IEEE Open Journal of the Computer Society

Received 17 July 2023; accepted 3 September 2023. Date of publication 5 September 2023;
date of current version 18 September 2023. The review of this article was arranged by Associate Editor Quan Chen.

Digital Object Identifier 10.1109/OJCS.2023.3312299

MetaCIDS: Privacy-Preserving Collaborative Intrusion Detection for Metaverse based on Blockchain and Online Federated Learning

VU TUAN TRUONG AND LONG BAO LE

Institut National de la Recherche Scientifique, University of Québec, Montréal, QC H5A 1K6, Canada

CORRESPONDING AUTHOR: LONG BAO LE (email: long.le@inrs.ca)

This work was supported in part by funding from the Innovation for Defence Excellence and Security (IDEaS) program from the Department of National Defence (DND).

ABSTRACT Metaverse is expected to rely on massive Internet of Things (IoT) connections so it inherits various security threats from the IoT network and also faces other sophisticated attacks related to virtual reality technology. As traditional security approaches show various limitations in the large-scale distributed metaverse, this paper proposes MetaCIDS, a novel collaborative intrusion detection (CID) framework that leverages metaverse devices to collaboratively protect the metaverse. In MetaCIDS, a federated learning (FL) scheme based on unsupervised autoencoder and an attention-based supervised classifier enables metaverse users to train a CID model using their local network data, while the blockchain network allows metaverse users to train a machine learning (ML) model to detect intrusion network flows over their monitored local network traffic, then submit verifiable intrusion alerts to the blockchain to earn metaverse tokens. Security analysis shows that MetaCIDS can efficiently detect zero-day attacks, while the training process is resistant to SPoF, data tampering, and up to 33% poisoning nodes. Performance evaluation illustrates the efficiency of MetaCIDS with 96% to 99% detection accuracy on four different network intrusion datasets, supporting both multi-class detection using labeled data and anomaly detection trained on unlabeled data.

INDEX TERMS Blockchain, collaborative intrusion detection, federated learning, metaverse, semi-supervised learning.

I. INTRODUCTION

Metaverse is emerging as the next-generation Internet, which is empowered by a variety of advanced technologies including the Internet of Things (IoT), blockchain, digital twins (DT), augmented/virtual reality (AR/VR), artificial intelligence (AI), 5G/6G wireless networks, and edge/cloud computing. It realizes a virtual world where users take part in via a digital representative called avatar, then immerse themselves into the 3D virtual environment using AR/VR wearable devices with a wide range of virtual activities and services such as working, playing, education, healthcare, and social services [1]. This virtual world synchronizes and operates in parallel with the physical world thanks to the virtual-physical mapping process of DT, in which real-world events are continuously reflected into the virtual world, while any changes in the virtual world are also synchronized to its physical counterpart.

To realize the virtual-physical synchronization, real-world data must be collected seamlessly by numerous IoT devices, sensors, and unmanned aerial vehicles (UAVs) to build the virtual environment, while user's personal data such as appearance, facial expression, voice, digital footprint, and behaviors are collected by AR/VR wearable devices to construct the digital avatars. Consequently, different security techniques must be deployed in the metaverse to protect these sensitive data from malicious actors. For example, access control mechanisms can prevent attackers from illegitimately accessing metaverse computational and data resource, while identity authentication systems can filter out sybil entities and proactively hinder phishing scam, impersonation, and other social engineering attacks in the metaverse.

Among various security solutions, intrusion detection system (IDS) [2] provides efficient countermeasures to deal with

a wide variety of threats in the metaverse such as denial of service (DoS), distributed denial of service (DDoS), botnet, malware, and malicious data injection. In terms of IDS structure, there are two main types of IDS which are network intrusion detection system (NIDS) and host-based intrusion detection system (HIDS). While NIDS monitors and analyzes network traffic being transmitted to a local metaverse network to recognize suspicious activity, HIDS programs are installed in individual metaverse host/device to detect abnormal behavior targeting on the host that they are serving.

However, as a centralized system, they both face various challenges when being adopted into the metaverse. Firstly, the centralized NIDS/HIDS server might be attacked, or they could act dishonestly to earn illegal benefit, thus manipulating the system and causing single point of failure (SPoF). Secondly, they suffer from the scalability issue as the number of metaverse devices is very large, e.g., millions or even billions of nodes. As a result, the NIDS server must process massive volume of metaverse network traffic, while the HIDS program must be installed and managed on numerous metaverse hosts and devices, making it exceeds the capacity of the centralized IDS. Thirdly, attackers can interfere in IDS storage or transmission to modify the detection results and logs, thus threatening the integrity of the system.

To tackle these problems, this paper proposes MetaCIDS, a CID framework for the metaverse that utilizes blockchain to decentralize the process of intrusion detection management. Instead of relying on a central authority, MetaCIDS is maintained by a blockchain network with multiple nodes taking responsibility for alert verification and decision making, thereby eliminating SPoF and manipulation, while improving the scalability enormously. Besides, detection results and logs are recorded on-chain with blockchain's immutable property to ensure the integrity of detection data.

In terms of intrusion detection approaches, although machine learning (ML) models have shown outstanding performance compared to traditional IDS techniques [3], the centralized training scheme is vulnerable to privacy risks since network data of metaverse users must be collected in a central server for training the model. Federated learning (FL) [4] has emerged as a feasible solution to address this issue, which allows local devices to train the model locally on their own collected data, then submit only the gradient updates to the central server for aggregation without revealing the data. However, there remains certain challenges if the conventional FL scheme is applied to train an intrusion detection model for the metaverse. Firstly, most metaverse users could not label their collected network data to train the model, since they, without expertise in cybersecurity, cannot distinguish between normal and malicious network flows. Secondly, the aggregation process is carried out by a central server, thus posing various centralization-related risks such as SPoF, tampering, privacy leakage, and manipulation. Thirdly, metaverse users lack necessary motivation to contribute their computational resource to collect data and train the FL model.

To tackle the first issue of labeling data, MetaCIDS deploys two different detection modules: i) a multi-class intrusion classifier trained on labeled data to detect attacks that have been previously seen in the training data based on the attention technique [5]; ii) an anomaly detection module trained on unlabeled data to detect attacks unseen in the training data (i.e., zero-day attacks) based on deep autoencoder (DAE). For the second problem of centralization, MetaCIDS leverages the blockchain network to decentralize the aggregation process, making it fair and stable even if there are certain nodes acting maliciously or under attacks. The third issue is mitigated by a concrete incentive and reputation mechanism that encourages metaverse users to contribute their available resources to protect the metaverse and correspondingly earn blockchain-based token rewards.

A. RELATED WORKS AND RESEARCH GAPS

Prior to our work, the application of IDS for the metaverse has not been well investigated. Outside of the metaverse context, there has been several works exploiting the following related use cases: i) the use of blockchain for CID management [6]; ii) centralized FL and blockchain-based decentralized FL for training IDS models [7], [8]. This section presents the state-of-the-art related works, thereby exploring the research gaps that MetaCIDS aims to address.

