interpretable federated transformer log learning model supporting threat forensics. The proposed model is designed under the following assumptions: (1) federated servers and clients are trusted; (2) The FL model is implemented without additional privacy protection or security mechanisms beyond preserving data locally at each client and the inherited privacy benefits of FL; (3) only offline learning is considered for training the proposed model with homogeneous Independent and Identically Distributed (IID) data; (4) distributed optimization follows a synchronous approach in which all clients are available at all times; and (5) a typical horizontal FL architecture [53] is followed in which local datasets present different samples but share the same feature space. The training process of our interpretable federated transformer log learning model for threat forensics is presented in Algorithm 1.

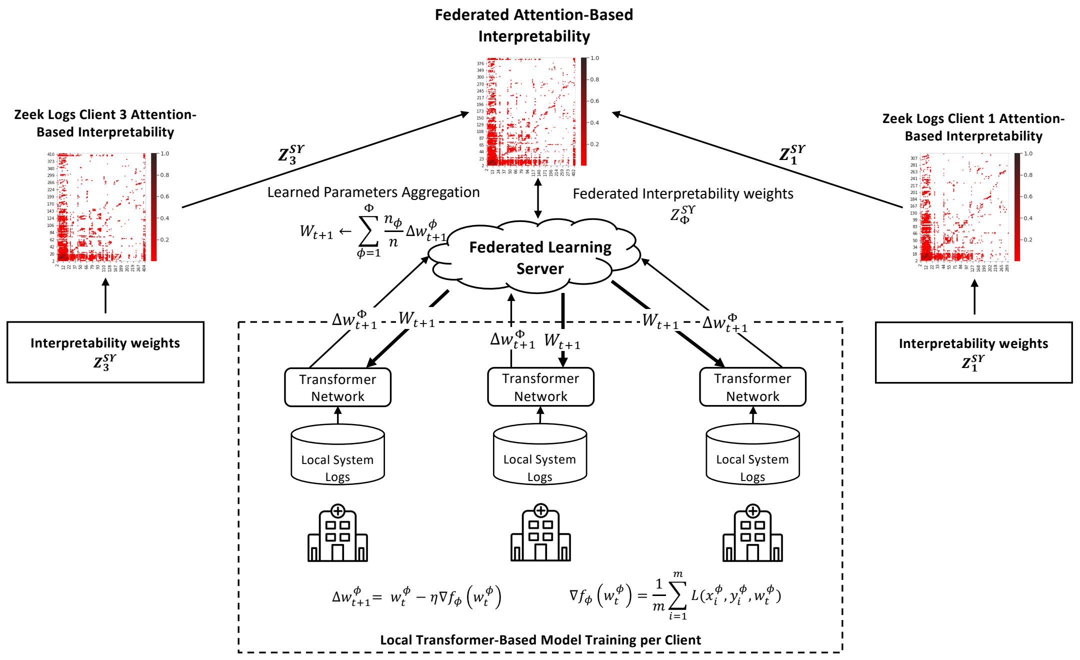

Federated Attention-Based Interpretability
Zeek Logs Client 3 Attention-Based Interpretability
Zeek Logs Client 1 Attention-Based Interpretability
Z_3^{SY}
Z_1^{SY}
Learned Parameters Aggregation
$W_{t+1} \leftarrow \sum_{\phi=1}^{\Phi} \frac{n_\phi}{n} \Delta w_{t+1}^{\phi}$
Federated Interpretability weights Z_Φ^{SY}
Federated Learning Server
Interpretability weights Z_3^{SY}
Interpretability weights Z_1^{SY}
Δw_{t+1}^{Φ} W_{t+1}
Transformer Network
Local System Logs
$\Delta w_{t+1}^{\phi} = w_t^{\phi} - \eta \nabla f_\phi (w_t^\phi)$
$\nabla f_\phi (w_t^\phi) = \frac{1}{m} \sum_{i=1}^{m} L(x_i^\phi, y_i^\phi, w_t^\phi)$
Local Transformer-Based Model Training per Client

Fig. 1. The architecture of the proposed Interpretable Federated Transformer Log Learning for Threat Detection. A set of clients Φ contributing to the global FL model receive a set of hyperparameters and initial model parameters W_0 for training local transformer-based models using their local data. Additionally, the interpretability module uses the model's calculated attention by key to compute $Z_\phi^{S\Upsilon}$. $Z_\phi^{S\Upsilon}$ for normal and cyber threat case scenarios. Each client sends the learned model parameters Δw_{t+1}^{ϕ} and their computed interpretability weights $Z_\phi^{S\Upsilon}$ to the FL server. The FL server aggregates the learned parameters to generate an updated global FL model W_{t+1}. Afterwards, the interpretability module at the FL server aggregates $Z_\phi^{S\Upsilon}$ to generate the federated interpretability weights $Z_\Phi^{S\Upsilon}$. Finally the FL server shares W_{t+1} and $Z_\Phi^{S\Upsilon}$

The starting point of our proposed model's training process, presented in Fig. 1, takes place at client devices from each organization unit producing log files (i.e., auth.log, syslog, kernel, and audit.log). Log messages are processed using a log parser that maps the time-series of log messages into a sequence of log keys. These sequences are embedded into vectors used for training a local model at each client that learns the underlying patterns from the log key sequence. Each client passes the local model's learned parameters to the FL server. The FL server aggregates the parameters from the participating clients to generate the global FL model which is shared back to the clients. Finally, an interpretability module is offered to provide insights to the model's decision making process by aggregating the calculated attention given to each unique element in the input sequence. Details of each component are described in the following.

A. Distributed Federated Learning Architecture

In Algorithm 1, we integrate the FedAvg algorithm presented by [34]. More specifically, we leverage this algorithm on the federated learning server depicted in Fig. 1. In this process, the FL server collaborates with Φ clients $c_1, \ldots, c_{\Phi-1}, c_\Phi$ for generating an updated global FL model. Each client trains a local model using local data and shares only the locally computed gradients or learned model parameters $\Delta w_{t+1}^{1}, \ldots, \Delta w_{t+1}^{\Phi-1}, \Delta w_{t+1}^{\Phi}$ with the FL server.

Initial hyperparameters and model parameters W_0 are passed by the FL server to each client. FederatedSGD (FedSGD) is an algorithm used for generating a global model in a federated setting. In this process, the client computes the average gradient $g_\phi = \nabla f_\phi(w_t^\phi)$ using its local data at the current model w_t^ϕ, as seen in Algorithm 1−I. Then, clients pass g_ϕ to the FL server that aggregates them $\nabla F(W_t) = \sum_{\phi=1}^{\Phi} (\frac{n_\phi}{n}) g_\phi$. The aggregated average gradients are used for computing an updated global FL model $W_{t+1} \leftarrow W_t - \eta \nabla F(W_t)$ where n_ϕ denotes the number of data points for client ϕ. FedAvg is an equivalent algorithm used for updating the global FL model. In this approach, the learned model parameters from each client are aggregated by the FL server. Given that $\forall \phi, \Delta w_{t+1}^{\phi} \leftarrow w_t^{\phi} - \eta g_\phi$, the updated global FL model can be computed as $W_{t+1} \leftarrow \sum_{\phi=1}^{\Phi} (\frac{n_\phi}{n}) \Delta w_{t+1}^{\phi}$, as seen in Algorithm 1−II. Afterwards, the updated global FL model is sent to all clients.

B. Log Parser

Our decentralized federated transformer log learning model requires vector representation for the time-series of log messages to learn the extensive and convoluted patterns and correlations embedded within log sequences. `Spell`, a public log parser [13, 19], is used for identifying log templates within

Algorithm 1: Interpretable Federated Transformer Log Learning Algorithm.

```
Input: Hyperparameters for local models defined by
       the federated server, number of clients $\Phi$, and
       number of rounds $R$.
Output: Updated Federated Transformer Log Learning
        Model $W_{t+1}$ and Interpretability weights $Z_{\Phi}^{S}$
        and $Z_{\Phi}^{S\Upsilon}$.
Initialization:
1 FL server defines the hyperparameters for the federated
   model including encoder/decoder layers $N$, attention
   heads $H$, and clients $\Phi$.
Procedure:
2 while $Round \neq 0$ do
      (I). For clients in business unit:
      for all $\phi \in \Phi$ do
          Local Transformer-based Model Training:
          Log parser maps all log messages $M$ to log
          sequences $S_j$:
          $S_j \leftarrow M$
          Log key sequences $S_j$ are embedded:
          $X_j \leftarrow S_j$
          Compute the average gradients with current
          model $w_t^{\phi}$:
          $g_{\phi} = \nabla f_{\phi}(w_t^{\phi})$
          Update the model parameters:
          $\Delta w_{t+1}^{\phi} \leftarrow w_t^{\phi} - \eta g_{\phi}$
          Send the learned model parameters $\Delta w_{t+1}^{\phi}$ to
          the Federated Learning (FL) server.
          Interpretability Module:
          Compute interpretability weights by $Z_{\phi}^{S}$ and
          $Z_{\phi}^{S\Upsilon}$ and share them with the FL server.
      end
      (II). For Federated Learning Server:
      Global Model Update:
      $W_{t+1} \leftarrow \sum_{\phi=1}^{\Phi}(\frac{n_{\phi}}{n})\Delta w_{t+1}^{\phi}$
      Interpretability Module:
      Generate the federated interpretability weights:
      $\mathrm{Z}_{\phi}^{S} = \sum_{h=1}^{H} Z_{h,\kappa}$
      $\mathrm{Z}_{\phi}^{S\Upsilon} = \sum_{h=1}^{H} Z_{h,\kappa,\Upsilon_v}$
      Share $Z_{\Phi}^{S\Upsilon}$ with clients.
      (III). For clients in business unit:
      for all $\phi \in \Phi$ do
          Update local model:
          $w_t^{\phi} = W_{t+1} \leftarrow \text{FederatedServer}(\phi, W_{t+1})$
          Compute goodness of fit test for $Z_{\phi}^{S\Upsilon}$:
          $D_i = \sum_{i=1}^{v}\sum_{k=1}^{K} \frac{Z_{\phi}^{S\Upsilon_i} - Z_{\Phi}^{S\Upsilon_i}}{Z_{\Phi}^{S\Upsilon_i}}$
          Update interpretability weights:
          $Z_{\phi}^{S\Upsilon} \leftarrow Z_{\Phi}^{S\Upsilon}$
      end
3 end
```

