

# P3GNN: A Privacy-Preserving Provenance Graph-Based Model for APT Detection in Software Defined Networking

Hedyeh Nazari<sup>1</sup>, Abbas Yazdinejad<sup>1</sup>, Ali Dehghanianha<sup>1</sup>, Fattane Zarrinkalam<sup>2</sup>, and Gautam Srivastava<sup>3,4</sup>

<sup>1</sup> Cyber Science Lab, Canada Cyber Foundry, University of Guelph, Canada  
[hnazari@uoguelph.ca](mailto:hnazari@uoguelph.ca), [ayazdine@uoguelph.ca](mailto:ayazdine@uoguelph.ca), [adehghan@uoguelph.ca](mailto:adehghan@uoguelph.ca)

<sup>2</sup> Dept. of Engineering, University of Guelph, Canada  
[fzarrink@uoguelph.ca](mailto:fzarrink@uoguelph.ca)

<sup>3</sup> Dept. of Math and Computer Science, Brandon University, Manitoba, Canada  
[srivastavag@brandou.ca](mailto:srivastavag@brandou.ca)

<sup>4</sup> Research Centre for Interneuronal Computing, China Medical University, Taichung, Taiwan  
[srivastavag@brandou.ca](mailto:srivastavag@brandou.ca)

**Abstract.** Software Defined Networking (SDN) has brought significant advancements in network management and programmability. However, this evolution has also heightened vulnerability to Advanced Persistent Threats (APTs), sophisticated and stealthy cyberattacks that traditional detection methods often fail to counter, especially in the face of zero-day exploits. A prevalent issue is the inadequacy of existing strategies to detect novel threats while addressing data privacy concerns in collaborative learning scenarios. This paper presents P3GNN (privacy-preserving provenance graph-based graph neural network (GNN) model), a novel model that synergizes Federated Learning (FL) with Graph Convolutional Networks (GCN) for effective APT detection in SDN environments. P3GNN utilizes unsupervised learning to analyze operational patterns within provenance graphs, identifying deviations indicative of security breaches. Its core feature is the integration of FL with homomorphic encryption, which fortifies data confidentiality and gradient integrity during collaborative learning. This approach addresses the critical challenge of data privacy in shared learning contexts. Key innovations of P3GNN include its ability to detect anomalies at the node level within provenance graphs, offering a detailed view of attack trajectories and enhancing security analysis. Furthermore, the model's unsupervised learning capability enables it to identify zero-day attacks by learning standard operational patterns. Empirical evaluation using the DARPA TCE3 dataset demonstrates P3GNN's exceptional performance, achieving an accuracy of 0.93 and a low false positive rate of 0.06.

**Keywords:** Advanced Persistent Threat, SDN, Graph Neural Network, Federated Learning.

---

This paper has been submitted to the ACM CCS 2024 Conference, AutonomousCyber 2024 Workshop for peer review and potential publication.

## 1 Introduction

SDN represents a novel approach in network architecture that aims to address prevailing challenges by separating the control plane from the data plane and using protocols such as OpenFlow [12,34] for communication between these planes [35]. This approach turns network switches into simple packet handlers, with decision-making centralized in a software controller. The controller offers a comprehensive overview of the network and programming abstractions, enabling a centralized management system. This system grants operators programmatic and real-time control over networks and devices, simplifying network management and introducing flexibility that was previously lacking [13,4]. However, the adoption of SDN introduces new security concerns, such as APT [26,36]. APTs pose an immense and formidable threat to cyberspace security. Characterized by their stealth and slow infiltration, APTs are typically launched by well-resourced groups, like state actors, and employ advanced tactics, techniques, and procedures (TTPs) for long-term network presence, targeting espionage or sabotage [2]. As APTs become more sophisticated, cybersecurity solutions must become more dynamic and adaptable.

Diverse strategies utilizing log files to counter APTs have evolved due to the limitations of traditional signature-based detection systems, particularly against zero-day exploits and advanced polymorphic malware [17]. Among these, Host-based Intrusion Detection Systems (HIDS) stand out for their efficacy. Further, integrating HIDS into SDN networks appears promising for enhanced security [37]. Building on HIDS, the Provenance-based Intrusion Detection System (PIDS), especially its provenance graph method, utilizes provenance data's contextual information to pinpoint APT attacks [31]. In single-host environments, a system provenance graph, structured as a directed acyclic graph (DAG), categorizes log file elements like processes, files, and sockets as vertices and their interactions as edges. This method effectively maps system processes, illustrating control and data flow between entities[16,22]. Previous research on log provenance graphs for APT detection faces limitations, particularly with misuse-based methods' inability to detect zero-day attacks. These methods, which compare provenance graphs to known attack graphs, struggle to identify new threats not captured in existing cyber threat reports [20]. In response, unsupervised anomaly-based methods offer a promising alternative by focusing on deviations from normal behavior within provenance graphs, essential for detecting new and emerging threats, including zero-day attacks [24].

Previous research on detecting APTs using provenance graphs highlights significant challenges due to the complexity of these dense networks that represent system interactions. GNNs, in contrast to standard Deep Neural Networks (DNNs), are better suited for these tasks due to their ability to interpret graph-structured data, enabling them to not only detect but also visualize malicious activities within network contexts [19,15]. However, the effectiveness of GNNs is often limited by the availability of extensive log datasets necessary for training. Despite the potential of centralized training to mitigate data scarcity by pooling resources from multiple sources, practical implementation is hindered by privacy

and security concerns, preventing organizations from sharing sensitive log data [7,23].