1) IDS IN METAVERSE

Regarding IDS for the metaverse, the authors in [9] proposed a GAN-based IDS model which consists of several ML techniques such as generative adversarial network (GAN) for data optimization, DAE for dimension reduction, and random forest (RF) for classification. Although the proposed architecture achieves high performance on the InSDN dataset, its complexity might be a burden hindering it from practical use cases for the metaverse. Instead of relying on ML models, the authors in [10] used the sequential probability ratio test (SPRT) method to detect wormhole attack within the metaverse context. The work is only limited to wormhole attack, while other severe threats are not considered. Moreover, both studies only concentrate on designing a detection model, while the process of detection management is not investigated. As mentioned above, these centralized schemes might be vulnerable to various centralization-related issues when being adopted into the metaverse.

To overcome these problems, our previous work [11] has been designed for both distributed training process and decentralized intrusion detection management (e.g., alert submission/verification and incentive, reputation mechanisms). However, the work does not support training the IDS model on unlabeled data, making it impractical in the real metaverse system. Without training on newly collected unlabeled data, the framework cannot detect zero-day attacks. Moreover, when metaverse users submit an intrusion alert, they must also provide the network flow associated with the alert to verify its integrity and prevent the DDoS attack. This might cause

privacy concern as the local data must be disclosed to other participants. In fact, this work is an extension of our previous framework [11], which fills all the mentioned gaps thanks to a novel semi-supervised model with an encoding mechanism to protect data privacy. Besides, we also supplement intensive experimental results and analysis to show the efficiency of MetaCIDS compared to its predecessor.

2) BLOCKCHAIN FOR IDS

The authors in [12] proposed an IDS platform to enable secure data transfer for the Internet of drones, in which blockchain and zero-knowledge proof empower the registration and verification process with decentralization and privacy preservation. The authors in [13] utilized the long short-term memory (LSTM) to design a high-performance IDS model, while blockchain is used to store and share the detection results. However, both frameworks did not design any incentive mechanism to motivate the participation of users and prevent them from submitting dishonest alerts. The authors in [14] not only used blockchain for storing detection results, but also designed a blockchain-based incentive mechanism for their CID framework in multi-microgrid (MMG) systems. As a result, the collaborative scheme helps reducing false negative rate (FNR), while eliminating SPoF in data storage. The authors in [15] proposed a CID architecture that attempts to improve user privacy and the integrity of detection via blockchain-based storage and encryption techniques. Dishonest alerts are stated to be filtered out in the consensus process of the blockchain network. However, this only provides a generic architecture at a conceptual level without practical or detailed system design.

Furthermore, all of the above blockchain-based CID frameworks have not taken into account the training process of IDS models. Consequently, these centralized training schemes are prone to data leakage, SPoF, and model tampering, while their pre-trained model is not able to detect zero-day attacks that are unseen in the fixed training dataset.

3) FEDERATED LEARNING FOR IDS

There is a wide range of studies exploring the use of FL-based IDS in the IoT and industrial IoT (IIoT) sectors [16], [17], [18], [19], [20], [21], [22]. For example, the authors in [17] used gated recurrent unit (GRU) combined with convolutional neural network (CNN) to design an IDS model for cyber-physical systems (CPSs), while FL enables a privacy-preserving training scheme. Similarly, the work in [20] also utilized FL for the training process, while the ML model is a combination of LTSM, GRU, and DT. However, most IoT devices, in practice, could not label their collected network data to train the IDS model, thus posing significant challenges to apply the frameworks in the above works. To mitigate the labeling issue, the studies in [19] and [22] offer semi-supervised FL designs based on AE and knowledge distillation, respectively. The main idea is to utilize unlabeled data to train an encoder module that extracts only important features of data, then feed the optimized data into a classifier to detect intrusion. To maximize privacy in training IDS models, the authors in [21] supplement differential privacy (DP) [26] techniques in the FL training process to prevent an inference attack (i.e., the central server infers sensitive information from the submitted gradient updates).

Nevertheless, all above FL-based IDS frameworks implement the aggregation process via a single central server, making it vulnerable to SPoF, manipulation, and poisoning attack (i.e., certain local devices send dishonest gradient updates to the server). Furthermore, the scope of these frameworks is only limited to training an IDS model, while the detection management process is not taken into account.

4) BLOCKCHAIN-BASED FEDERATED LEARNING FOR IDS

To make FL more secure and stable during the training process of IDS models, several works [23], [24], [25] integrated blockchain into the FL training process. For instance, the authors in [24] designed an FL-based CID framework for UAV networks that uses blockchain to store and share the training models, thereby enhancing the security and integrity of the IDS models. Although blockchain is used in training, the aggregation process is still centralized in [24]. The two works [23] and [25] proposed to decentralize the aggregation process by blockchain-based consensus mechanisms. Specifically, instead of relying on a central server, multiple consensus nodes (e.g., miners) take responsibility for aggregating local gradient updates into a global model. Thereby, the global model is aggregated properly even if certain hosts are malicious or under attacks.

Although decentralization is supported in these frameworks, none of them offers a proper incentive mechanism to encourage users to train the IDS model. The frameworks could not exploit unlabeled data, while the zero-day attack is still an unsolved challenge. Also, the authors only focus on training FL models without considering CID management.

B. MOTIVATIONS AND KEY CONTRIBUTIONS

Our goal is to design a decentralized metaverse CID framework that fills the presented research gaps. The novel contributions achieved by MetaCIDS compared to existing frameworks are illustrated in Table 1, which can be summarized as follows:

- MetaCIDS offers a comprehensive training scheme that enables privacy preservation and resists to inference attack thanks to FL and DP technique. The FL aggregation process is decentralized by blockchain to ensure model integrity and resistance to SPoF and poisoning attack.
- A semi-supervised model is designed based on the DAE and attention technique that utilizes both labeled and unlabeled data. Thereby, MetaCIDS offers both multi-class intrusion detection and zero-day attack detection.

TABLE 1. Comparison Between MetaCIDS and Other Existing Frameworks Regarding Intrusion Detection

Utilized techniques	Blockchain				Federated Learning (FL)							Blockchain and FL			
Reference	[14]	[13]	[12]	[15]	[16]	[17]	[18]	[19]	[20]	[21]	[22]	[23]	[24]	[25]	Ours
Privacy in training model	The purpose is mainly for storing detection results, while the training process is completely centralized				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Zero-day attack scenario					✓		✓	✓			✓				✓
Support unlabeled data					✓			✓			✓				✓
Inference attack										✓			✓		✓
SPoF in model aggregation					The model is aggregated in a centralized server, thus prone to SPoF, manipulation, and poisoning attack							✓		✓	✓
Integrity of IDS model												✓	✓	✓	✓
Poisoning resistance												✓		✓	✓
Collaborative detection	✓	✓		✓	The purpose is only limited to training a machine learning model for IDS, without involving the process of intrusion detection management										✓
Privacy in detection				✓											✓
Fake alert filtering				✓											✓
Incentive mechanism	✓														✓
Reputation system															✓
Integrity of detection result	✓	✓	✓	✓											✓

- MetaCIDS provides a complete workflow for the process of intrusion detection management based on blockchain and smart contracts. Metaverse users can download the IDS model from blockchain, use it to collaboratively detect intrusion, then submit alerts to the blockchain.
- A concrete incentive mechanism and reputation system are designed to motivate metaverse users to contribute their available resource to protect the metaverse, while eliminating malicious actors and sybil avatars.