a log time-series. Given a subset of log files l where $l \in L$, each log file (i.e., audit.log, auth.log, kernel.log, sys.log) is composed of a finite unstructured sequence of log messages $l = \{m_i : m_i \in M, i = 1, 2, \ldots\}$ where m_i denotes the message at index position i, and M denotes all the log messages in l.

Mapping log messages to log keys requires the log parser to first identify the event template and variable elements from each log message. The mathematical representation of the mapping function is $\beta : m_i \rightarrow (e_i, v_i) \forall i, e_i \in E, i = 1, 2, \ldots$ where $E = e_1, e_2, \ldots, e_n$ represents all the identified distinct event templates in the log file l, e_i represents the event template identified in m_i, and v_i is the corresponding list of variables. Each distinct template in E is assigned a unique log key κ_i using the mapping function $\psi : e_i \rightarrow \kappa_i \forall i, e_i \in E, \kappa_i \in K, i = 1, \ldots, n$, where K represents the set of all unique key values. The time-series of the mapped log keys S is presented in the formula below:

$$\begin{aligned} S = \{\kappa_1, \kappa_2, \ldots, \kappa_n\} \leftarrow \psi(e_i) \leftarrow e_i \leftarrow \beta(m_i) \forall i, \\ m_i \in M, e_i \in E, \kappa_i \in K \end{aligned} \tag{1}$$

After all parsing operations are finished, S is segmented into sub-sequences S_j based on defined window frames Λ as presented in the following equations:

$$\begin{aligned} S_j &= ((\kappa_{(j-1)\Lambda+p})_{p=1,\ldots,\Lambda})_{j=1,\ldots,n/\Lambda} \\ &= ((\kappa_{(j-1)\Lambda+p})_{p=1}^{\Lambda})_{j=1}^{n/\Lambda} \\ &= \{S_1, \ldots, S_{n/\Lambda}\} \end{aligned} \tag{2}$$

C. Local Log Learning: Transformer-Based Model

Our proposed model integrates the work presented by Vaswani et al. in [46]. Local models are composed of a set of stacked encoder modules, a set of stacked decoder modules, and the interconnections between them.

Encoder: The encoder is composed of a multi-head attention and a feed-forward sub-layer. A residual connection is employed for each sub-layer followed by layer-normalization. The elements in log key sequence are embedded to form a vector list $X = (x_1, x_2, \ldots, x_n)$ where the vector size is defined as d_m. A positional encoding vector is added to each input embedding to provide context of their corresponding position in the sequence. The resulting vector list is passed to the self-attention layer.

Self-attention, used to define the relationships between every key and the other elements in the sequence, is calculated using 3 vectors, namely, query(Q), key(K), and value(V) generated by multiplying every vector x_i with 3 matrices W^Q, W^K, W^V. The resulting vectors Q, K, V have the corresponding dimensions d_q, d_k, d_v; all being smaller than d_m. The output of the self-attention layer Z_s is calculated as follows:

$$Z_s = Attention(Q, K, V) = softmax(Q \cdot K^T / \sqrt{d_k})V \tag{3}$$

To improve the performance of the attention layer, we implement the multi-headed attention mechanism which linearly projects the Q, K, and V matrices H times, where H represents the number of attention heads to be used. This approach allows the model to jointly address information from log keys from different representation sub-spaces at different positions in the sequence. The resulting process is a multi-head $Z_{cn} = Concat(Z_1, Z_2, ..., Z_H)$ matrix generated by the concatenator of individual Z_h matrices resulting from each attention-head. The Z_{cn} matrix is then multiplied with a weight matrix W^O, trained jointly with the model, resulting in a matrix Z that captures the information from all attention heads. The output matrix Z is finally passed to the feed-forward sub-layer and then to the input of the next encoder in the stack.

$$\begin{aligned} Z &= Multi-head(Q,K,V) \\ &= Concat(Z_1, Z_2, ..., Z_H)W^O \\ where & \\ Z_h &= Attention(QW_h^Q, KW_h^K, VW_h^V), \\ & h = 1, \ldots, H \end{aligned} \tag{4}$$

Decoder: Similar to the encoder module, the decoder module is composed of a stack of decoders. The decoder shares the same components of the encoder with the addition of an encoder-decoder attention sub-layer positioned between the self-attention and feed-forward sub-layers. Matrices K and V result from the encoder's module output. The Q matrix is shared by the previous decoder in the stack. These three matrices allow every position in the decoder to attend all positions in the log key input sequence. In addition, the auto-regressive decoder module adds the output of each step, the predicted log key, and a positional encoding. In contrast with the encoder, the self-attention layer of the decoder is only permitted to attend earlier positions in the output sequence; this is achieved by masking future positions before the softmax calculation takes place.

After the multi-head attention, each encoder and decoder has a pointwise feed-forward layer. This is applied separately and identically to each element in the log key sequence. The feed-forward layer consists of two linear transformation which use ReLu activation function and is formally expressed as:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{5}$$

The logits vector is ingested by a softmax layer that transforms the logits vector into probabilities. The element in the vector with the highest probability is selected and the associated log key becomes the output of this specific time step.

The training data consists of sequences of mapped keys from syslogs collected from uncompromised systems. Sequences of log keys within a window frames Λ are used for training the local model. The local model is trained to predict a list of candidate log keys given previous log keys. The transformer neural network architecture creates a multi-classification model, each type of log key representing a class. As stated before, this SOTA architecture uses attention to gain context from previous inputs when predicting the next log key. With structured data and a built model, the log key anomaly detection model is trained to predict the next log key, representing the next log entry. The generated prediction is fed to the model to continue generating predictions.

After training the model, new logs are processed to generate a new prediction of log keys. First, a log key sequence, based on a window frame Λ, is fed to the model. Based on this input, the model predicts the top g candidates. If the ground truth key G_j (which follows the input sub-sequence S_j) matches one of the top g candidates, then S_j is classified as a normal sequence. Otherwise, the input sub-sequence S_j is classified as abnormal. A sequence S is classified as a cyber threat if the number of sub-sequences classified as abnormal is greater than or equal to a threshold τ defined by the user.

In order to backtrace the source of detected anomalies, the interpretability module presents the sequence of log keys that triggered the detection of an anomaly and the attention for each key in the input sequence. If a log key is classified as abnormal, the sequence of log keys that initiated the incident can be tracked down. The sequence that triggered that anomaly is considered a risk pattern. Risk patterns are then extracted from the sequence to generate new test cases.

D. Model Interpretability

The interpretability module at each client provides insight to the level of influence each unique log key $\kappa \in K$ has in a particular sequence of log keys S. The federated transformer log learning model $W_t + 1$ is used by the interpretability module to evaluate a given sequence S. While the federated transformer log learning model is trained using log key sequences S from uncompromised systems, the interpretability module at each client and at the FL server can be used for log key sequences collected from compromised or uncompromised systems.