Considering these challenges, in this paper, we introduce P3GNN, an explainable privacy-preserving model specifically designed for APT detection. P3GNN innovatively combines the principles of FL with GCN, using unsupervised learning to examine operational patterns in provenance graphs for identifying deviations that signal security breaches [9]. A key aspect of P3GNN is its strong focus on privacy and security. By integrating FL with homomorphic encryption, specifically the Paillier homomorphic cryptosystem (PHC), it ensures the confidentiality and integrity of data and gradients during collaborative learning, offering dual-layer protection for sensitive information [11]. This integration is vital because, while FL allows for collaborative model training across different organizations without direct data sharing, it inherently provides limited privacy protection and doesn't fully resolve security issues, especially regarding data shared with third-party servers [27]. The concern of an untrustworthy server exploiting uploaded parameters to infer feature values from private training datasets is effectively addressed by the addition of partially homomorphic encryption in P3GNN. Additionally, P3GNN's capability to detect APTs at the node level within provenance graphs, identify anomalous nodes, and trace the full attack trajectory, is crucial for security analysts. This functionality offers a comprehensive view of attack paths and enhances the analysis of APT incidents. The main contributions of this paper can be summarized as follows:

- We propose a novel interpretable, privacy-preserving provenance graph-based FL model P3GNN for detecting APTs. Our model leverages GCN to adeptly navigate the intricate interconnections present in log provenance graphs.
- Our model operates in an unsupervised manner, effectively learning the standard operational patterns within provenance graphs and thereby identifying any deviations from the system's typical behavior. This capability enables our system to detect anomalies that could signify zero-day attacks, which are particularly challenging to identify due to their novel and previously unrecorded nature.
- The integration of FL with homomorphic encryption in P3GNN maintains the confidentiality of data and gradients in collaborative learning, addressing data sharing challenges and protecting log owners' privacy. This allows training on diverse datasets, including attack samples from multiple sources, without transferring sensitive log data. Training is conducted locally, leveraging distributed data while ensuring privacy and security.
- P3GNN can pinpoint anomalous nodes within provenance graphs at the node level. This feature significantly enhances the model's capability to trace the entire trajectory of APT attacks, providing valuable insights for security analysis. This attribute of P3GNN addresses a common issue in the field of artificial intelligence regarding interpretability.

The following is the specification of the threat model we consider in this paper.

- **High Volume of Log Data with Subtle Anomalies:** The main challenge in detecting APTs within SDNs lies in the sheer volume of log data,

where malicious activity can easily masquerade as normal system behavior. This makes it challenging to identify threats without examining the interconnections among various system events. Furthermore, the subtlety of such activities necessitates deep analysis. Advanced ML techniques are crucial for effectively identifying these hidden threats.

- **Privacy Risk in Data Sharing for GNN Training:** In a decentralized infrastructure for GNN training, sharing sensitive log data with external entities poses a significant privacy risk, potentially exposing critical operational details.
- **Confidentiality of Gradients in FL setup:** The confidentiality of gradients is a concern, as an honest but curious server could potentially access sensitive user information through shared gradients and model parameters. An adversary can gain access to the private logs of an organization through gradient inversion attacks [14].

By achieving our contributions, we address the following design goals in this research:

- **Enhanced Anomaly Detection in Large Log Data:** To tackle the challenge of detecting APTs within vast log datasets, our solution employs an anomaly-based detection system using provenance graphs. These graphs help interpret the relationships and context within system logs, identifying deviations from normal behavior. We use GNNs to effectively handle and analyze these complexities, allowing for detailed tracking of anomalies at the node level and enabling quick investigations.
- **Privacy-Preserving Data Handling for GNN Training:** In response to the privacy risks associated with data sharing in a decentralized GNN training infrastructure, our model leverages FL. This approach allows the training data to remain within the confines of the organization’s secure environment, thereby negating the need to transfer sensitive information externally. This method ensures that privacy is maintained during the GNN training process.
- **Secure Gradient Sharing:** To mitigate the risk of gradient inversion attacks and protect user data, the model incorporates PHC for encrypting local gradients before transmission to the server. This encryption step is crucial in maintaining the confidentiality of sensitive information throughout the gradient-sharing phase, ensuring robust protection against potential adversarial access.

The rest of this paper is organized as follows. Section 2 reviews the related literature. Section 3 introduces the structure of the P3GNN model. Section 4 evaluates the proposed model, compares the results to related works, and provides interpretability and complexity analysis for the model. Lastly, we conclude the paper in section 5.

## 2 Related Work

APT detection has been the subject of numerous studies. In provenance-based methods, Milajerdi et al. [21] developed Holmes, a real-time APT detection sys-

tem that utilizes provenance graphs for correlating suspicious information flows. It employs expert knowledge of TTPs to match defined exploits within these graphs. This integration allows Holmes to effectively summarize attackers' actions through robust detection signals and high-level graphs, offering precise detection and low false alarm rates in real-world APT scenarios. Also, in [33], Xie et al. proposed a model named Pangoda, designed for large data environments. It analyzes provenance graphs to detect anomalies, balancing the focus between individual paths and the overall graph structure. This enables quick identification of intrusions and improves detection accuracy. In [30], the authors introduce ProvDetector, a malware detection tool that utilizes provenance graphs. ProvDetector operates by embedding paths within a provenance graph and applying the Local Outlier Factor method to identify malware effectively. Poirot, as proposed in [20], is another provenance-based system designed for threat hunting. It focuses on correlating indicators identified by other systems and constructs attack graphs using expert knowledge from existing cyber threat reports. The system matches the provenance graphs with these attack graphs to detect threats, but it faces challenges in identifying unknown threats not already included in the TTPs and cyber threat reports. DeepTaskAPT introduced in [18] is an LSTM-based deep learning model for detecting APTs by analyzing event sequences in the provenance graph. This model identifies unusual activities by being trained solely on benign sequences, which are created using a tree-based method for task generation. In [8], a system named UNICORN is presented for detecting APT anomalies using sketch-based, time-weighted provenance encoding and graph analysis. It focuses on supply-chain attacks, with multi-hop exploration and evolutionary modeling enhancing accuracy and efficiency.

In the context of APT detection in SDNs, the authors in [25] introduce a Hidden Markov Model-based APT detection method that effectively detects and analyzes multi-stage APTs in SDN, offering accurate threat identification with minimal overhead. In [26], they propose a method that employs an attack tree-based model for SDN, enabling the detection of APTs by correlating multiple attack behaviors to establish an attack path. However, these methods fail to take into account the privacy of log owners. Huynh Thai Thi et al. present an FL approach for enhancing cyber threat detection in SDN. This method leverages collaborative intelligence across multiple parties while preserving data privacy. The technique focuses on proactive APT attack detection, utilizing FL to improve the performance of detection systems against unknown threats. For the IDS model, the authors employed three local models, including Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), and Convolutional Neural Network (CNN) [28]. In [29], the XFedHunter model uses the FedAvg algorithm for decentralized APT detection in SDNs, enhancing privacy. It integrates CNN, GRU, and E-GraphSAGE for intrusion detection and employs the SHAP framework for model explainability. While this approach focuses on privacy by design, it does not explicitly protect model parameters like gradients and weights during training.