The rest of the paper is structured as follows. Section II presents the system model with threat formulation and an overview of the MetaCIDS architecture. Section III proposes the detailed design of MetaCIDS, including the semi-supervised IDS model, FL-based training, and blockchain-empowered CID management. Section IV analyzes the security and performance of MetaCIDS with various experiments, while Section V concludes the paper.

II. SYSTEM AND THREAT MODELS

In this section, we first present the system model, threat model, and our design goal. Then, we provide an overview of MetaCIDS whose details are given in the following section.

A. SYSTEM MODEL

The metaverse is based on the deployment of a massive number of devices in different types and capacities (e.g., IoT sensors, AR/VR devices, wearable haptic devices, smart vehicles, and edge/cloud servers) as shown in Fig. 1. These devices frequently communicate with each other and with edge/cloud servers to maintain the metaverse operation. Consequently, attackers can manipulate metaverse devices by interfering to their communication or through malicious software installed on the devices. These attacks often make the network flows sent from/to the targeted devices become abnormal compared to usual traffic patterns. As metaverse devices possess increasingly computational capacity, these devices can be exploited to build a CID scheme, in which they can join a consortium blockchain to collaboratively train an intrusion detection model, then leverage such the model to protect themselves as well as the metaverse.

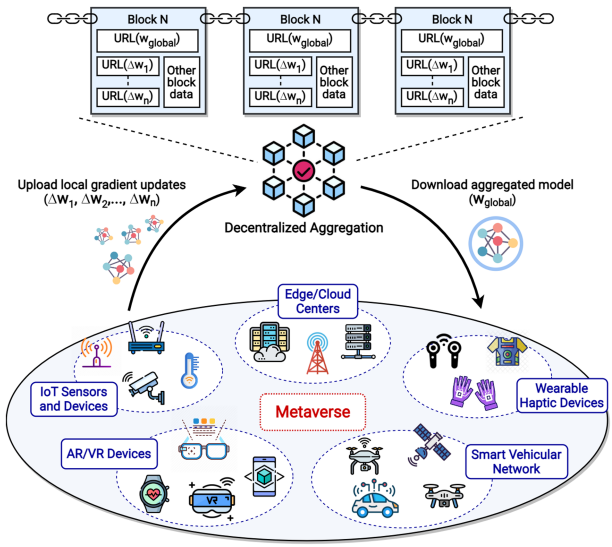

FIGURE 1. FL training process for the IDS model in MetaCIDS. Gradient updates and the global model are stored on IPFS, in which "URL" is the IPFS link to download the data.

Since the training task requires significant computation, it should be carried out by edge/cloud servers or powerful devices such as AR/VR headsets. Other weaker devices (e.g., IoT sensors and haptic devices) can still contribute to defend the metaverse by monitoring their local network traffic to detect intrusion based on the IDS model stored on the blockchain. The key elements of MetaCIDS is illustrated in Fig. 2.

B. THREAT MODEL AND DESIGN GOAL

1) THREATS IN TRAINING IDS MODEL

The training process of IDS models pose various security and privacy risks, which can be summarized as follows.

Data Privacy Leakage: The safety of metaverse users can be threatened if their network traffic data is leaked to any other party during the training process.

Zero-day Attack: IDS model which is trained on a fixed dataset would not be able to detect a novel attack pattern that is unseen in the training dataset.

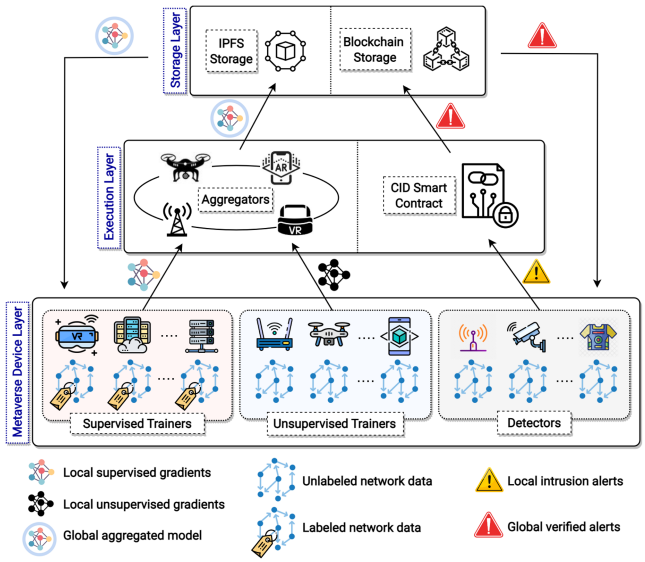

FIGURE 2. Overall architecture and operation of MetaCIDS.

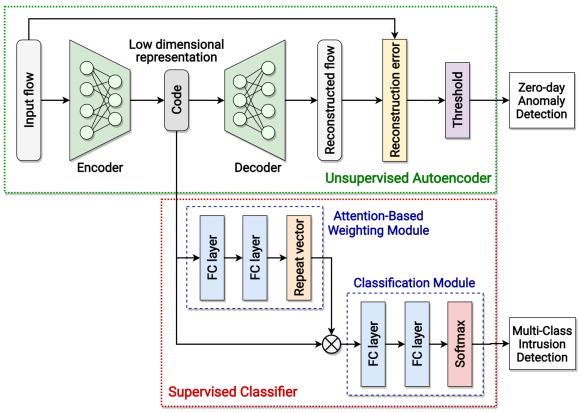

FIGURE 3. Architecture of the lightweight semi-supervised IDS model.

Lack of Labeled Data: In practice, most metaverse users are not able to label their network traffic to train the IDS model. Only a small proportion of users are expected to possess expertise in cybersecurity to do this task.

Inference Attack: In FL training, although the local data of users is not revealed, sensitive information can still be inferred from the submitted local gradient updates.

SPoF in Aggregation: If FL model aggregation is processed by a single central server, the entire training system can be collapsed when the server is attacked by hackers.

Model Tampering: Attackers might interfere in the transmission or storage environment to tamper the IDS model during its training, threatening the integrity of model.

Poisoning Attack: Some malicious metaverse users can intentionally send wrong gradient updates to distort the training process. Consequently, the IDS model cannot converge to an efficient solution under poisoning conditions.

2) THREATS IN CID MANAGEMENT

In terms of CID management, several challenges can be listed as follows.