Initially, each client uses their currently available data from their uncompromised systems. As the federated model ingests each sub-sequence $S_j \in S$, it calculates the attention for each key $k_i \in S_j$. After processing all sub-sequences $S_j \in S$, the interpretability module aggregates the attention of each unique log key $\kappa \in K$ across all S_j. We refer to this influence as the interpretability weights. This operation is reflected in the following equation:

$$Z_\phi^S = \sum_{h=1}^{H} Z_{h,\kappa}, \forall \kappa \in K, \forall S_j \in S \tag{6}$$

Furthermore, the interpretability module provides additional granularity by aggregating the attention of every unique key $\kappa \in K$ across all input sequences S_j for each ground truth key G_j. G_j is also an element of Υ which denotes the list of unique ground truth keys $|G_j|$ from all clients. We denote υ as the index for each unique key in Υ. The weight calculated by aggregating the attention of every unique key κ for all sequences S_j influencing each Υ_υ is denoted by the following equation:

$$Z_{\phi}^{S\Upsilon} = \sum_{h=1}^{H} Z_{h,\kappa,\Upsilon_v}, \forall \kappa \in K, \forall S_j \in S, \forall \Upsilon_v \in \Upsilon \quad (7)$$

Each client sends the interpretability weights Z_{ϕ}^{S} and $Z_{\phi}^{S\Upsilon}$ to the FL server. The FL server aggregates the computed weights from each client ϕ. Afterwards, the FL server shares the federated interpretability weights Z_{Φ}^{S} and $Z_{\Phi}^{S\Upsilon}$ with each client. The same process is followed for logs from compromised systems. As an example, multiple users can send to the FL server the Z_{ϕ}^{S} and $Z_{\phi}^{S\Upsilon}$ computed from log sequences S from compromised systems to generate federated interpretability weights for cyber threats Z_{Φ}^{S} and $Z_{\Phi}^{S\Upsilon}$.

After the client receives Z_{Φ}^{S} and $Z_{\Phi}^{S\Upsilon}$ for uncompromised and/or compromised systems from the FL server, the interpretability module conducts a multinomial distribution goodness of fit test on compromised and/or uncompromised systems based on chi-square for $Z_{\phi}^{S\Upsilon}$ with respect to $Z_{\Phi}^{S\Upsilon}$ to quantify their statistical distribution difference. The interpretability module tests if the interpretability weights for a sequence of interest $Z_{\phi}^{S\Upsilon}$ has a specific distribution by computing the Karl Pearson's chi-square statistic as follows:

$$D_i = \sum_{i=1}^{v} \sum_{k=1}^{K} \frac{\left(Z_{\phi}^{S\Upsilon_i} - Z_{\Phi}^{S\Upsilon_i} \right)^2}{Z_{\Phi}^{S\Upsilon_i}}, \forall v \in \Upsilon, \forall \kappa \in K \quad (8)$$

Using the multinomial, we can then test if the given sample $Z_{\phi}^{S\Upsilon}$ has a similar distribution to $Z_{\Phi}^{S\Upsilon}$ by testing:

$$\begin{aligned} &H_0 : Z_{\phi}^{S\Upsilon} = Z_{\phi}^{S\Upsilon_1}, \ldots, Z_{\phi}^{S\Upsilon_v} = Z_{\Phi}^{S\Upsilon} \quad VS \\ &H_1 : Z_{\phi}^{S\Upsilon} \neq Z_{\Phi}^{S\Upsilon} \end{aligned} \quad (9)$$

Given an observation $\left(Z_{\phi}^{S\Upsilon} = Z_{\phi}^{S\Upsilon_1} \ldots Z_{\phi}^{S\Upsilon_v} \right)$, the valid p-values are calculated:

$$p - value = P\left(\chi^2(v-1) > \sum_{i=1}^{v} \sum_{k=1}^{K} \frac{\left(Z_{\phi}^{S\Upsilon_i} - Z_{\Phi}^{S\Upsilon_i} \right)^2}{Z_{\Phi}^{S\Upsilon_i}} \right) \quad (10)$$

The interpretability module uses the p-value to fail to reject the hypothesis H_0 or reject the hypothesis H_1 that the distribution of the interpretability weights for the given sample $Z_{\phi}^{S\Upsilon_v}$ shares the same distribution as $Z_{\Phi}^{S\Upsilon_v}$. The test rejects the null hypothesis because the cyber threat induces a perturbation in the space from the joint distribution point of view, making the test statistics able to account for the difference between both cases as Chacon et al. [10] [9] exhibited in his research.

IV. Experimental Evaluation

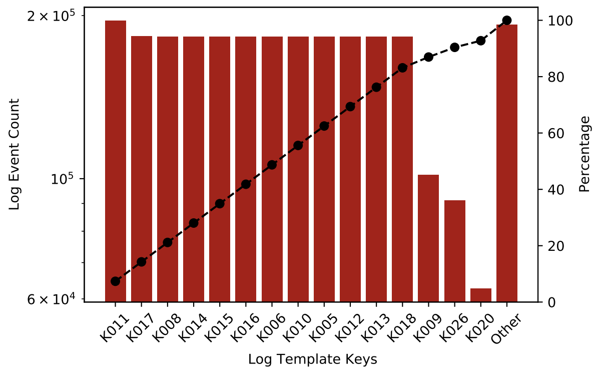
A. Datasets Considered

Our experiments were conducted using the two datasets summarized in Table I. The HDFS dataset [50] is composed of 11,175,629 logs collected from a cluster of 200 Amazon virtual machines running Hadoop-based jobs that can easily be obtained from Zenodo [8]. The original work labeled all logs in this dataset using handcrafted rules to classify them between normal and abnormal. Abnormal samples in this dataset are identified by matching a set of block ids listed as abnormal to their appearance in the log messages. The list of abnormal logs amount to 16,838 blocks ids.

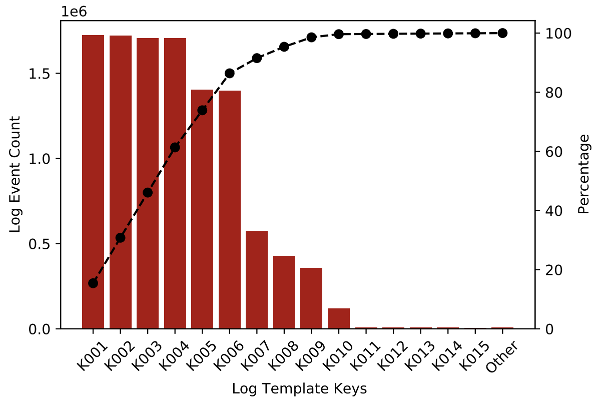
Our CTDD dataset is composed of logs registering system activity running uncompromised cloud collaboration services. This environment consists of a cluster of virtual machines deployed across multiple networks in the Jetstream educational cloud that offer interactive computing in the cloud such as Jupyter Lab. The normal operation syslog samples presented in this dataset were collected from 62 VMs, each operated by different users. No user was allowed to access a VM assigned

TABLE I
Statistics of HDFS and CTDD datasets

Datasets	Duration	# of logs	# of Anomalies
HDFS	38.7 hours	11,175,629	16,838 (blocks)
CTDD	235 days	8,448,715	2,501 (logs)



(a) CTDD Log Keys Pareto, showing top 15 log template keys and the remaining 431 log template keys classified as "Other"



(b) HDFS Log Keys Pareto, showing top 15 log template keys and the remaining 35 log template keys classified as "Other"

Fig. 2. Distribution of Log Key Templates by Dataset CTDD vs. HDFS

TABLE II
MALICIOUS SAMPLES OF CYBER THREAT DETECTION DATASET

Attack Case	Description of Scenario	References	# of logs
GonnaCry	GonnaCry is an academic ransomware program which allows users to infect a client by encrypting files, peripheral devices, and destroy original files. It does not have all the features of WannaCry2.0	S0366	129
ech0raix	A ransomware family that targets QNAP Network Attached Storage (NAS) devices. Devices are compromised by bruteforce attacks or by exploiting known vulnerabilities. The ransomware executes a malicious payload that encrypts targeted file extensions on the NAS.	T1486	27
ACK Flood	An attack that exhausts OS finite TCP connections by sending a flood of ACK packets for nonexistent connections while leveraging the stateful nature of the TCP protocol.	T1499.001	114
NTP DDoS Amplification	A DoS Reflection Amplification Attack. The attack sends packets to a third-party server with a spoofed source IP address.	T1498	406
SYN Flood	An attack that exhausts OS finite TCP connections by sending large quantities of SYN packets where the 3-way TCP handshake is never completed	T1499.001	132
malaria	An injection attack that injects malicious code via trace system calls. The trace system call injection is usually executed by writing arbitrary code into a running process.	T1055-008	182
nemox	A half-virus for infecting any ELF files in a specific directory.	T1027-001	109
nf3ct0r	A virus for infecting ELF files. Also know as an ELF infector.	T1027-001	135
utrojan	Universal Trojan for accessing an unauthorized system.	T1036-004	221
Lin Blackhole	A malicious program created using C programming language which upon infecting the client provides a backdoor to the attacker.	T1587-001 and T1588-001	74
Lin Ovason	A malicious program created using C programming language which upon infecting the client provides a password protected backdoor to the attacker.	T1587-001 and T1588-001	101
Python Backdoor	A malicious program written using Python programming language infects the client to serve a backdoor to the attacker. The attacker can obtain a simple reverse shell using tools like netcat or socat	T1587-001 and T1588-001	179
Binom ASM	A computer virus that is written using assembly language searches for ELF files in the victim client and injects malicious payload. This computer virus requires admin-level access for successful infection. It targets the files located in the bin directory in the victim's machine	T1027-001	292
Eternity ASM	A computer virus that is written using assembly language that infects a target ELF file in the victim's machine.	T1027-001	152
Dataseg Code Injector	A malware that injects unwanted/malicious code into the data segment of the binaries mainly for defense evasion purposes.	T1027-001	202
Bash Spyware	A simple bash script that uses built-in tools for harvesting internal system data and sends it to a compromised mail server.	T1119	46
Ransomware Attack	A low footprint ransomware attack based on Data Encryption Standard (DES) that target Linux-based systems.	T1486	45

to another user. The threat samples from our dataset were collected from 16 VMs running malicious software samples.