### 3 Proposed Model

In this section, we present an in-depth view of our model, P3GNN, specifically designed for detecting APTs in SDNs. The P3GNN model integrates three key technological paradigms: the APT detection procedure, the SDN infrastructure, and, the collaborative and privacy-preserving aspects of FL. Each of these components plays a pivotal role in creating a robust, efficient, and privacy-preserving APT detection system. Figure 1 shows the architecture of the different components in P3GNN. The following subsections will explain each component's contributions as well as how they work together to enhance our model's effectiveness and security.

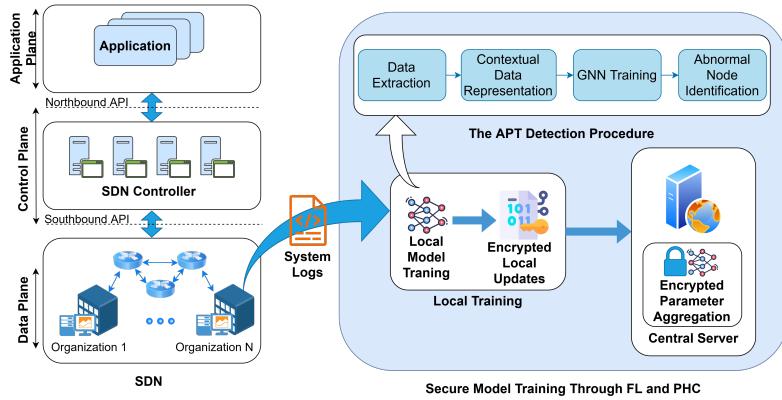


Fig. 1: The architecture of P3GNN

#### 3.1 SDN Infrastructure

In the design of our system, we view the SDN network as a potential zone for detecting malicious activities, which are likely to appear within the network's flow data. SDN encompasses three planes: the Control Plane, responsible for traffic direction decisions, often on a network operating system; the Data Plane, comprising physical or virtual routers and switches that forward packets per control plane instructions; and the management plane, overseeing network administration, monitoring, and policy enforcement. SDN utilizes northbound APIs for communication between the control plane and higher-level applications, enabling network programming, and southbound APIs (like OpenFlow) for connecting the control plane to data plane devices for management. To effectively monitor and scrutinize this data for any signs of such activities, it's essential to first set up the SDN network switches. A fundamental component of this process involves the data plane functionality of SDN switching devices, which are pivotal

in packet forwarding. The data plane in an SDN switch operates distinctly compared to traditional networking devices. While legacy networks primarily rely on forwarding packets based on IP or MAC addresses, SDN switches introduce a more versatile approach [32]. These switches are tasked with forwarding network flow data for APT detection analysis via the OpenFlow protocol.

### 3.2 APT Detection Procedure in P3GNN

The APT detection procedure of the P3GNN model as shown in Figure 2 is structured into four phases: data extraction, contextual data representation, GNN training, and finally abnormal node identification and analysis. The subsequent sections will provide an in-depth exploration of each phase, detailing their functionalities and contributions to the model’s capability in APT detection.

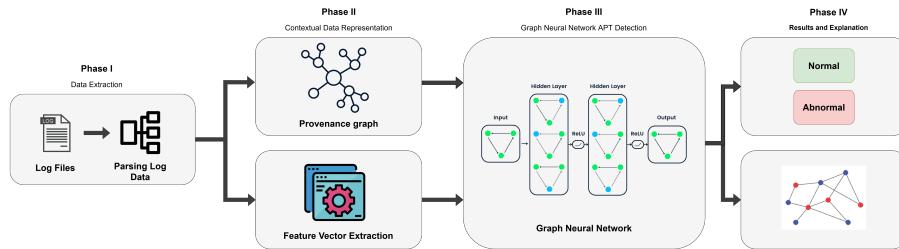


Fig. 2: APT detection procedure

**Data extraction:** In the initial phase of our model, we focus on the extraction and initial preprocessing of log data. The primary source of our data comprises system logs, encompassing a wide range of information such as timestamps, process names, and execution paths. Utilizing regular expression techniques, we parse these logs to transform the raw data into a structured format. We define  $\Sigma$  as the set of all unique symbols present in the log data. Using regular expressions, we create definitions for specific patterns found in log entries. For example, to identify a UUID, we use a regular expression denoted as  $R_{\text{uuid}}$ . This expression is defined as "uuid:" ·  $(\Sigma^*)$  · "", where  $\Sigma^*$  represents any sequence of characters from our defined symbol set.

The process of transforming log entries into structured data is encapsulated by the parsing function ParseLog. Let  $L$  represent the collection of all log entries. Each entry  $l$  in  $L$  is a tuple containing elements like timestamp, process name, and execution path. The function ParseLog applies the defined regular expressions to each log entry  $l$  from  $L$ , converting it into a structured data format represented by  $d$  in the set  $D$ . This transformation can be formally expressed as:

$$\text{ParseLog}(l) : L \rightarrow D \quad (1)$$

This methodical approach allows us to efficiently convert unstructured log data into a format that is more suitable for analysis and further processing.

**Contextual data representation:** Our approach to detecting anomalous network nodes thrives in environments without predefined attack patterns, moving away from traditional supervised learning that relies on clear labels, which are often unavailable due to the lack of specific attack knowledge during training. We utilize a GCN designed to differentiate and comprehend the subtle behaviors of benign nodes. The core idea is that nodes involved in malicious activities exhibit distinct behavioral anomalies compared to benign ones, allowing our model to learn and identify unique patterns of relationships among node types. The feature extraction process involves an approach wherein each node's interactions and connections are translated into feature vectors. Thus, the contextual data representation phase is as follows. Our primary objective is to construct a provenance graph and extract features from the preprocessed logs. The provenance graph, denoted as  $G = (V, E)$ , is built based on the relationships identified in the logs. Here,  $V$  represents the set of vertices (entities such as files, processes, sockets), and  $E$  represents the set of edges (interactions like read, write, fork).