SPoF in Detection: IDS might be less efficient if the detection result is decided by only a single authority. This can raise trust issues and SPoF, especially in a large-scale and distributed environment like the metaverse.

Fake Alert Submission: In a distributed CID system, some participants can submit fake alerts to earn rewards without truly contributing any effort. Moreover, hackers might do this to disrupt the IDS operation, thus facilitating their attacks.

Privacy Leakage in IDS: CID systems often require users to attach the suspicious network flow associated with the intrusion alert to prevent fake alert. However, this can lead to privacy leakage as the local data must be disclosed.

Tampering of Detection Result: Attackers can launch attacks targeting on IDS storage infrastructure to modify the detection results and logs, thus bypassing the detection system.

Scalability Issue: A centralized IDS in which the central server must monitor and analyze all network data to detect intrusion will inevitably face scalability problem, especially in the metaverse with millions or even billions of devices.

Sybil Attack: Attackers can create numerous sybil avatars to obtain illegitimately high impact on the overall system.

C. DESIGN OVERVIEW

We propose that MetaCIDS conducts online learning to train the model continuously on newly collected network data instead of relying on a specific available dataset. The model architecture is illustrated in Fig. 3, which consists of a supervised classifier for multi-class detection and an unsupervised autoencoder (AE) for zero-day attack detection.

The consortium blockchain in MetaCIDS is responsible for managing both the training process and the detection task. It uses practical Byzantine Fault Tolerance (pBFT) [27] consensus algorithm, in which the blockchain operation is divided into rounds, while a committee consisting of multiple consensus nodes is newly elected in every round. The committee regulates the blockchain operation by verifying all transactions submitted to the blockchain, proposing a new block, and reaching consensus on the block before adding it into the main chain. We choose pBFT since this consensus algorithm can tolerate up to 33% byzantine nodes in the committee thanks to its majority-voting-based protocol, while it also achieves fast finality, thus avoiding blockchain forking [27]. Specifically, the framework includes the following entities.

Unsupervised Trainers: They are able to collect network data and have computation resource to train the IDS model. However, they do not possess cybersecurity expertise to label the collected data. Thus, they can utilize available and unused resource of their devices to train the unsupervised autoencoder.

Supervised Trainers: They are experts or cybersecurity companies that want to participate in MetaCIDS to secure the metaverse, thereby earning metaverse tokens. They can label network data and use it to train the supervised classifier.

Aggregators: They play the role of consensus nodes in the blockchain committee, who take responsibility for aggregating local gradient updates into a global model. Aggregators also verify intrusion alerts and all blockchain transactions to generate a new block and append it into the blockchain.

Detectors: They are devices without strong capacity for training the IDS model. Instead, they download the on-chain model to analyze the monitored network traffic, thereby detecting intrusion and submitting alerts to the blockchain.

CID Smart Contract: This smart contract receives intrusion alerts from detectors, then verifies the alerts based on the associated suspicious network flows. It also emits events to inform participants about the verified attacks.

IPFS Storage: To mitigate the storage burden of blockchain, all data related to the IDS model and local gradient updates are stored on InterPlanetary File System (IPFS). To download some specific data from IPFS, only a simple URL is required.

For simplicity, we use the term "trainers" in this paper for both supervised and unsupervised trainers if no specific information is supplemented.

1) OVERVIEW OF FL TRAINING PROCESS

The training task in MetaCIDS is based on stochastic gradient descent (SGD) updates, in which each training round corresponds to a certain number of SGD epochs. In the first round, trainers download the global IDS model from the IPFS URL available on the blockchain. Next, they use their collected network data to locally train the model, thereby obtaining the SGD updates for that round. To prevent the inference attack, trainers perturb their SGD updates by a DP noise [26] before sending it to the blockchain committee for aggregation. The aggregators in the committee use Multi-Krum [28], an aggregation algorithm that can tolerate up to 33% poisoning attack, to compute the global model based on the received updates. Then, the committee executes pBFT to reach consensus on the aggregated model and publishes a new block containing the IPFS URL of that model. In the next round, trainers download the new global model and repeat the training process continuously on their newly monitored network data. The aggregators are rewarded certain metaverse tokens for each block appended onto the blockchain, while the trainers would receive reputation scores for their contribution, which help them increasing the possibility of being chosen to the committee in the next rounds.

2) OVERVIEW OF CID MANAGEMENT

As the global model's URL is transparent on the blockchain, detectors can download it to detect intrusion over their local network. Firstly, the detector uses the AE's encoder module to encode the network flow that they suspect to be malicious. The encoded flow is then submitted to the blockchain as an alert transaction, which triggers the CID smart contract's function. Consequently, the smart contract automatically verifies the validity of the submitted intrusion alert. If the alert is verified to be honest, the CID smart contract emits an even informing all nodes in registered metaverse networks about the intruder. The detector is then rewarded certain metaverse tokens for their contribution. Otherwise, the detector is slashed that number of tokens due to their dishonest alert. Throughout this process, data privacy is ensured since the suspicious network flow is encoded by the AE's encoder module before committing.

III. METACIDS: BLOCKCHAIN-EMPOWERED METAVERSE CID WITH PRIVACY-PRESERVING FEDERATED LEARNING

Detailed design of MetaCIDS is presented in this section. Our design combines semi-supervised FL with blockchain to enable online model training for efficient privacy-preserving intrusion detection.

A. SEMI-SUPERVISED IDS MODEL

MetaCIDS integrates an IDS model that consists of two sub-models, an unsupervised AE and an attention-based supervised classifier, which is illustrated in Fig. 3 with "FC layer" stands for a fully-connected layer.

1) UNSUPERVISED AUTOENCODER

AE is an unsupervised neural network model that consists of two main components, an encoder and a decoder. The encoder takes a network flow as the input, then maps it into a low-dimensional representation (i.e., the compressed code). On the other hand, the decoder tries to reconstruct the original network flow from the compressed code. The AE is trained to minimize the difference between the original flow and the reconstructed flow based on a loss function such as mean squared error (MSE). In MetaCIDS, the AE model has the following roles:

Zero-day Attack Detection: Since the AE is continuously trained on massive volume of network data to minimize the MSE loss, it is expected to reconstruct any regular network flows with a negligible reconstruction error (RE). When the AE fails to reconstruct a flow (i.e., the reconstructed flow has an abnormally high error), it often indicates the presence of a new attack pattern that is unseen in the training data. Therefore, the AE model is used in MetaCIDS to detect zero-day attacks which could not be recognized by the classifier.

Dimensionality Reduction: The collected network flows often consist of a huge number of features, while not all of them contain useful information for intrusion detection. Therefore, in MetaCIDS, the low-dimensional code produced by the encoder module is used as input for the classifier (presented in Section III-A2) instead of using the original network flow. This can help reducing computation, while improving the classifier's performance.