Syslogs from uncompromised VMs running cloud collaboration services were collected from 3 clusters, all of them hosting Ubuntu 18.04 operating systems. Each VM was assigned to a student to perform a variety of data analytics and machine learning activities. "Sudo" privileges were restricted for "Ubuntu" user in the Practicum 2020 cluster while the IS 7033 and ITESM 2020 clusters had unrestricted "root" privileges. Furthermore, VMs in the IS 7033 clusters were safeguarded by a firewall that allowed access exclusively to each user's IP addresses while the Practicum 2020 and ITESM 2020 clusters were assigned to security groups that restricted ingress traffic to most network ports. In addition, a total of 2,501 syslogs were collected from 16 compromised VMs running malicious threat samples presented in Table II. The compromised VMs were deployed using the same cloud image used in the Practicum 2020 environment.

Compared to `HDFS`, our `CTDD` dataset presents a greater variety of log templates and a more complex relationship between log sequences in the time-series. This observation was validated by manually generating strict rules to parse each log in the `HDFS` dataset were we discovered a total of 50 different unique log templates. We validated that the only elements that varied in each template were the parameter values such as block IDs, IPs, port numbers, etc. In the pareto presented in Fig. 2 we can further observe that the top 10 templates with the highest count in the `HDFS` dataset match 99.63% of the data points. We performed a similar study with the logs obtained from the Practicum 2020 cluster and malicious environments that form part of our dataset. Different from the strict rule generation applied in the `HDFS` dataset, we identified 446 unique log templates using the spell parser. It is important to note that this data subset only represents 31.47% of the `CTDD` dataset where the top 10 log templates with the highest count match 69.37% of our data points.

In our experimentation, we make use of the $2,657,112$ syslogs collected from the Practicum 2020 environment and the $2,501$ syslogs collected from the 16 malicious environments presented in Table II. These logs come from a much larger set of syslog files that included information that were simultaneously collected in separate log files (e.g., kernel and collection agents logs), large error logs produced by Jupyter notebook, and malformed logs which were removed.

TABLE III

Experimental results using the HDFS and CTDD datasets. Hyperparameters used for this study include: $N = \{1, 4, 6\}$ encoder/decoder layers, $H = \{1, 2\}$ attention heads, and $\Phi = \{1, 2, 4, 8, 10\}$ clients. Our model's top performance on the HDFS dataset was achieved with 10 clients, 4 encoder layers, 4 decoder layers, and 1 attention head. On our CTDD dataset, the model's top performance was achieved with 10 clients, 2 attention heads, and with 4 and 6 encoder and decoder layers.

Φ	N	H	HDFS				CTDD			
			Accuracy	**Precision**	**Recall**	**F-score**	**Accuracy**	**Precision**	**Recall**	**F-score**
1	1	1	83.986	92.524	70.240	79.857	**79.394**	**77.333**	**77.333**	**77.333**
2			90.508	84.219	97.211	90.250	73.939	74.242	65.333	69.504
4			89.642	83.063	96.823	89.417	78.788	77.027	76.000	76.510
8			83.054	87.229	73.223	79.615	80.606	77.922	80.000	78.947
10			85.487	76.612	97.720	85.888	80.606	76.543	82.667	79.487
1		2	**94.432**	**93.755**	**93.936**	**93.845**	79.394	78.873	74.667	76.712
2			83.986	92.524	70.240	79.857	78.788	77.778	74.667	76.190
4			89.685	83.938	95.440	89.320	79.394	77.333	77.333	77.333
8			91.242	89.143	91.802	90.453	81.212	77.500	82.667	80.000
10			92.634	88.285	96.507	92.213	80.606	77.215	81.333	79.221
1	4	1	89.883	85.492	93.476	89.306	78.788	72.000	65.333	75.524
2			93.051	90.788	94.179	92.452	78.182	76.712	74.667	75.676
4			88.096	80.063	98.084	88.162	78.788	77.027	76.000	76.510
8			93.072	88.711	97.017	92.678	80.606	78.667	78.667	78.667
10			**93.072**	**88.677**	**97.065**	**92.682**	78.788	77.027	76.000	76.510
1		2	92.820	89.249	95.634	92.331	78.182	78.261	72.000	75.000
2			91.209	85.504	96.992	90.886	74.545	76.190	64.000	69.565
4			89.762	82.534	98.108	89.650	79.394	78.873	74.667	76.712
8			92.776	88.285	96.871	92.379	80.000	79.167	76.000	77.551
10			92.842	88.300	97.017	92.453	**81.818**	**79.221**	**81.333**	**80.263**
1	6	1	92.152	92.344	90.104	91.210	78.788	79.412	72.000	75.52
2			89.291	84.480	93.476	88.751	78.182	78.261	72.000	75.000
4			88.600	84.339	91.826	87.924	78.182	76.712	74.667	75.676
8			90.606	86.017	94.591	90.100	79.394	76.623	78.667	77.632
10			91.560	86.296	96.677	91.192	81.212	78.205	81.333	79.739
1		2	92.568	91.697	91.875	91.786	78.788	78.571	73.333	75.862
2			88.677	81.531	96.895	88.551	77.576	77.143	72.000	74.483
4			82.670	89.570	69.779	78.446	77.576	77.941	70.667	74.126
8			85.630	79.813	91.293	85.168	80.606	78.667	78.667	78.667
10			79.830	76.660	90.420	82.974	**81.818**	**79.221**	**81.333**	**80.263**

B. Training and Testing Details

Local models at each client are based on the transformer-model architecture presented in [46]. Local log files are parsed with `Spell` [13] to map the sequence M of log messages into multiple log keys sequences S_j using a sliding window frame approach of size Λ. The embedded vector representation X_j of log key sequences S_j are fed to the corresponding client's local model for training. In our study, we train the interpretable federated transformer log learning model for threat detection using the datasets in Table I. In our experiments, the following hyper-parameters were set to the following fixed values: model size $d\_model = 512$ for each hidden layer, $epochs = 5$, $dropout = 0.2$, confidence interval $g = 10$, window size $\Lambda = 10$, and number of rounds $R = 10$. Given that the learning rate does not present much variance as a function of other parameters, as shown in [34], we fixed the learning rate $\eta = 0.01$. We experimented by varying the number of layers in the encoder and decoder stacks $N = \{1, 4, 6\}$, the number of attention heads $H = \{1, 2\}$, and the number of clients contributing to the update of the global FL model $\Phi = \{1, 2, 4, 8, 10\}$.

After finishing training the global FL model, we tested the model using previously unseen non-malicious samples and the logs collected from the compromised systems. Cyber threats are detected by comparing the ground truth key G (the key following currently input sequence S_j) against the top $g = 10$ predicted candidates. In the event where G is not listed within the top g predicted candidates, the model labels the prediction as an anomaly else it labels it a normal event. The user dictates the maximum anomaly threshold (number of sequences S_j labeled as abnormal) to classify S as a cyber threat. In our experiments, we set maximum anomaly threshold to 1, instructing the model to classify S as a cyber threat in the event that a single sequence S_j with S is labeled as an anomaly by the model.

All training and testing tasks were implemented using Pytorch 1.7.1+cu110, Python 3.6.9, and OpenNMT [27] base model implementation. The model was trained in a virtual machine with a V100 GPU (CUDA 11.0) provided by Jetstream Cloud [42, 44].

C. Experimental Results

We are motivated by the existing model's performance and interpretability features for cyber threat detection tasks. Although each training run for an individual client local model is relatively small, we trained over 1,200 individual models in this experiment. In this subsection, we will first present the results obtained with the `HDFS` dataset. Afterwards, we present our proposed model's performance with our `CTDD` dataset. We compare the experimental results achieved by our proposed model with those presented by SOTA unsupervised centralized methods, namely, LogRobust [54], DeepLog [14], LogAnomaly [35], and HitAnomaly [20]. As SOTA methods used a centralized training approach, we used standard stochastic gradient descent training on the full training dataset with no client partitioning (a model built by a central entity) $\Phi = 1$ to make the intended comparison.