We define  $N_v$  as the total number of unique node types and  $N_e$  as the total number of unique edge types present in  $G$ . To facilitate the mapping of node and edge types to unique integers, we introduce two mapping functions:  $M_v : \text{Types} \rightarrow \{0, \dots, N_v - 1\}$  for nodes, and  $M_e : \text{Types} \rightarrow \{0, \dots, N_e - 1\}$  for edges.

Each node  $v \in V$  is assigned a type label using  $M_v$ , and its feature vector is determined by a function  $F : V \rightarrow \mathbb{N}^{2N_e}$ . The feature vector  $F(v)$  for a node  $v$  comprises the count of incoming and outgoing edges of each type. Specifically, for  $i \in \{0, \dots, N_e - 1\}$ , the feature vector includes:

$$a_i = |\{e \in E_{\text{in}}(v) : M_e(\text{type}(e)) = i\}| \quad (2)$$

$$a_{i+N_e} = |\{e \in E_{\text{out}}(v) : M_e(\text{type}(e)) = i\}| \quad (3)$$

where  $a_i$  represents the count of incoming edges of type  $i$ , and  $a_{i+N_e}$  denotes the count of outgoing edges of the same type. The final feature vector  $F(v)$  is represented in Equation 4:

$$F(v) = [a_0, a_1, \dots, a_{N_e-1}, a_{N_e}, a_{N_e+1}, \dots, a_{2N_e-1}] \quad (4)$$

**GNN training:** Our GCN model is engineered to predict complex node relationships within a graph, using node types and provenance graph structures as inputs. Its primary goal is to accurately predict feature vectors for each node by leveraging their connections and positions within the graph. The architecture features three graph convolutional layers, which aggregate and transform node features according to their connectivity. Each convolution operation is defined as:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}) \quad (5)$$

In this equation,  $H^{(l)}$  represents the node feature matrix at layer  $l$ ,  $\tilde{A}$  is the adjacency matrix of the graph,  $\tilde{D}$  is the degree matrix of  $\tilde{A}$ ,  $W^{(l)}$  is the weight matrix for layer  $l$ , and  $\sigma$  denotes the activation function, ReLU (Rectified Linear Unit). This convolutional operation effectively captures the local graph topology and feature information. Following the first two convolutional layers, batch normalization and ReLU activation functions are applied, with dropout used for regularization purposes. The model's predicted feature vector can be represented in Equation 6:

$$\hat{F}(v) = H^{(final)}W^{(final)} \quad (6)$$

Then, the Mean Squared Error (MSE) is used to assess the difference between the model's prediction and the actual feature vector  $F(v)$  for each node in the graph. Formally, the MSE loss is defined in Equation 7:

$$\text{MSE} = \frac{1}{n} \sum_{v \in V} |F(v) - \hat{F}(v)|^2 \quad (7)$$

Here,  $n$  represents the total number of nodes in the graph  $G$ ,  $F(v)$  denotes the actual feature vector of node  $v$ , and  $\hat{F}(v)$  is the feature vector predicted by the model for node  $v$ .

**Abnormal node identification:** In the final phase of our model, we focus on identifying abnormal nodes by analyzing deviations from expected behavior. The model determines whether a node is anomalous by calculating the error value based on the absolute difference between the predicted feature vector  $F_p(v)$  and the actual feature vector  $F(v)$ . The error score for node  $v$  is computed as follows:

$$\text{error}(v) = |\hat{F}(v) - F(v)| \quad (8)$$

By setting an appropriate threshold, we distinguish between normal node behavior and significant deviations indicative of anomalies. This threshold is determined using a data-driven approach, where we analyze the distribution of error values in the validation set to select a value that optimally balances accuracy with the minimization of false positives and negatives in anomaly detection.

### 3.3 Secure Model Training through FL and PHC

Our model employs a dual-key mechanism, entailing a public key for encryption and a private key for decryption, specifically used by the PHC. A specialized entity, referred to as the Key Generation Authority (KGA), undertakes the task of generating these keys. During the setup phase, this KGA chooses two distinct large prime numbers,  $p$  and  $q$ , ensuring  $p \neq q$  and the greatest common divisor (gcd) of  $pq$  and  $(p - 1)(q - 1)$  is 1. Subsequently, the KGA defines  $N = pq$  and computes Carmichael's function as  $\lambda(N) = \text{lcm}(p - 1, q - 1)$ . A random integer  $g$  is then chosen from  $\mathbb{Z}_{N^2}^*$  such that  $\text{gcd}(L(g^\lambda \bmod n^2), N) = 1$ . Here, the function  $L(x)$  is formulated as  $L(x) = \frac{x-1}{N}$ . The encryption public key is

thus represented as  $(N, g)$ , and the decryption private key is  $(\lambda, \mu)$ , where  $\mu$  is a distinct integer satisfying  $\mu = L(g^\lambda \bmod N^2)^{-1} \bmod N$ . These keys are disseminated among organizations while being kept hidden from the server.

As each federated communication cycle commences, organizations retrieve the global aggregated updates from the preceding round from the server, and update their local model parameters according to the Equation 9:

$$W_{t+1}^{old} = W_t^{old} + U_t^{global} \quad (9)$$

In this equation,  $W_{t+1}^{old}$  symbolizes the model updated for the round  $t + 1$ , with  $W_t^{old}$  and  $U_t^{global}$  representing the previous round's final organization model and aggregated updates, respectively. The organizations engage in local training on their datasets, developing the GNN model and calculating the new parameter vector  $W_{new}^{t+1}$ . The updated vector for the local model,  $U_{t+1}$ , intended for subsequent server transmission, is derived as  $U_{t+1} = W_{t+1}^{new} - W_{t+1}^{old}$ .

Following the reception of keys from the KGA, each organization, indicated as organization  $k$ , executes the PHC algorithm. This involves generating a random number  $r_{t+1}^k$  in  $Z_{n^2}^*$  where  $0 < r_{t+1}^k < n$ , converting the update vector of each organization into ciphertext, expressed as  $C_{t+1}^k = g^{U_{t+1}^k} \cdot r_{t+1}^k \bmod n^2$ .