Privacy Preservation: It is often required to provide the original network flow when committing an intrusion alert for

verification. However, this leads to privacy concern as the network data is exposed widely. In MetaCIDS, only the encoded representation of the flow is required when submitting alerts. As a result, data privacy is ensured since the network flow has been hidden, while the decoder cannot reconstruct the encoded attack flow with an acceptably low error.

To detect intrusion based on the AE model, MetaCIDS computes the distance between the reconstructed flow and the original flow (i.e., the reconstruction error). If the error exceeds a statistic-based threshold presented in (8), its corresponding network flow is determined to be malicious.

2) ATTENTION-BASED SUPERVISED CLASSIFIER

The presented unsupervised AE model can offer only binary prediction instead of multi-class detection, while its performance is often lower than other supervised models which are trained on labeled data. Thus, MetaCIDS also provides a supervised intrusion classifier based on the attention technique [5]. Specifically, the classifier consists of two different modules:

Attention-Based Weighting Module: As mentioned above, the classifier's input is the encoded flow acquired from the AE's encoder. This means that the feature list has been perturbed uncontrollably, while it is also challenging to deploy any feature engineering techniques in a distributed system like MetaCIDS. Therefore, this weighting module allows the classifier to flexibly pay attention to important features of the encoded flow without requiring additional data processing from the trainers. It assigns each feature a weight so that a higher-weighted feature is considered more impactful.

Classification Module: This module is a light-weight neural network with a SoftMax layer to produce multi-class intrusion detection. Since important information has been extracted thanks to the AE's encoder and the weighting module, the classification module does not have to be complex while still achieves high performance. This helps reducing the communication overhead significantly.

B. BLOCKCHAIN-BASED ONLINE FEDERATED LEARNING

We propose to employ online FL to train our devised semi-supervised model for intrusion detection where local trainers collaborate with a set of aggregators to achieve an efficient trained model. The training process is illustrated in Fig. 1, while detailed design is given in the following.

1) PRIVACY-PRESERVING LOCAL TRAINING

The local training task is described in Algorithm 1, where N is the total number of trainers registered to join the consortium blockchain. Specifically, trainers firstly download the global model (denoted by w_{global}) from the latest block. The global model w_{global} consists of two sub-models, which are an AE model w_{ae} and a classifier w_{cls}. The sub-model w_{ae} includes an encoder $w_{\text{ae}}^{\text{encoder}}$ and a decoder $w_{\text{ae}}^{\text{decoder}}$. Once entering the training, trainers start collecting network data and train the model locally on their collected dataset. If a trainer is an unsupervised trainer (i.e., she cannot label the network data), she only trains the w_{ae} on her unlabeled data, while the classifier w_{cls} remains unchanged. On the other hand, each supervised trainer T_i^{sup} only trains the classifier. To do so, she encodes the data:

$$X_i^{\text{code}} = f(X_i, w_{\text{ae}}^{\text{encoder}}), \tag{1}$$

where X_i is the network data of the supervised trainer T_i^{sup}, $w_{\text{ae}}^{\text{encoder}}$ is the latest AE's encoder, X_i^{code} is the encoded low-dimensional data, and $f(\cdot)$ is a feedforward function.

T_i^{sup} uses the encoded data X_i^{code} to train the classifier w_{cls}. As a result, each trainer T_i obtains a gradient update Δw_i for the global model. The untrained part of Δw_i is set to all zeros. For example, unsupervised trainers set the gradient update of the classifier to all zeros as they do not train the classifier.

To prevent inference attack, each T_i also add a (ϵ, δ)-DP noise [26] into their gradient update to perturb the update:

$$\Delta w_i^{\text{DP}} = \Delta w_i + \xi_i, \tag{2}$$

where ξ_i is the DP noise and Δw_i^{DP} is the perturbed update. This noise is sampled from a zero-mean Gaussian distribution, thus minimizing the overall impact on the global model's performance when the number of trainers is large. The detailed impact of DP noise can be found in [26].

Next, each trainer T_i uploads their gradient update to IPFS to acquire an URL link, denoted by U_i. When the training period ends, T_i sends the perturbed local update to the committee for global aggregation via a blockchain transaction:

$$tx^{\text{train}} = \{\text{nonce}, U_i, \text{is\_supervised}, Sig_{sk}(tx^{\text{train}})\}, \tag{3}$$

where "nonce" is an incrementing counter that prevents double submission, is_supervised is a binary value indicating the trainer's type (supervised or unsupervised), U_i is the IPFS link of the local gradient update, and $Sig_{sk}(tx^{\text{train}})$ is the digital signature signed by the trainer's secret key.

MetaCIDS requires trainers to submit the gradient updates through IPFS to decline the communication and storage burden since IPFS URLs are much smaller than the gradient updates, especially when there are a huge number of trainers involved.

2) DECENTRALIZED GLOBAL AGGREGATION

All transactions for gradient submission are collected into a transaction pool. The committee conducts an additional communication round to ensure that the transaction pools of all aggregators are synchronized with each other before entering the aggregation period. Algorithm 2 illustrates the aggregation process, where M is the committee size (i.e., the total number of aggregators, including the leader). Firstly, the leader and aggregators download all gradient updates from IPFS based on the provided URLs. Then, the leader executes Multi-Krum [28] to aggregate these gradients into a single update with poisoning tolerance. Specifically, each local update

Algorithm 1: Local Training.

Input: Set of trainers $\mathcal{T} = \{T_1, T_2, \ldots, T_N\}$, the corresponding local datasets $\mathcal{X} = \{X_1, X_2, \ldots, X_N\}$
Output: IPFS URLs of local updates $\mathcal{U} = \{U_1, U_2, \ldots, U_N\}$

for $T_i \in \mathcal{T}$ **do**
  Download the global model w_{global} from blockchain;
  if X_i is unlabeled **then**
    Train the autoencoder $w_{\text{ae}} \in w_{\text{global}}$ on the local dataset X_i to obtain the local gradient update Δw_i;
  else if X_i is labeled **then**
    Encode the local dataset X_i by the AE's encoder module to obtain the compressed data: $X_i^{\text{code}} = f(X_i, w_{\text{ae}}^{\text{encoder}})$;
    Train the classifier $w_{\text{cls}} \in w_{\text{global}}$ on X_i^{code} to obtain the local gradient update Δw_i;
  end if
  Perturb Δw_i by a Gaussian DP noise: $\Delta w_i^{\text{DP}} = \Delta w_i + \xi_i$;
  Upload Δw_i^{DP} to IPFS, thus obtaining the URL link U_i;
end for
return $\mathcal{U} = \{U_1, U_2, \ldots, U_N\}$.

Algorithm 2: Global Aggregation.