First, with the `HDFS` dataset, our model's parsing process identified 31 distinct event templates from the full dataset. Table IV shows that our model's best performance, achieving an F-score of 0.9384 with one encoder layer, one decoder layer, and 2 attention heads. Deeplog, which identified $E = 29$ distinct event templates, showed a better performance as the model learns from log event templates and parameter values. LogRobust identified $E = 29$ distinct event templates and showed to benefit from learning semantic information from log events and contextual information from log sequences. While their F-score performance was slightly better than ours (0.9500 and 0.9384 respectively), it shows the lowest recall between all the compared works. On the other hand, LogAnomaly which learns the semantic and syntax information from log templates, achieved an F-score of 0.9000. With an F-score of 0.9970, HitAnomaly achieved the best performance of all compared models. We attribute this achievement to the larger number of identified event templates ($E = 46$ event templates parsed with Drain). The approach taken for learning from sequences of event templates and parameter values. HitAnomaly, LogAnomaly, and LogRobust also introduce automatic approaches to deal with unstable log data while DeepLog require human intervention to deal with new templates.

The second set of experiments was also performed with the `HDFS` dataset using a federated learning setting. The federated model was evaluated using a range of different clients $\Phi = 2, 4, 8, 10$, number of layers $N = 1, 4, 6$, and attention heads $H = 1, 2$. Given the lack of integration of Federated Learning in SOTA works, we were only able to perform such an experiment using our proposed model. The experiments demonstrated that the proposed model outperforms models built by a central entity $\Phi = 1$ for most cases. Moreover, we note that the performance stabilizes as the number of rounds R is increased from 1 to 10. We also note that when fixing the number of rounds R, the performance improves as the number of clients contributing to the global FL model is increased. In other words, the model converges faster with a lower number of rounds R as the number of contributing clients Φ increases. This trend is also observed in [34]. As shown in Table III, our model achieved its peak performance (accuracy=0.9307, precision=0.8867, recall=0.9706, F-score=0.9268) when setting the hyperparameters to $\Phi = 10$, $N = 4$, $H = 1$, and 10 epochs. We also observed a stable increase in performance when using 2 attention heads instead of 1. The exception was observed when setting the number of encoder/decoder layers to 6.

The third set of experiments were performed with the `CTDD` dataset. In this dataset, the parser identified 446 unique event templates E. The higher log diversity is an indicator of the increased complexity of our dataset over the `HDFS` dataset. We can observe that this complexity impacts our model's performance across all metrics. Yet, it shows our model's capability to detect cyber threats in a real-world operational setting. We also used a standard stochastic gradient descent training on the full training dataset with no client partitioning $\Phi = 1$ to show the transformer model's performance in a centralized training approach. In contrast with our findings in the `HDFS` dataset, the performance achieved with one or two clients showed a similar performance for all cases. The peak performance (accuracy=0.7939, precision=0.7733, recall=0.7733, F-score=0.7733) in a centralized setting was achieved with 1 encoder/decoder layer and 1 attention head. Furthermore, upon source code availability, we reproduced one of the SOTA works (i.e., DeepLog[1]) with the `CTDD` dataset . The proposed model presented in DeepLog was implemented, trained, and tested with the `CTDD` dataset. From the pre-processed log data, we removed the parameter values and trained the model with sequences of log keys only to provide a proper comparison with our model. A drastic performance difference was observed in DeepLog given a F-score of 0.9600 with the `HDFS` dataset compared to a F-score of 0.7682 with the `CTDD` dataset. This is a direct impact from the increased complexity in the `CTDD` dataset as previously discussed and revealed in Fig. 2

Finally, the fourth set of experiments were performed using a federated learning approach with the `CTDD` dataset. Throughout these experiments, we observed an improved performance stability as the number of contributing clients increased. We also observed a greater performance with 2 attention heads and a slight improvement when increasing the number of encoder/decoder layers with $\Phi = 10$ clients contributing to the global model. The best performance using a federated learning setting (accuracy=0.8181, precision=0.7922, recall=0.8133, F-score=0.8026) was achieved with 4 and 6 encoder/decoder layers, 10 clients, and 2 attention heads. The experimental results are shown in detail in Table III.

In addition, an analysis on the log keys (ground truths)

TABLE IV
PERFORMANCE OF SOTA MODELS ON `HDFS` AND `CTDD` DATASETS.

Methods	Dataset	Precision	Recall	F-score
DeepLog	HDFS	0.9500	0.9600	0.9600
DeepLog	CTDD	0.7631	0.7733	0.7682
LogAnomaly	HDFS	0.8400	0.9700	0.9000
HitAnomaly	HDFS	0.9910	0.9850	0.9970
LogRobust	HDFS	1.0000	0.9100	0.9500
Proposed Method Centralized	**HDFS**	**0.9375**	**0.9393**	**0.9384**
Proposed Method Federated	**HDFS**	**0.8867**	**0.9706**	**0.9268**
Proposed Method Centralized	**CTDD**	**0.7733**	**0.7733**	**0.7733**
Proposed Method Federated	**CTDD**	**0.7922**	**0.8133**	**0.8026**

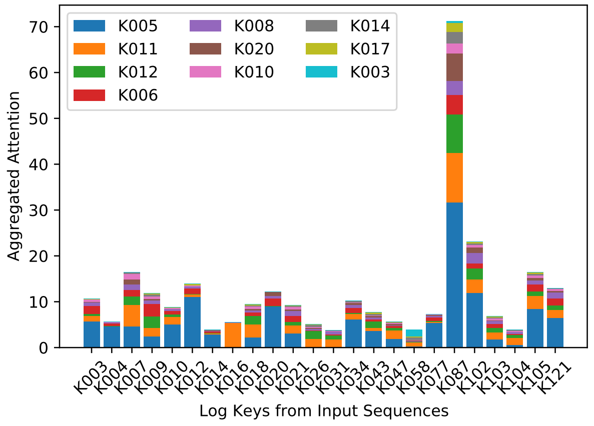
[1]DeepLog: the only publicly available code and reproducible model.

labeled as true positives and true negatives was made. In Fig. 3a each colored bar represents one of the top 10 log keys (ground truth) labeled as false positive during the inference of all the logs from uncompromised clients. The x-axis presents the 24 log keys that had the highest contribution to this classification. Similarly, in Fig. 3b each colored bar represents one of the top 10 log keys (ground truth) labeled as false positive during the inference of all the logs from compromised systems. The x-axis presents the 21 log keys that had the highest contribution to this classification. These graphs share three common log keys (K0011, K012, and K003) classified as false positive and true positive for each corresponding case.

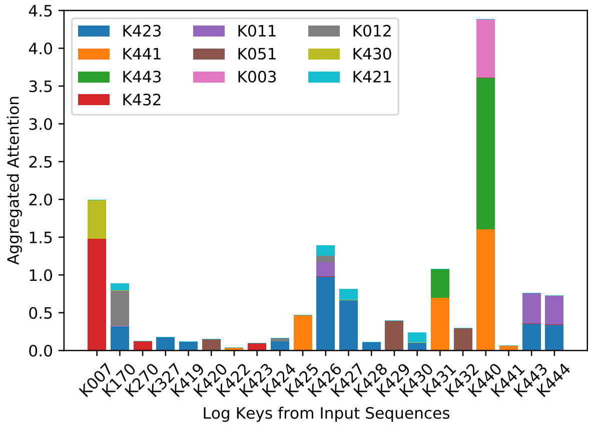
We found that the log key K011 "ens3 Configured" was often preceded by the keys K005 "ens3 DHCP lease lost" and K010 "ens3 DHCPv4 address" which are all related to the linux network manager "systemd-networkd" that assigns a defined IP address to the interface. Such sequece of operations is a normal behaviour commonly seen as normal. The log key K087 "High aggregate context switch rate" appears in the syslogs when the average number of context switches per CPU per second exceeded threshold over the past sample interval. This log is often preceded by activity from Jupyter Lab where intensive computing processes where executed by the user during short periods of time. Therefore, the model learns that 19/20 times the K011 is preceded by logs generated by the network manager and 1/20 the preceded keys will include syslogs generated by Jupyter Lab and the Linux performance co-pilot. For this reason, the model in 1/20 cases will opt to identify K011 as anomaly (true positive) if the key K087 is observed within the input sequence even when log keys linked to the network manager are also observed. For the true positives cases, the log key K011 is labeled as a true positive when it is preceded by K426 "proto precision", K443 "callbacks suppressed", and K444 "nf_conntrack table full dropping packet" which are related to the precision time protocol, network connectivity, limits on syslog messages posted by the kernel. Because K011 in most cases preceded by "systemd-networkd", the model predicts other keys than K011 and considers this as an anomaly (true positive).

V. Model Interpretability

The attention mechanism used in the presented transformer model allows us to deal with problems emerging from time-varying data (sequences) and provide an interpretation of the model's behavior. The attention mechanism looks at all the different log keys at the same time and learn to "pay attention" to the correct ones to successfully predict a log key (top g candidates) that has the highest probability to follow the given input sequence. In this context, attention is simply a notion of memory gained from attending at multiple inputs through time. During the training phase, attention weights store the memory that is gained through time of the relationship between log keys in the input sequence and the corresponding successful predictions. By inspecting the distribution of attention weights for all keys in a given input sequence, we can gain insights into the behavior of the model, as well as to understand its limitations.



(a) Top log keys (ground truths) detected as false positives in the `CTDD` dataset for normal operations.