Post-encryption, the organization sends the ciphertext  $C_{t+1}^k$  to the server for aggregation and generation of global model updates. Utilizing the PHC, the product of two ciphertexts results in a decryption equivalent to the sum of their original messages. The server, upon collecting all organization ciphertexts, applies the Federated Averaging (FedAvg) algorithm and PHC for aggregating updates. In the final phase, the server distributes the aggregated global updates back to the organizations. Each organization then employs the private keys to execute the decryption algorithm, obtaining the final results and updating the global model for the training round  $t + 2$ , as shown:

$$U_{(t+1)}^{global} = L \left( \left( C_{(t+1)}^{global} \right)^\lambda \bmod n^2 \right) \mu \bmod n \quad (10)$$

$$W_{t+2}^{old} = W_{t+1}^{old} + U_{t+1}^{global} \quad (11)$$

Here,  $W_{t+2}^{old}$  is the updated global model at the commencement of round  $t + 2$ . This iterative process continues until the global model achieves convergence.

## 4 Performance Evaluation

### 4.1 Experimental Setup

We used PyTorch 2.0.1 and CUDA 11.8 with an Intel i7-13700F CPU and NVIDIA GeForce RTX 4070 GPU. Our dataset's training set (60%) contains only normal data, while the validation and test sets (20% each) include both normal and anomalous data, facilitating effective model evaluation and threshold tuning. The model, configured with a 0.5 learning rate and an Adam optimizer, underwent training across three rounds, each involving 100 epochs per

organization. We tested scalability by varying the number of participating organizations from 2 to 10, observing stable performance with minimal fluctuations. Data encryption was managed using the python-paillier library with a 1024-bit key [5]. The architecture of our GNN model is detailed in Table 1.

Table 1: Parameters of the GNN Model

Layer	Input Dimension	Output Dimension
GCNConv1	9	256
BatchNorm1	-	256
GCNConv2	256	128
BatchNorm2	-	128
GCNConv3 (Output Layer)	128	23

## 4.2 Dataset

The dataset for this study came from the DARPA TCE3 program, during red-team versus blue-team adversarial engagements in April 2018. These engagements, designed to simulate diverse cyberattack scenarios, involved the red team conducting attacks and the blue team defending. The setup included multiple servers such as web, SSH, email, and SMB, replicating a real-world network. The scenario realism was further enhanced by simulating extensive normal user activities, including routine system administration, web browsing, and the installation of various tools. The threat descriptions and ground truth reports from DARPA provided key insights into the red team’s tactics and strategies [10]. Despite the dataset’s large volume of events, malicious events represent only a small fraction, making anomaly and threat detection challenging due to the significant imbalance between benign and malicious activities [3].

## 4.3 Evaluation Metrics

To evaluate our model, we use several metrics based on the definitions of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Accuracy is the proportion of correct predictions ( $\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$ ), Precision is the ratio of correct attack identifications to all labeled attacks ( $\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$ ), and Recall is the proportion of actual attacks correctly identified ( $\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$ ). The F1-score provides a balance between precision and recall ( $\text{F1-score} = 2 \cdot \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}$ ), and False Positive Rate measures the rate of benign instances incorrectly marked as attacks ( $\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$ ).

## 4.4 Detection Evaluation

The results of our proposed model and the performance comparison with other anomaly detection models are presented in Table 2. Our approach stands out

in its unsupervised learning methodology, which enables the detection of APT attacks without prior exposure to malicious data. The model exhibits a high accuracy of 0.93 and maintains a low false positive rate of 0.06. This performance is particularly noteworthy as it is achieved in the absence of supervised learning techniques commonly employed in existing models. Due to the lack of existing research on unsupervised learning methods applied to this dataset, this study undertakes a comparative analysis with supervised learning techniques. Our model’s performance was benchmarked against several state-of-the-art models, including DeepTaskAPT [18], Deeplog [6], LogShield, and BERT [1], and conventional baseline models (such as Random Forest and Linear Regression) [18]. While some of these methods may demonstrate superior performance metrics in certain aspects, they are predominantly reliant on supervised learning models. This reliance is a significant limitation, especially in addressing zero-day challenges where the model has no prior knowledge of attack patterns. Moreover, these existing models overlook the critical aspect of privacy in log data. In contrast, our model integrates FL and homomorphic encryption, addressing privacy concerns more effectively.

Table 2: Comparison of Model Performance

Model	Method	Acc	Prec	Rec	F1-score	FPR
P3GNN	Unsupervised	0.93	0.73	0.91	0.81	0.06
DeepTaskAPT	Supervised	0.985	N/A	N/A	N/A	0.011
DeepLog	Supervised	0.83	N/A	N/A	N/A	0.16
Random Forest	Supervised	0.80	N/A	N/A	N/A	0.18
Linear Regression	Supervised	0.08	N/A	N/A	N/A	0.93
LogShield	Supervised	0.98	0.99	0.99	0.984	N/A
BERT	Supervised	0.89	0.91	0.95	0.90	N/A

#### 4.5 Computational Complexity Analysis of P3GNN

The complexity of the P3GNN model arises from data extraction, graph construction, GNN training, and anomaly detection. Incorporating FL and PHC adds to this complexity by introducing encryption and decryption overheads, crucial for ensuring data privacy in distributed networks. The base complexity of the P3GNN model, without FL and PHC, depends on the log data volume, the graph’s size and structure (vertices  $V$  and edges  $E$ ), and the depth of computational requirements for GNN training. These factors contribute to a largely linear complexity, scaling with the size of the data and the graph. However, the model’s efficiency in this configuration benefits from direct, unencrypted operations on data and parameters. In contrast, when FL and PHC are integrated into the model, each operation involving model parameters—especially during the GNN training phase—incurs additional complexity due to encryption and

decryption. This complexity is polynomial, specifically  $O(n_{\text{params}} \cdot k^3)$ , where  $n_{\text{params}}$  is the number of model parameters, and  $k$  represents the encryption key size. This added complexity is a direct consequence of the cryptographic processes required to secure the model’s distributed learning environment. Table 3

Table 3: Comparison of Computational Complexity

Component	Without FL and PHC	With FL and PHC
Data Extraction	$O(n_{\text{logs}} \cdot c)$	$O(n_{\text{logs}} \cdot c)$
Contextual Data Rep.	$O(n_{\text{vertices}} + n_{\text{edges}})$	$O(n_{\text{vertices}} + n_{\text{edges}}) + O(n_{\text{params}} \cdot k^3)$
GNN Training	$O( V  +  E )$ per layer	$O( V  +  E )$ per layer + $O(n_{\text{params}} \cdot k^3)$
Abnormal Node Ident.	$O(n_{\text{vertices}})$	$O(n_{\text{vertices}}) + O(n_{\text{params}} \cdot k^3)$

summarizes the computational complexity of P3GNN with and without privacy preservation techniques. The added computational complexity from integrating FL and PHC into the P3GNN model is a necessary trade-off for ensuring robust privacy and security in distributed ML environments. This complexity enables collaborative training across multiple entities without exposing sensitive data, aligning with stringent data protection regulations and safeguarding against unauthorized access, making it indispensable for securely handling sensitive or confidential information.