Input: The leader A_{leader}, aggregator set $\mathcal{A} = \{A_1, A_2, \ldots, A_{M-1}\}$, IPFS URLs of local updates $\mathcal{U} = \{U_1, U_2, \ldots, U_N\}$, learning rate η
Output: The global aggregated model w_{global}, a scoring list of local updates $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$

for $U_i \in \mathcal{U}$ **do**
  A_{leader} downloads model Δw_i^{DP} from the IPFS link U_i;
  for $A_j \in \mathcal{A}$ **do**
    A_j downloads model Δw_i^{DP} from the IPFS link U_i;
  end for
end for
A_{leader} executes Multi-Krum algorithm on all local gradient updates to obtain the aggregated update Δw_{global} and the scoring list: $\langle \Delta w_{\text{global}}, \mathcal{S} \rangle \leftarrow \text{MultiKrum}(\Delta w_1^{\text{DP}}, \Delta w_2^{\text{DP}}, \ldots, \Delta w_N^{\text{DP}})$
A_{leader} updates the global model: $w_{\text{global}} = w_{\text{global}} + \eta \Delta w_{\text{global}}$;
A_{leader} and all $A_i \in \mathcal{A}$ execute pBFT protocol to reach consensus on the model w_{global} and the scoring list $\mathcal{S}$;
return $w_{\text{global}}, \mathcal{S}$.

Δw_i^{DP} is assigned a score S_i:

$$S_i = \sum_{i \to k} ||\Delta w_i^{\text{DP}} - \Delta w_k^{\text{DP}}||^2, \tag{4}$$

where $i \to k$ denotes the fact that Δw_k^{DP} belongs to the $(N - f - 2)$ gradient updates that are closest to Δw_i^{DP}, with f is the number of poisoning trainers.

As a result, the leader acquires a scoring list according to N gradient updates of N trainers: $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$. A high Multi-Krum score indicates that the gradient update is greatly different from the rest of updates, implying the presence of a poisoning attack. Therefore, f updates with highest scores are filtered out, while the rest are aggregated into a single update Δw_{global}. Next, the leader updates the global model as follows:

$$w_{\text{global}} = w_{\text{global}} + \eta \Delta w_{\text{global}}, \tag{5}$$

where η is a predefined learning rate.

Finally, the leader creates a new block $\mathcal{B}$ and adds the aggregated model w_{global}, the scoring list $\mathcal{S}$, and all other necessary information into the block. If the committee reaches an agreement on the block $\mathcal{B}$ via the consensus mechanism presented in Section III-E, it is guaranteed that the global model w_{global} has been aggregated correctly.

3) INCENTIVE FOR TRAINING IDS MODEL

To encourage metaverse users to contribute their resource for training the model, the following rewards are offered to trainers and aggregators:

Aggregation Reward: If the committee reaches consensus on an aggregated model, the leader and all aggregators are rewarded certain metaverse tokens, which can be used to purchase digital assets and virtual services in the metaverse.

Unsupervised Training Reward: All unsupervised trainers whose gradient update is selected for aggregation are rewarded 1 reputation score, which helps them increasing the probability of being elected to the committee in the next rounds. Those who are filtered out by Multi-Krum algorithm are slashed 1 reputation score due to their low-quality gradient updates.

Supervised Training Reward: As labelling network data is a sophisticated task that requires additional computation and expertise, validated supervised trainers are rewarded 2 reputation scores, while the rejected ones are punished 1 score.

Reputation scores are also represented as blockchain tokens. However, unlike native metaverse tokens, reputation tokens are not transferable among metaverse users. While the main purpose of reputation is for committee election (presented in Section III-D), it can also be considered as a reference indicating whether a user is reliable in the virtual world.

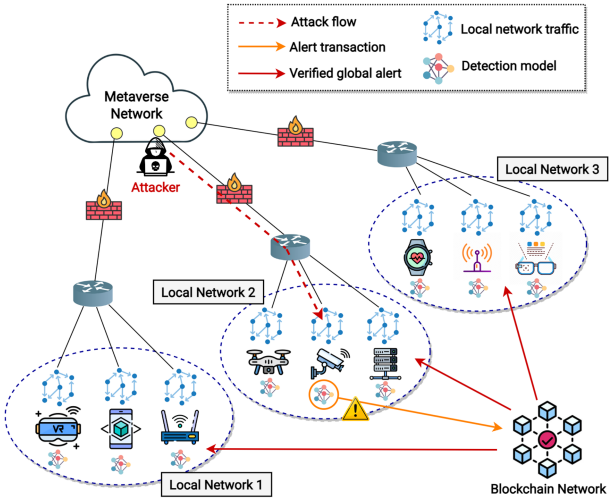

FIGURE 4. Collaborative intrusion detection management in MetaCIDS. In this example, the camera in the local network 2 is being attacked. It uses the IDS model retrieved from the blockchain to successfully detect the attacking network flow, then submits an intrusion alert transaction into the blockchain. The CID smart contract verifies the alert transaction, then emits a global intrusion alert to all nearby local metaverse networks.

C. BLOCKCHAIN-EMPOWERED COLLABORATIVE DETECTION

Traditional centralized IDS frameworks are often designed to operate within a specific local network, therefore cannot detect attacks that originate outside of that environment. In contrast, MetaCIDS offers a scalable detection scheme in which numerous users from multiple metaverse local networks collaboratively detect and report intrusion over a distributed and large-scale system (illustrated in Fig. 4). An alert about an attacker from a local network may help other nearby networks to proactively protect themselves from the reported intruder. Moreover, compounded alerts from multiple sources can efficiently increase the detection rate.

1) LOCAL INTRUSION DETECTION

Detectors in MetaCIDS continuously collect local network flows in every round, then execute Algorithm 3 to detect intrusion among these flows. In particular, detectors must download the IDS model from the latest block of the blockchain. For each collected network flow F_i, a detector firstly encodes the flow by the AE's encoder (i.e., $w_{\text{ae}}^{\text{encoder}}$) to obtain the low-dimensional code F_i^{code}, then input F_i^{code} to the classifier for multi-class detection:

$$\text{Label}_i = f(F_i^{\text{code}}, w_{\text{cls}}), \tag{6}$$

where $f(\cdot)$ is a feedforward function, w_{cls} is the classifier's parameters, Label_i is the network flow's predicted label, and F_i^{code} denotes the encoded flow.

If Label_i specifies an attack (e.g., DoS, DDoS, botnet), the detector appends the corresponding flow into a suspicious list for later submission. On the other hand, a negative label (i.e., "benign") may indicate that there is no attack associated with that network flow, or this is a zero-day attack that is unseen in the supervised training data. Therefore, in this case, the detector continues to input F_i^{code} into the AE's decoder $w_{\text{ae}}^{\text{decoder}}$ to reconstruct the network flow, thereby obtaining F_i^{reconstr}. The reconstruction error is then calculated as follows:

$$\text{RE}_i = \text{L2\_Distance}(F_i, F_i^{\text{reconstr}}), \tag{7}$$

where $\text{L2\_Distance}(\cdot)$ is to compute the L2 distance between two vectors, and F_i^{reconstr} represents a reconstructed flow.