(b) Top log keys (ground truths) detected as true positives in the `CTDD` dataset for malicious samples.

Fig. 3. Aggregated attention for each log key influencing the prediction of true positives and false positives in the `CTDD` dataset.

The attention-based interpretability module aggregates attention calculated in the transformer model, as shown in Section V, and generates visualizations presenting the difference attention distribution between normal operations and cyber threats. To demonstrate the applicability of our proposed model and the use case of our interpretability module, we present two case scenarios: (1) A real-world operational setting in which multiple business units connecting multiple uncompromised systems are contributing to the generation of a global federated model, and (2) An organization being attacked with a DoS attack.

A. Cyber Threat Forensics in a Real-World Operational Setting

For a real-world example, let us imagine an American healthcare system with a network of providers and health facilities offering a full range of healthcare services from preventative to post-acute care. The American healthcare system seeks to integrate a novel cyber threat detection system into the existing Security Information and Event Management (SIEM).

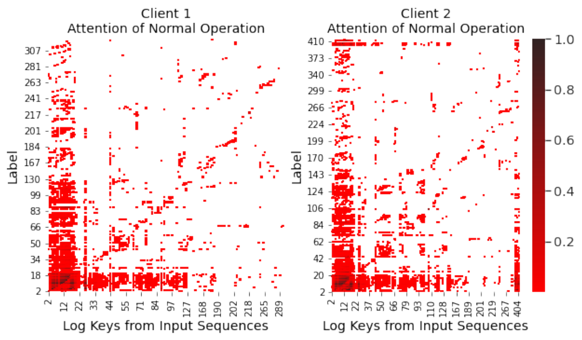


Fig. 4. Interpretability weights $Z_{\phi}^{S\Upsilon}$ computed by the client's corresponding interpretability module with their uncompromised data.

The security analysts have identified specific facilities that have not been previously compromised by threat actors. The healthcare system has maintained HIPAA compliance and has stored system, event, and audit logs from the past 6 years. Training a module for cyber threat detection in a centralized setting would require aggregating data from each provider and health facility into a single server. Nevertheless, as log files contain workforce members login, failed login attempts, software updates, downloaded programs, change of passwords, EHR logins, patient data access, changes to eHPI, among other information, sharing logs to a central entity poses high security risks and potential violations to HIPAA. To prevent such a scenario and facilitate model learning from log data for multiple health facilities, the security administrators decide to use our interpretable federated transformer log learning model for threat forensics.

After each care facility trains a local model using exclusively their local data, they use the interpretability module to identify the log keys (from a set of input sequences) that had the highest influence to the model's correct or erroneous predictions. As the model was trained with data from uncompromised systems, the initial saliency map computed by the interpretability module provides a representation of the attention distribution during normal operations for all log keys identified across all input sequences with respect to each evaluated ground truth label.

In Fig. 4 the interpretability module breaks down the attention of every key in the input sequence influencing each correctly predicted log key (ground truth). The saliency map shows that lower keys highly influence most correctly predicted log keys (ground truths) following all input sequences. Different from client 1, we can observe that the model's predictions for client 2 had a higher influence from log keys plotted in the right-hand side of the subplot. Yet, for both clients, we can observe that log keys plotted in the left-hand side of each plot had the highest influence to the model's prediction.

B. Attack Scenario

In this scenario, an organization contracts a hacker to launch a cyber attack on its competitor with a motive to bring operational disruption and distraction. In this case, the hacker/attacker intends to launch a volumetric DDoS attack

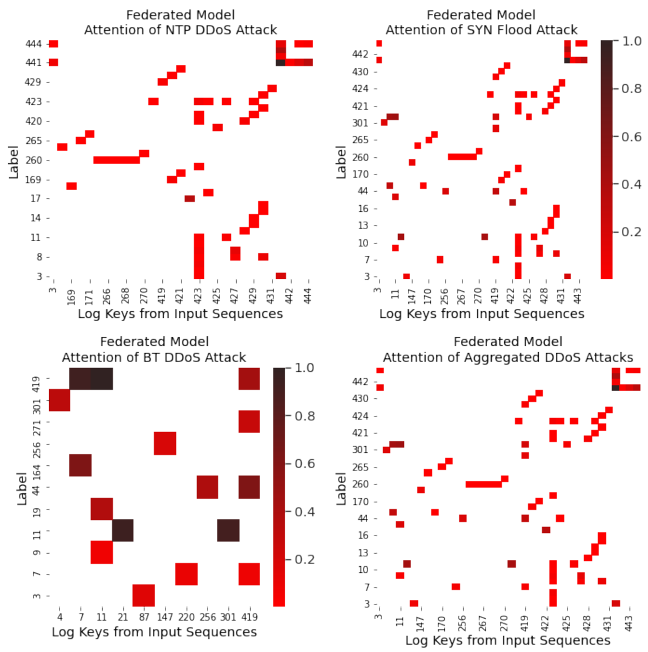


Fig. 5. Top Left: Aggregated attention by input log key of NTP DoS attack. Top Right: Aggregated attention by input log key of SYN Flood attack. Bottom Left: Aggregated attention by input log key of BT DoS attack. Bottom Right: Federated attention of NTP DDoS, SYN Flood, and BT DDoS attacks.

against the target organization's web server. The attacker uses a popular scanning tool in the reconnaissance phase to find that the victim's organization is running Ubuntu 18.04 and is protected with a simple firewall. The attacker notices that an NTP amplification DDoS attack would be a perfect protocol to exploit as the victim's web server had its UDP port 123 open and the server seems to be an open resolver. As the attacker understands that the victim's organization is unable to detect the source of the attack, the attacker decides to carry out the NTP amplification DDoS attack rather than other volumetric DDoS attacks. The attacker aims to send high amounts of spoofed requests to the NTP servers that has the monlist command enabled with the response pointing the victim organization's web server. The attacker now controls the NTP server and sends out large amounts of UDP responses back to the victim's web server. This consumes the victim's network bandwidth, disrupting normal operation and service availability to the victim organization's clients. The successful attack allows the attacker to disrupt the victim organization and its clients for a desired amount of time.

The organization's forensic investigators use the interpretability module to backtrace the system activity, recorded in the syslogs, that triggered the detection of a cyber threat. Fig. 6 presents a series of sub-sequences S_j from the victim's syslog. The associated templates to each key κ in S_j is presented in Fig. 7. Each sub-sequence S_j is evaluated by the model to compute the attention for each log key and predict the top g candidates. We can observe that the predicted candidates for sub-sequences S_2 and S_3 do not match their corresponding ground truth keys G_j, which are subsequently labeled as anomalies.

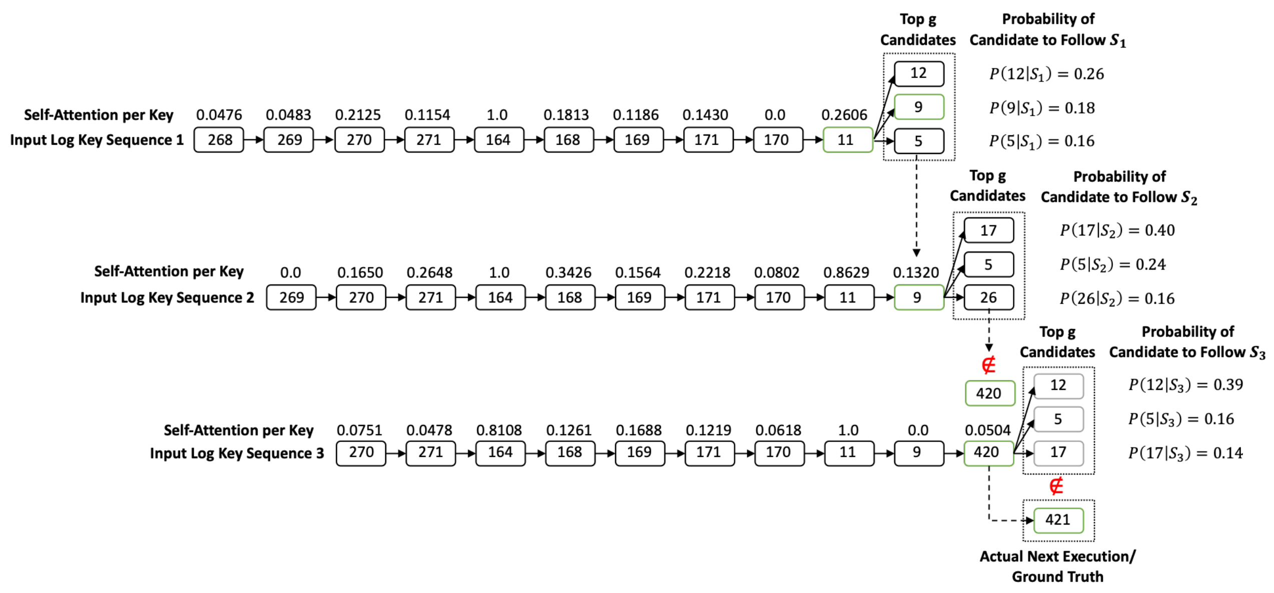


Fig. 6. Log key sub-sequences of NTP DDoS attack in sequence S processed by the federated model for predicting the g candidates. The "Ground Truth Keys" following sub-sequences $\{S_1, S_2, S_3\} \in S$ are compared against the corresponding predicted list of g candidates for identifying anomalies leading to cyber threats. The number of log keys for each sub-sequence is determined by the window frame parameter Λ. For every sub-sequence S_j, attention is calculated for every key in S_j. In the proposed transformer-based model, attention is ultimately passed to a fully connected layer and subsequently to a Softmax layer for predicting the top g candidates and the probability for each of them. For this case scenario, in sub-sequence S_1 the attention concentrates highly in keys 164, 168, and 11 to correctly predict the log key 9 that follows the sub-sequence. In contrast, the following two sub-sequences S_2 and S_3 show the attention concentrated in keys 164, 169, and 11. This last log key showed an attention 3 to 4 times larger than in S_1. This difference in attention across generates predictions of top g candidates that do not match the following ground truth keys, indicating a deviation from the normal system behavior that leads to the detection of this cyber threat.