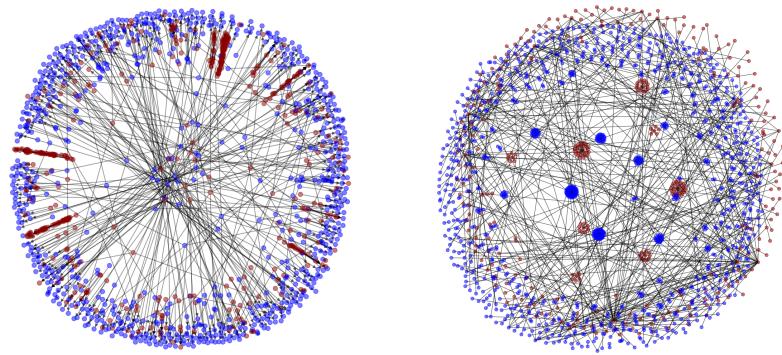
#### 4.6 Interpretability of P3GNN

This section delves into the interpretability of our model, particularly focusing on its capability to detect attacks at the node level and trace the entirety of an anomalous neighborhood involved in an attack. By leveraging GNNs, P3GNN goes beyond mere identification, enabling the dissection of attack patterns through node-level analysis.

**Visualization and Node-Level Analysis:** P3GNN stands out for its ability to precisely identify anomalous nodes within a network. Figure 3a emphasizes the paths and sequences formed by anomalous nodes. The flow of malicious activities (red) amidst the benign (blue) nodes reveals the model’s detection of attack trajectories, which can be critical for understanding the sequential progression and strategies of the attackers. This visualization through the capabilities of GNNs enhances interpretability, and is particularly useful for path analysis, as it enables security analysts to track potential attack trajectories and comprehend how malicious activities may propagate. It is an essential tool for formulating effective defensive strategies.

Additionally, P3GNN highlights clusters of malicious nodes that can indicate the initial breach points or critical attack vectors. Figure 3b illustrates a subgraph extracted from the overall network, highlighting the prominent clusters of malicious and benign nodes. In our network visualizations, clusters of

nodes emerge organically due to the inherent structure and connectivity patterns within the network data. These clusters, emerging as a result of the interconnected nature of network activities, are marked by dense interconnectivity, which is a hallmark of coordinated or related malicious activities. Such clusters demand urgent investigative action. In this way, security analysts are empowered to conduct focused investigations and implement targeted security measures.



(a) Pathways and sequences of activities (b) Clustering of anomalous nodes

Fig. 3: Detailed view of anomalous node interconnectivity

**Case Study: Nginx Backdoor w/ Drakon In-Memory:** To further illustrate the model’s interpretability, we analyze a specific attack incident that occurred within the DARPA network infrastructure. In this incident, the attacker targeted the system (Cadet’s data), initially exploiting a vulnerability in the Nginx server using a malformed HTTP request. This led to the execution of a Drakon implant in the Nginx server’s memory, establishing a shell connection to the attacker’s console through HTTP. The attacker’s subsequent actions involved multiple attempts to download and elevate privileges for a micro APT implant. Despite repeated failures in privilege elevation, the attacker managed to execute the Drakon implant with root privileges and later conducted a port scan on multiple network targets to assess vulnerabilities, leaving the console connection open throughout. This attack sequence highlights several critical steps:

- Initial exploitation of Nginx and Drakon implant execution.
- Repeated attempts to download and elevate the micro APT implant.
- Successful execution of the Drakon implant with elevated privileges.
- Network reconnaissance through port scanning.

The interpretability of our model is rooted in its ability to not just identify the occurrence of an attack but to unravel the sequence of events constituting it. When an anomalous activity is identified, the model raises alarms for these specific nodes, signaling potential security breaches. To comprehensively map out the attack's progression, we employ a depth-first search algorithm on the neighborhood of the identified anomalous nodes. In the DARPA attack case, the model successfully pinpointed each critical step, providing a comprehensive view of the attack's progression. Figure 4 presents a graphical representation of the attack steps as identified by our model.

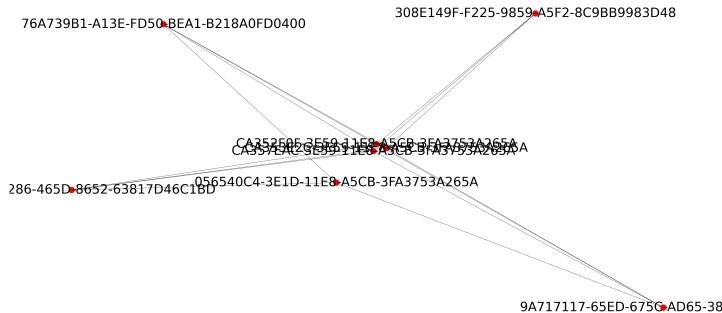


Fig. 4: Visualization of attack sequence

Our model's detection ability is further demonstrated by correlating discrete log entries with corresponding graph interactions. By examining the flow of UUIDs within the network and mapping them to recorded events, as shown in table 4, our system constructs a narrative that aligns with known attack patterns. For instance, during the initial compromise phase, our model detected the event 'EVENT\_READ' and 'EVENT\_EXECUTE' in conjunction with specific UUID interactions, implying unauthorized access and potential credential compromise. As the attack transitioned to establish a foothold, modifications in process and principal changes were captured, highlighting the attacker's efforts to gain a stronger grip within the system.