If RE_i exceeds a predefined threshold, the network flow F_i is considered malicious and appended into the suspicious list. While the threshold can be estimated based on experiments, MetaCIDS uses the following statistical threshold:

$$T_{\text{error}} = \overline{\text{RE}} + \alpha \cdot \sqrt{\frac{\sum_{i=1}^{n}(\text{RE}_i - \overline{\text{RE}})^2}{n-1}}, \tag{8}$$

where $\overline{\text{RE}}$ is the mean of reconstruction errors of all collected network flows, n is the number of collected network flows, and α is a parameter controlling the impact of RE's standard deviation on the overall threshold. A larger α results in a higher threshold, thus reducing false positive alarms, but it may increase the number of false negative predictions.

Algorithm 3: Local Intrusion Detection.

Input: Set of collected network flows $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$, a RE threshold for autoencoder-based detection T_{error}
Output: List of suspicious network flows $\mathcal{F}^s$

Download the global IDS model w_{global} from blockchain;
for $F_i \in \mathcal{F}$ **do**
  Use the AE's encoder module $w_{\text{ae}}^{\text{encoder}}$ to encode the network flow: $F_i^{\text{code}} = f(F_i, w_{\text{ae}}^{\text{encoder}})$;
  Use the classifier $w_{\text{cls}} \in w_{\text{global}}$ to classify the network flow: $\text{Label}_i = f(F_i^{\text{code}}, w_{\text{cls}})$;
  if $\text{Label}_i \neq$ "Benign" **then**
    Adds the network flow F_i into the suspicious list $\mathcal{F}^s$;
  else
    Use the AE's decoder module $w_{\text{ae}}^{\text{decoder}}$ to reconstruct the network flow: $F_i^{\text{reconstr}} = f(F_i^{\text{code}}, w_{\text{ae}}^{\text{decoder}})$
    Compute the L2 distance between the original flow and the reconstructed flow: $\text{RE}_i = \text{L2\_Distance}(F_i, F_i^{\text{reconstr}})$
    if $\text{RE}_i > T_{\text{error}}$ **then**
      Adds the network flow F_i into the suspicious list $\mathcal{F}^s$;
    end if
  end if
end for
return $\mathcal{F}^s$.

2) ALERT SUBMISSION AND VERIFICATION

If the suspicious list of a detector is not empty, the detector can submit intrusion alerts to the CID smart contract via an alert transaction:

$$tx^{\text{alert}} = \{\text{amount} = \Theta, addr_{src}, F^{\text{code}}, Sig_{sk}(tx^{\text{alert}})\}, \quad (9)$$

where $addr_{src}$ is the IP address and port number of the suspicious attacker, F^{code} is the encoded representation of the suspicious network flow, and Θ is certain number of metaverse tokens deposited to the smart contract, which will be slashed if the alert is verified to be dishonest.

Once receiving the intrusion alert, the CID smart contract automatically inputs F^{code} to the CID model to verify the alert. If both the AE and the classifier output a negative result (or a "benign" label), the detector's stake of Θ tokens is slashed due to her fake alert. Otherwise, the alert is accepted and the CID smart contract will emit an intrusion event to the participants of all nearby local metaverse networks.

3) INCENTIVE FOR INTRUSION DETECTION

If an intrusion alert is accepted by the CID smart contract, the detector who submitted the alert will receive a *detection reward*, which results in Θ metaverse tokens. Besides, the smart contract also sends back the deposited tokens to the detector.

D. REPUTATION-BASED COMMITTEE ELECTION

MetaCIDS randomly elects committee members for the next round based on a reputation-based election mechanism presented in Algorithm 4. Specifically, the current leader uses verifiable random function (VRF) [29] to generate a random number ϕ_0 and a proof π. The proof can be used to verify whether ϕ_0 was truly randomized. The seed of VRF is the hash of the previous block so that the random process cannot be manipulated. The leader hashes this random number $M-1$ times to finally obtain a set of M random numbers $\Phi = \{\phi_0, \phi_1, \ldots, \phi_{M-1}\}$. Each random number $\phi_i \in \Phi$ is then used to elect one aggregator from the current trainers for the committee in the next round. In particular, the current leader uses ϕ_i to compute a random index as follows:

$$\text{idx}_{\text{elect}} = k \cdot \frac{\phi_i}{2^{||\phi_i||} - 1}, \quad (10)$$

where k is the total number of available reputation tokens of all trainers, and $||\phi_i||$ is the bit-length of ϕ_i (e.g., 256 bits).

Consequently, the trainer who owns the reputation token at index $\text{idx}_{\text{elect}}$ is elected to the next-round committee. Intuitively, the more reputation tokens a trainer owns, the higher probability that one of her reputation tokens is selected, making her being an aggregator in the next round. When all M aggregators have been chosen, the one with highest reputation among them is elected as the next-round leader. If more than one aggregators have the same highest score, the one with lowest ϕ_i is chosen to be the leader.

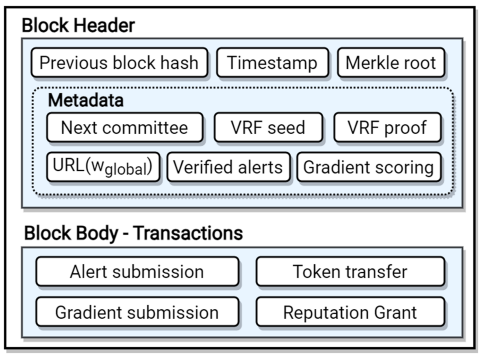

FIGURE 5. Structure of a block in MetaCIDS blockchain.

Algorithm 4: Committee Election.

Input: Set of all existing reputation tokens $\{r_1, \ldots, r_k\}$, set of trainers $\mathcal{T} = \{T_1, T_2, \ldots, T_N\}$, committee size M, previous block $\mathcal{B}^{t-1}$

Output: Next round aggregator set $\mathcal{A}^{t+1}$, next round leader A^{t+1}_{leader}, proof of randomness π

1: Generate a seed for VRF: vrf_seed $\leftarrow$ hash($\mathcal{B}^{t-1}$);
2: Generate a VRF-based random number ϕ_0 and the corresponding proof π: $\langle \phi_0, \pi \rangle \leftarrow$ VRF(vrf_seed, secret_key);
3: Generate $M-1$ more random numbers by hashing $M-1$ times the random number ϕ_0. Consequently, the current leader obtains a total of M random numbers: $\Phi = \{\phi_0, \ldots, \phi_{M-1}\}$;
4: **for** $\phi_i \in \Phi$ **do**
5: Use ϕ_i to select a random index: $\text{idx}_{\text{elect}} = k \cdot \frac{\phi_i}{2^{||\phi_i||}-1}$;
6: Elect the aggregator who currently owns the reputation token at index "$\text{idx}_{\text{elect}}$" into the next round committee $\mathcal{A}^{t+1}$;
7: **end for**
8: Elect the highest-reputation aggregator in $\mathcal{A}^{t+1}$ to be A^{t+1}_{leader};
9: **return** $\mathcal{A}^{t+1}, A^{t+1}_{\text{leader}}, \pi$.