Additionally, forensic investigators can compare the saliency map of the NTP attack with the ones computed from interpretability weights computed for other attacks. In this case, the saliency map for previously known SYN Flood and BT DDoS attacks is shown in Fig. 5. By comparing the saliency maps of the NTP DDoS and SYN Flood attacks, we can observe their shared attention distribution. Further, the distribution similarity of each attack can be compared against a saliency map of the federated interpretability weights $Z_{\Phi}^{S\Upsilon}$ of the previously mentioned attacks. In Fig. 5, we present the interpretability weights of the NTP DDoS, BT DDoS, SYN Flood, and the federated interpretability weights aggregating these. This feature, along with the previous features presented, provides forensic investigators with actionable information that leads to an efficient analysis of system operation.

```
Log Key - Log Template

  268     <*> <*> ctnetlink v0.<*> registering wit...
  269     link_config autonegotiation is unset or ...
  270     WARNING Unknown index <*> seen reloading...
  271     <*> Link UP
  164     time <*> <*>"" level info msg ""Loading ...
  168     time <*> <*>"" level info msg ""Docker d...
  169     time <*> <*>"" level info msg ""Daemon h...
  171     time <*> <*>"" level info msg ""API list...
  170     Started Docker Application Container Eng...
  11      ens3 Configured
    9     Started ntp-systemd-netif.service.
  420     Stopping Network Time Service...
  421     ntpd exiting on signal <*> Terminated
```

Fig. 7. Corresponding log templates of each key in evaluated sequence of NTP DDoS attack.

C. Hidden attacks and Negative Cases

In order to explore the limitations of our proposed model, we investigated two different case studies that includes hidden attacks (i.e, low syslogs footprint attacks) and negative cases (i.e., failed anomaly detection cases). That said, we developed a low footprint Ransomware Attack (Table II) that encrypts and decrypts both "/opt/" and "/proc/" directories on a Linux-based system. The ransomware attack posted a very low number of syslogs demonstrating its low footprint. Nonetheless, our model was able to detect its activities as malicious with high precision. On the other hand, one of the attack cases that we considered in Table II, namely, ech0raix manifested as a negative case. In this instance, our proposed model failed to detect any malicious activity linked to ech0raix. After investigating the logs, we associated this failed case to the fact that all ech0raix network traffic activities where posted to the kernel logs (detected by UFW), while our study only focuses on syslogs.

To that extent, we can confidently state that our proposed model is able to detect hidden attacks, while possible negative cases may still occur due to distinct log posting (e.g., kernel logs, auth logs).

VI. Discussion

In the scenario presented in the previous section, we used the interpretability module to generate a saliency map from logs of a system targeted by a NTP DDoS attack and the saliency map from logs collected from uncompromised client systems.

The same approach was followed to evaluate all log sequences from uncompromised and compromised systems included in our `CTDD` dataset. The saliency map presented in Fig. 8 (left) shows the aggregated attention for each log key observed across all input sequences (obtained from uncompromised client logs) where the log key (ground truth) following the input sequence was correctly predicted. Similarly, the saliency map presented in Fig. 8 (right) shows the aggregated attention for each log key observed across all input sequences (obtained from compromised client logs) where the log key (ground truth) following the input sequence did not match any of the top g candidates predicted by the model.

When comparing the two saliency maps, we can observe the attention distribution difference across all log keys. Also, we can observe that most of the attention in uncompromised systems concentrates in log keys with lower value while attention in compromised systems, shown in Fig. 8 (right), concentrate in log keys with higher value. From this information, we can note that log keys in the input sequence with high values highly influence the detection of anomalies (true positives) and the correct prediction for true negatives.

To measure the distribution difference for a specific sample, the interpretability module performs a goodness of fit test for the corresponding $Z_{\phi}^{S\Upsilon}$ with respect to a given federated $Z_{\Phi}^{S\Upsilon}$. We computed $Z_{\phi}^{S\Upsilon}$ for the syslogs of the cyber threats listed in Table II. We calculated the chi-square statistic for each $Z_{\phi}^{S\Upsilon}$ with respect to the federated $Z_{\Phi}^{S\Upsilon}$ of all cyber threats, shown in Fig. 8 (right). We performed the same comparison with respect to the federated $Z_{\Phi}^{S\Upsilon}$ of uncompromised systems, shown in Fig. 8 (left).

The chi-square statistic $Z_{\phi}^{S\Upsilon}$ of each cyber threat with respect to the federated $Z_{\Phi}^{S\Upsilon}$ of compromised systems, shown in Fig. 9 (left), computed to an average of 42.664 and a $p-value = 1$. Given these results, we can confidently state that the attention obtained from the cyber threats logs share the same distribution as the federated $Z_{\Phi}^{S\Upsilon}$ of compromised systems. On the other hand, the chi-square statistic of each cyber threat with respect

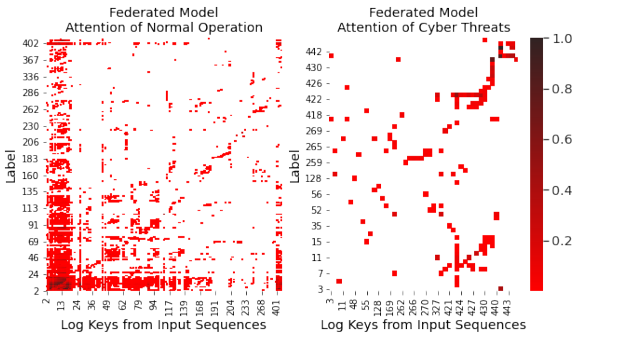


Fig. 8. On the left: Saliency map of aggregated attention of Clients 1 and 2 computed by the FL interpretability module. On the right: The saliency maps show a clear difference in the distribution of attention from independent sample data points representing different case scenarios (normal non-compromised environment and NTP DDoS cyber threat). The attention of client systems shows a higher concentration on $k_i \in K, i = 1, \ldots, 26$ while the attention of the NTP DDoS cyber threat concentrates on $k_i \in K, i = 424, \ldots, 444$.

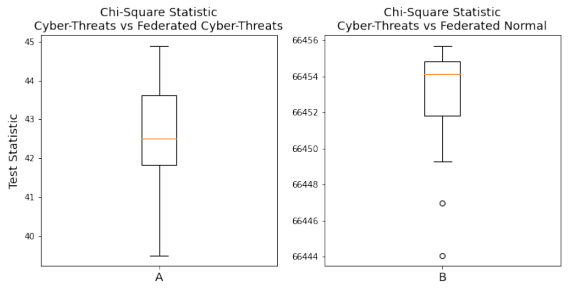


Fig. 9. Box plot of Chi-Square statistic computed for $Z_{\phi}^{S\Upsilon}$ of each cyberattack in the `CTDD` dataset with respect to the federated$Z_{\Phi}^{S\Upsilon}$ of all cyber threats (left) and $Z_{\Phi}^{S\Upsilon}$ of all uncompromsed systems.

to federated $Z_{\Phi}^{S\Upsilon}$ of uncompromised systems, shown in Fig. 9 (right), computed to an average of 66452.6871 and a $p-value = 0$. The chi-square statistics for the latter comparison were far greater than the range within the chi-square statistic with 445 degrees of freedom. Given these results, we can note that the attention distribution in $Z_{\phi}^{S\Upsilon}$ for any cyber threat differs from the distribution of federated $Z_{\Phi}^{S\Upsilon}$ of uncompromised systems.

VII. Conclusion

We proposed an interpretable federated transformer log learning model for detecting cyber threats with interpretability capabilities useful for threat forensics. Existing approaches consist of centralized anomaly detection models that overlook data privacy and data jurisdiction laws. As of now, none of the existing works explore the interpretability of the model's predicted outcomes, obscuring the user's visibility into the model's decision-making factors. In this way, the techniques presented in this paper are an improvement over SOTA works. The proposed approach constructs a time-series vector from log sequences extracted from syslogs, in order to capture system operational activity and user activity. Using an unsupervised approach, a transformer-based model at each client learns the underlying patterns of the time series. In addition, it integrates a federated learning approach for aggregating the learned patterns from local models to produce an updated global FL model. Furthermore, it uses the attention values to provide visibility to the model's decision-making process and highlights differences in attention between normal sequences and threat sequences.