Subsequently, an escalation of privileges was inferred from principal changes logged, suggesting that the attacker was attempting to gain higher-level access. This was further corroborated by the event 'EVENT\_CHANGE\_PRINCIPAL' linked to UUIDs accessing sensitive system files. Our model's depth-first search illuminated these steps in the context of the larger attack framework, showcasing the depth of our analysis. In the final phases, internal reconnaissance and efforts to maintain presence were identified through repeated 'EVENT\_READ' and 'EVENT\_EXECUTE' actions. These steps were pivotal in the attacker's strategy to gather intelligence and ensure long-term access to the compromised system. By integrating log analysis with graph theory, our model transcends

traditional detection mechanisms, offering a nuanced and dynamic approach to cybersecurity. This case study serves as a testament to the model's efficacy in recognizing complex attack patterns and the associated network behaviors.

Table 4: Correlation of Attack Phases and Log Entries with Graph Interactions

Attack Phase	Log Entry Details	Graph Interaction (UUIDs)
Initial Compromise	EVENT_READ, EVENT_EXECUTE on /etc/login.conf, /etc/group	CA352F0F-3E59-11E8-A5CB-3FA3753A265A → 9A717117-65ED-675C-AD65-38102C67C832, CA357EAC-3E59-11E8-A5CB-3FA3753A265A → 308E149F-F225-9859-A5F2-8C9BB9983D48
Establish Foothold	EVENT_MODIFY_PROCESS, EVENT_CHANGE_PRINCIPAL	CA353E2C-3E59-11E8-A5CB-3FA3753A265A → 24782D99-5286-465D-8652-63817D46C1BD, CA353E2C-3E59-11E8-A5CB-3FA3753A265A → 76A739B1-A13E-FD50-BEA1-B218A0FD0400
Escalate Privileges	EVENT_CHANGE_PRINCIPAL indicating privilege escalation	CA353E2C-3E59-11E8-A5CB-3FA3753A265A → 308E149F-F225-9859-A5F2-8C9BB9983D48
Internal Reconnaissance	EVENT_READ on various system files	CA357EAC-3E59-11E8-A5CB-3FA3753A265A → 9A717117-65ED-675C-AD65-38102C67C832
Maintain Presence	Repeated EVENT_READ, EVENT_EXECUTE	CA352F0F-3E59-11E8-A5CB-3FA3753A265A → Various UUIDs

## 5 Conclusion

In conclusion, this paper introduced the P3GNN, a provenance graph-based model for APT detection in SDNs that integrates GCN with FL and homomorphic encryption for privacy preservation. The model effectively detects APTs without prior attack knowledge, maintaining data privacy and integrity. Our tests using the DARPA TC dataset show robust performance, achieving an accuracy of 0.93 and a false positive rate of 0.06. This indicates a significant improvement over traditional supervised learning methods, which often struggle with zero-day attacks and privacy concerns. Furthermore, the model's capability to pinpoint anomalous nodes and trace the entire attack trajectory offers invaluable insights for security analysis, addressing a critical need in AI for interpretability. In future work, we plan to broaden our study by exploring our approach's applicability across different domains and scenarios, with a focus on empirical validation in these contexts. A key extension will involve adapting our method to defend against APT attacks across multiple hosts by shifting from host-level to network-level monitoring, enabling the detection of lateral movements in APT scenarios. Additionally, we plan to incorporate real-time data processing into the P3GNN model, enhancing its effectiveness in dynamic network environments and enabling a faster response to emerging threats. Additionally, we plan to automate the generation of Cyber Threat Intelligence reports using our P3GNN model to streamline the documentation of attack details and mitigation strategies. This automation will utilize provenance data to improve the efficiency and consistency of threat reporting and analysis in cybersecurity.

## References

1. Afnan, S., Sadia, M., Iqbal, S., Iqbal, A.: Logshield: A transformer-based apt detection system leveraging self-attention (2023)
2. Alshamrani, A., Myneni, S., Chowdhary, A., Huang, D.: A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Communications Surveys & Tutorials* **21**(2), 1851–1877 (2019). <https://doi.org/10.1109/COMST.2019.2891891>
3. Anjum, M.M., Iqbal, S., Hamelin, B.: Analyzing the usefulness of the darpa optc dataset in cyber threat detection research. In: Proceedings of the 26th ACM Symposium on Access Control Models and Technologies. p. 27–32. SACMAT ’21, Association for Computing Machinery, New York, NY, USA (2021). <https://doi.org/10.1145/3450569.3463573>, <https://doi.org/10.1145/3450569.3463573>
4. Dabbagh, M., Hamdaoui, B., Guizani, M., Rayes, A.: Software-defined networking security: pros and cons. *IEEE Communications Magazine* **53**(6), 73–79 (2015). <https://doi.org/10.1109/MCOM.2015.7120048>
5. Data61, C.: Python paillier library. <https://github.com/data61/python-paillier> (2013)
6. Du, M., Li, F., Zheng, G., Srikumar, V.: Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1285–1298 (2017)
7. Ghiasvand, S., Ciorba, F.M.: Anonymization of system logs for privacy and storage benefits. *CoRR* **abs/1706.04337** (2017), <http://arxiv.org/abs/1706.04337>
8. Han, X., Pasquier, T., Bates, A., Mickens, J., Seltzer, M.: Unicorn: Runtime provenance-based detector for advanced persistent threats. *arXiv preprint arXiv:2001.01525* (2020)
9. Jagarlamudi, G.K., Yazdinejad, A., Parizi, R.M., Pouriyeh, S.: Exploring privacy measurement in federated learning. *The Journal of Supercomputing* pp. 1–41 (2023)
10. Keromytis, A.D.: Transparent computing engagement 3 data release. README-E3. md (2018)
11. Konečný, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: Strategies for improving communication efficiency (2017)
12. Lara, A., Kolasani, A., Ramamurthy, B.: Network innovation using openflow: A survey. *IEEE Communications Surveys & Tutorials* **16**(1), 493–512 (2014). <https://doi.org/10.1109/SURV.2013.081313.00105>
13. Latif, Z., Sharif, K., Li, F., Karim, M.M., Biswas, S., Wang, Y.: A comprehensive survey of interface protocols for software defined networks. *Journal of Network and Computer Applications* **156**, 102563 (2020). <https://doi.org/https://doi.org/10.1016/j.jnca.2020.102563>, <https://www.sciencedirect.com/science/article/pii/S1084804520300370>
14. Liang, H., Li, Y., Zhang, C., Liu, X., Zhu, L.: Egia: An external gradient inversion attack in federated learning. *IEEE Transactions on Information Forensics and Security* (2023)
15. Liang, Y., Li, S., Yan, C., Li, M., Jiang, C.: Explaining the black-box model: A survey of local interpretation methods for deep neural networks. *Neurocomputing* **419**, 168–182 (2021). <https://doi.org/https://doi.org/10.1016/j.neucom.2020.08.011>, <https://www.sciencedirect.com/science/article/pii/S0925231220312716>

16. Lv, Y., Qin, S., Zhu, Z., Yu, Z., Li, S., Han, W.: A review of provenance graph based apt attack detection:applications and developments. In: 2022 7th IEEE International Conference on Data Science in Cyberspace (DSC). pp. 498–505 (2022). <https://doi.org/10.1109/DSC55868.2022.00075>
17. Maggi, F., Matteucci, M., Zanero, S.: Detecting intrusions through system call sequence and argument analysis. *IEEE Transactions on Dependable and Secure Computing* **7**(4), 381–395 (2010). <https://doi.org/10.1109/TDSC.2008.69>
18. Mamun, M., Shi, K.: Deeptaskapt: Insider apt detection using task-tree based deep learning. In: 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). pp. 693–700 (2021). <https://doi.org/10.1109/TrustCom53373.2021.00102>
19. Manzoor, E., Milajerdi, S.M., Akoglu, L.: Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1035–1044 (2016)
20. Milajerdi, S.M., Eshete, B., Gjomemo, R., Venkatakrishnan, V.: Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. p. 1795–1812. CCS ’19, Association for Computing Machinery, New York, NY, USA (2019). <https://doi.org/10.1145/3319535.3363217>, <https://doi.org/10.1145/3319535.3363217>
21. Milajerdi, S.M., Gjomemo, R., Eshete, B., Sekar, R., Venkatakrishnan, V.: Holmes: Real-time apt detection through correlation of suspicious information flows. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 1137–1152 (2019). <https://doi.org/10.1109/SP.2019.00026>
22. Pasquier, T., Han, X., Goldstein, M., Moyer, T., Eyers, D., Seltzer, M., Bacon, J.: Practical whole-system provenance capture. In: Proceedings of the 2017 Symposium on Cloud Computing. pp. 405–418 (2017)
23. Rabieinejad, E., Yazdinejad, A., Dehghantanha, A., Srivastava, G.: Two-level privacy-preserving framework: Federated learning for attack detection in the consumer internet of things. *IEEE Transactions on Consumer Electronics* (2024)
24. Sakhnini, J., Karimipour, H., Dehghantanha, A., Yazdinejad, A., Gadekallu, T.R., Victor, N., Islam, A.: A generalizable deep neural network method for detecting attacks in industrial cyber-physical systems. *IEEE Systems Journal* (2023)
25. Shan-Shan, J., Ya-Bin, X.: The apt detection method in sdn. In: 2017 3rd IEEE International Conference on Computer and Communications (ICCC). pp. 1240–1245. IEEE (2017)
26. Shan-Shan, J., Ya-Bin, X.: The apt detection method based on attack tree for sdn. In: Proceedings of the 2nd International Conference on Cryptography, Security and Privacy. p. 116–121. ICCSP 2018, Association for Computing Machinery, New York, NY, USA (2018). <https://doi.org/10.1145/3199478.3199481>, <https://doi.org/10.1145/3199478.3199481>
27. Shokri, R., Shmatikov, V.: Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. pp. 1310–1321 (2015)
28. Thi, H.T., Hoang Son, N.D., Duy, P.T., Pham, V.H.: Federated learning-based cyber threat hunting for apt attack detection in sdn-enabled networks. In: 2022 21st International Symposium on Communications and Information Technologies (ISCIT). pp. 1–6 (2022). <https://doi.org/10.1109/ISCIT55906.2022.9931222>

29. Thi, H.T., Son, N.D.H., Duy, P.T., Khoa, N.H., Ngo-Khanh, K., Pham, V.H.: Xfedhunter: An explainable federated learning framework for advanced persistent threat detection in sdn (2023)
30. Wang, Q., Hassan, W.U., Li, D., Jee, K., Yu, X., Zou, K., Rhee, J., Chen, Z., Cheng, W., Gunter, C.A., et al.: You are what you do: Hunting stealthy malware via data provenance analysis. In: NDSS (2020)
31. Wu, Y., Xie, Y., Liao, X., Zhou, P., Feng, D., Wu, L., Li, X., Wildani, A., Long, D.: Paradise: real-time, generalized, and distributed provenance-based intrusion detection. IEEE Transactions on Dependable and Secure Computing **20**(2), 1624–1640 (2022)
32. Xia, W., Wen, Y., Foh, C.H., Niyato, D., Xie, H.: A survey on software-defined networking. IEEE Communications Surveys & Tutorials **17**(1), 27–51 (2014)
33. Xie, Y., Feng, D., Hu, Y., Li, Y., Sample, S., Long, D.: Pagoda: A hybrid approach to enable efficient real-time provenance based intrusion detection in big data environments. IEEE Transactions on Dependable and Secure Computing **17**(6), 1283–1296 (2020). <https://doi.org/10.1109/TDSC.2018.2867595>
34. Yazdinejad, A., Bohloli, A., Jamshidi, K.: Efficient design and hardware implementation of the openflow v1. 3 switch on the virtex-6 fpga ml605. The Journal of Supercomputing **74**, 1299–1320 (2018)
35. Yazdinejad, A., Dehghantanha, A., Karimipour, H., Srivastava, G., Parizi, R.M.: An efficient packet parser architecture for software-defined 5g networks. Physical Communication **53**, 101677 (2022)
36. Yazdinejad, A., Parizi, R.M., Dehghantanha, A., Zhang, Q., Choo, K.K.R.: An energy-efficient sdn controller architecture for iot networks with blockchain-based security. IEEE Transactions on Services Computing **13**(4), 625–638 (2020)
37. Yazdinejadna, A., Parizi, R.M., Dehghantanha, A., Khan, M.S.: A kangaroo-based intrusion detection system on software-defined networks. Computer Networks **184**, 107688 (2021). <https://doi.org/https://doi.org/10.1016/j.comnet.2020.107688>, <https://www.sciencedirect.com/science/article/pii/S1389128620312949>