Our approach demonstrated its log agnostic capability and applicability on high-dimensional time series. Our model's peak F-score (93.84%) in the `HDFS` dataset was achieved using two encoder and decoder layers, and 1 attention head. Moreover, our work outclasses SOTA works by integrating data privacy and interpretability that reveal indicators, as well as, main contributor's for a model's decision-making process. In future work, we will be exploring the applicability of our approach on different types of multivariate time-series data including network and audit logs.

References

[1] "Check point research," accessed: 2021-04-29. [Online]. Available: https://blog.checkpoint.com/2021/01/05/attacks-targeting-healthcare-organizations-spike-globally-as-covid-19-ca_ses-rise-again/

[2] "Cisa," accessed: 2021-04-29. [Online]. Available: https://us-cert.cisa.gov/sites/default/files/publications/AA20-302A_Ransomware\%20_Activity_Targeting_the_Healthcare_and_Public_Health_Sector.pdf

[3] "Ctdd," accessed: 2021-05-21. [Online]. Available: https://github.com/cyberthreat-datasets/ctdd-2021-os-syslogs

[4] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140 699–140 725, 2020.

[5] N. Bendre, H. T. Marín, and P. Najafirad, "Learning from few samples: A survey," *arXiv preprint arXiv:2007.15484*, 2020.

[6] A. Brown, A. Tuor, B. Hutchinson, and N. Nichols, "Recurrent neural network attention mechanisms for interpretable system log anomaly detection," in *Proceedings of the First Workshop on Machine Learning for Computing Systems*, 2018, pp. 1–8.

[7] M. Burguess. (2020) Hackers are targeting hospitals crippled by coronavirus. [Online]. Available: https://www.wired.co.uk/article/coronavirus-hackers-cybercrime-phishing

[8] H. by LogPAI Team, "Loghub," Jan. 2018. [Online]. Available: https://doi.org/10.5281/zenodo.3227177

[9] H. Chacon, S. Silva, and P. Rad, "Deep learning poison data attack detection," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2019, pp. 971–978.

[10] H. D. Chacon and P. Rad, "Effect of backdoor attacks over the complexity of the latent space distribution," *arXiv preprint arXiv:2012.01931*, 2020.

[11] J. Chow, B. Pfaff, T. Garfinkel, K. Christopher, and M. Rosenblum, "Understanding data lifetime via whole system simulation," in *USENIX Security Symposium*, 2004, pp. 321–336.

[12] R. Clarke and T. Youngstein, "Cyberattack on britain's national health service—a wake-up call for modern medicine," *N Engl J Med*, vol. 377, no. 5, pp. 409–11, 2017.

[13] M. Du and F. Li, "Spell: Streaming parsing of system event logs," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 859–864.

[14] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1285–1298.

[15] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020.

[16] A. Goel, K. Po, K. Farhadi, Z. Li, and E. De Lara, "The taser intrusion recovery system," in *Proceedings of the twentieth ACM symposium on Operating systems principles*, 2005, pp. 163–176.

[17] S. E. Hansen and E. T. Atkins, "Automated system monitoring and notification with swatch." in *LISA*, vol. 93, 1993, pp. 145–152.

[18] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "Towards automated log parsing for large-scale log data analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 931–944, 2017.

[19] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *2017 IEEE International Conference on Web Services (ICWS)*. IEEE, 2017, pp. 33–40.

[20] S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, and Z. Luan, "Hitanomaly: Hierarchical transformers for anomaly detection in system log," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2064–2076, 2020.

[21] A. D. Kent, "Comprehensive, multi-source cyber-security events data set," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2015.

[22] A. Kharaz, S. Arshad, C. Mulliner, W. Robertson, and E. Kirda, "{UNVEIL}: A large-scale, automated approach to detecting ransomware," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 757–772.

[23] I. F. Kilincer, F. Ertam, and A. Sengur, "Machine learning methods for cyber security intrusion detection: Datasets and comparative study," *Computer Networks*, p. 107840, 2021.

[24] T. Kim, X. Wang, N. Zeldovich, M. F. Kaashoek *et al.*, "Intrusion recovery using selective re-execution." in *OSDI*, 2010, pp. 89–104.

[25] S. T. King and P. M. Chen, "Backtracking intrusions," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, 2003, pp. 223–236.

[26] S. T. King, Z. M. Mao, D. G. Lucchetti, and P. M. Chen, "Enriching intrusion alerts through multi-host causality." in *NDSS*. Citeseer, 2005.

[27] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "Opennmt: Open-source toolkit for neural machine translation," in *Proc. ACL*, 2017. [Online]. Available: https://doi.org/10.18653/v1/P17-4012

[28] C. Kuner, "Data protection law and international jurisdiction on the internet (part 1)," *International Journal of Law and Information Technology*, vol. 18, no. 2, pp. 176–193, 2010.

[29] K. H. Lee, X. Zhang, and D. Xu, "High accuracy attack provenance via binary-based execution partition." in *NDSS*, 2013.

[30] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "Deepfed: Federated deep learning for intrusion detection in industrial cyber-physical systems," *IEEE Transactions on Industrial Informatics*, 2020.

[31] B. Lindauer, J. Glasser, M. Rosen, K. C. Wallnau, and L. ExactData, "Generating test data for insider threat detectors." *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, vol. 5, no. 2, pp. 80–94, 2014.

[32] Y. Liu, M. Zhang, D. Li, K. Jee, Z. Li, Z. Wu, J. Rhee, and P. Mittal, "Towards a timely causality analysis for enterprise security." in *NDSS*, 2018.

[33] S. Lu, X. Wei, Y. Li, and L. Wang, "Detecting anomaly in big data system logs using convolutional neural network," in *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*. IEEE, 2018, pp. 151–158.

[34] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[35] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun *et al.*, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs." in *IJCAI*, 2019, pp. 4739–4745.

[36] K.-K. Muniswamy-Reddy, U. J. Braun, D. A. Holland, P. Macko, D. Maclean, D. W. Margo, M. I. Seltzer, and R. Smogor, "Layering in provenance systems," in *Proceedings of the 2009 USENIX Annual Technical Conference (USENIX'09)*. USENIX Association, 2009.

[37] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for iot," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 756–767.

[38] D. J. Pohly, S. McLaughlin, P. McDaniel, and K. Butler, "Hi-fi: collecting high-fidelity whole-system provenance," in *Proceedings of the 28th Annual Computer Security Applications Conference*, 2012, pp. 259–268.

[39] J. E. Prewett, "Analyzing cluster log files using logsurfer," in *Proceedings of the 4th Annual Conference on Linux Clusters*. Citeseer, 2003.

[40] J. P. Rouillard, "Real-time log file analysis using the simple event correlator (sec)." in *LISA*, vol. 4, 2004, pp. 133–150.

[41] A. J. Slagell, K. Lakkaraju, and K. Luo, "Flaim: A multi-level anonymization framework for computer and network logs." in *LISA*, vol. 6, 2006, pp. 3–8.

[42] C. A. Stewart, T. M. Cockerill, I. Foster, D. Hancock, N. Merchant, E. Skidmore, D. Stanzione, J. Taylor, S. Tuecke, G. Turner *et al.*, "Jetstream: a self-provisioned, scalable science and engineering cloud environment," in *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, 2015, pp. 1–8.

[43] D. J. B. Svantesson, *Extraterritoriality in data privacy law*. Ex Tuto Publishing, 2013.

[44] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson *et al.*, "Xsede: accelerating scientific discovery," *Computing in science & engineering*, vol. 16, no. 5, pp. 62–74, 2014.

[45] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," *arXiv preprint arXiv:1710.00811*, 2017.

[46] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[47] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019.

[48] S. R. Wibisono and A. I. Kistijantoro, “Log anomaly detection using adaptive universal transformer,” in *2019 International Conference of Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*. IEEE, 2019, pp. 1–6.

[49] J. Wiggen, “The impact of covid-19 on cyber crime and state-sponsored cyber activities,” 2020.

[50] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan, “Largescale system problem detection by mining console logs,” *Proceedings of SOSP’09*, 2009.

[51] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, “Detecting large-scale system problems by mining console logs,” in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, 2009, pp. 117–132.

[52] R. B. Yadav, P. S. Kumar, and S. V. Dhavale, “A survey on log anomaly detection using deep learning,” in *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*. IEEE, 2020, pp. 1215–1220.

[53] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[54] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li *et al.*, “Robust log-based anomaly detection on unstable log data,” in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 807–817.

[55] Y. Zhao, J. Chen, D. Wu, J. Teng, and S. Yu, “Multi-task network anomaly detection using federated learning,” in *Proceedings of the Tenth International Symposium on Information and Communication Technology*, 2019, pp. 273–279.