E. BLOCK PROPOSAL AND CONSENSUS MECHANISM

Once entering the consensus process, the leader encapsulates all necessary information into a new block, which is illustrated in Fig. 5. In the block's metadata, the *next committee* field specifies who are elected to be aggregators/leader in the next round, while the VRF seed and proof are provided to allow every participant to verify the randomness of the committee election process. The metadata also includes the global model's URL, a list of Multi-Krum-based gradient scores, and all accepted intrusion alerts. On the other hand, the block's body consists of submitted blockchain transactions.

Finally, the committee executes pBFT consensus mechanism to reach agreement on the new block with three phases:

Preprepare: The leader broadcasts the new block to all aggregators in the committee via a signed *preprepare message*.

Prepare: Aggregators re-compute all information included in the block (e.g., Multi-Krum aggregation, committee election, and transaction validation). If the re-computed information is all matched with the block attached in the received preprepare message, aggregators broadcast a *prepare message* within the committee to confirm the block's validity.

Commit: When the leader and aggregators receive at least 50% of prepare messages, they continue to broadcast a *commit message* to confirm that the majority of committee members have reached an agreement. Finally, the committee reaches consensus when receiving the majority of commit messages.

Once the final consensus is reached, the new block is appended into the blockchain. All metaverse users in MetaCIDS download the new block and continue their training/detection tasks in the next round.

F. SECURITY ANALYSIS

With the presented architecture, it can be shown that MetaCIDS is resistant to all threats formulated in Section II-B.

1) SECURITY IN MODEL TRAINING

Data privacy is completely ensured in MetaCIDS's training process since the trainers only submit the gradient updates without disclosing their data, while inference attack is mitigated by the DP noise. Furthermore, the aggregation process is decentralized thanks to blockchain consensus, making it robust to SPoF and manipulation. Multi-Krum is used to offer poisoning resistance for model aggregation, while the integrity of model is ensured as the global model is stored on-chain with blockchain-enabled transparency, immutability, and auditability. MetaCIDS is also robust to zero-day attack thanks to the unsupervised AE model that utilizes the abundant source of unlabeled network data.

2) SECURITY IN CID MANAGEMENT

Fake alerts in MetaCIDS are filtered out automatically by the CID smart contract in a decentralized manner, thereby mitigating SPoF in detection. On the other hand, malicious detectors will be eliminated from the system via the incentive/punishment mechanism, while sybil identities would gain no impact thanks to the employed reputation system. All detection results are stored on the blockchain to ensure transparency and tamper-proof. Besides, privacy leakage in alert submission is totally solved in MetaCIDS since the detectors do not have to reveal the original network flows when submitting the alerts.

IV. PERFORMANCE EVALUATION

A. EXPERIMENTAL SETUP

MetaCIDS utilizes a consortium blockchain framework based on Hyperledger Fabric, a well-established open-source blockchain development platform [32]. The blockchain network comprises 100 nodes, of which 20 are designated as aggregators (i.e., consensus nodes) in each round based on their simulated reputation profiles. The remaining nodes consist of 60 unsupervised trainers, 10 supervised trainers, and 10 detectors. Each node runs a Docker container to participate in the network and carry out various tasks, such as FL training, Multi-Krum-based model aggregation, and intrusion detection.

The experiments and simulations were conducted on an Intel Core i9-13900 K CPU (3.2 GHz) with 64 GB of RAM. Hyperledger Caliper, a blockchain benchmarking tool that generates transaction workloads and monitors blockchain performance, was used to evaluate the blockchain's performance. To assess the IDS model and FL training process, four network intrusion datasets were employed, including CSE-CIC-IDS2018 [33], CIC-IDS2017 [33], NSL-KDD [34], and UNSW-NB15 [35]. Each dataset contains a substantial number of network flows, with each flow comprising 70 to 80 network traffic features.

B. MULTI-CLASS DETECTION PERFORMANCE

To account for the unbalanced nature of IDS datasets, we evaluated the multi-class detection performance of MetaCIDS based on precision and recall in addition to accuracy. Table 2 presents a comparison of MetaCIDS's classifier model to other supervised models such as SVM, MLP, KNN, Naive Bayes, LightGBM [30], and Tabnet [31]. Despite being significantly lightweight, our classifier outperforms these models on all four IDS datasets. Specifically, it achieves accuracy, precision, and recall figures of nearly 99% for the UNSW-NB15 dataset and around 95–97% for the two CIC-IDS datasets. The performance on the NSL-KDD dataset is lower at 82.25% accuracy since the benign flows in this dataset have already been downsampled by the publisher to mitigate data unbalance.

Overall, the MetaCIDS's classifier achieves the highest performance in most cases. This improvement is attributed to the AE's encoder for dimensional compression and the attention-based weighting module for feature importance.

C. ZERO-DAY DETECTION PERFORMANCE

This experiment evaluates the MetaCIDS's unsupervised AE model in terms of zero-day attack detection. Specifically, we train the AE model on unlabeled data derived from the four mentioned datasets. For each dataset, we reserve the data of two different attack types, while training the model on the remaining data. Consequently, the data of the two reserved attacks are considered zero-day attacks as they are unseen during the training process. Then, the unsupervised model is used to reconstruct both regular network flows and zero-day attack flows. To determine the RE threshold for distinguishing between regular and zero-day attack flows, we used the formula presented in (8).

As depicted in Fig. 6, MetaCIDS's unsupervised model can efficiently detect zero-day attacks with a superior detection rate in most cases. In particular, the model achieved a 100% detection rate for DoS GoldenEye and SSH Bruteforce attacks on the CSE-CIC-IDS2018 dataset, where the reconstruction

TABLE 2. Multi-Class Detection Performance of MetaCIDS Compared to Other Classification Models on Different IDS Datasets

Datasets	CSE-CIC-IDS2018			CIC-IDS2017			NSL-KDD			UNSW-NB15		
Models	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
SVM	88.05	89.31	87.51	89.23	92.32	89.92	80.69	72.39	75.15	98.11	98.22	98.05
MLP	94.81	94.92	94.01	95.48	96.54	95.76	80.70	70.95	75.04	98.61	98.69	98.60
KNN	93.56	93.38	93.29	95.42	96.72	95.89	80.74	72.50	75.86	98.19	98.14	98.13
NB	76.22	75.51	74.16	85.37	87.15	85.81	63.90	66.46	59.98	97.57	96.09	96.60
LightGBM [30]	60.55	62.54	57.89	52.81	83.07	60.04	78.19	65.09	70.41	96.89	97.28	96.86
Tabnet [31]	84.11	93.84	87.06	95.52	96.86	96.58	82.19	73.97	**77.43**	98.74	98.66	**98.69**
MetaCIDS	**96.01**	**95.96**	**95.31**	**96.56**	**97.21**	**96.76**	**82.25**	**74.22**	77.07	**98.77**	**98.78**	98.68

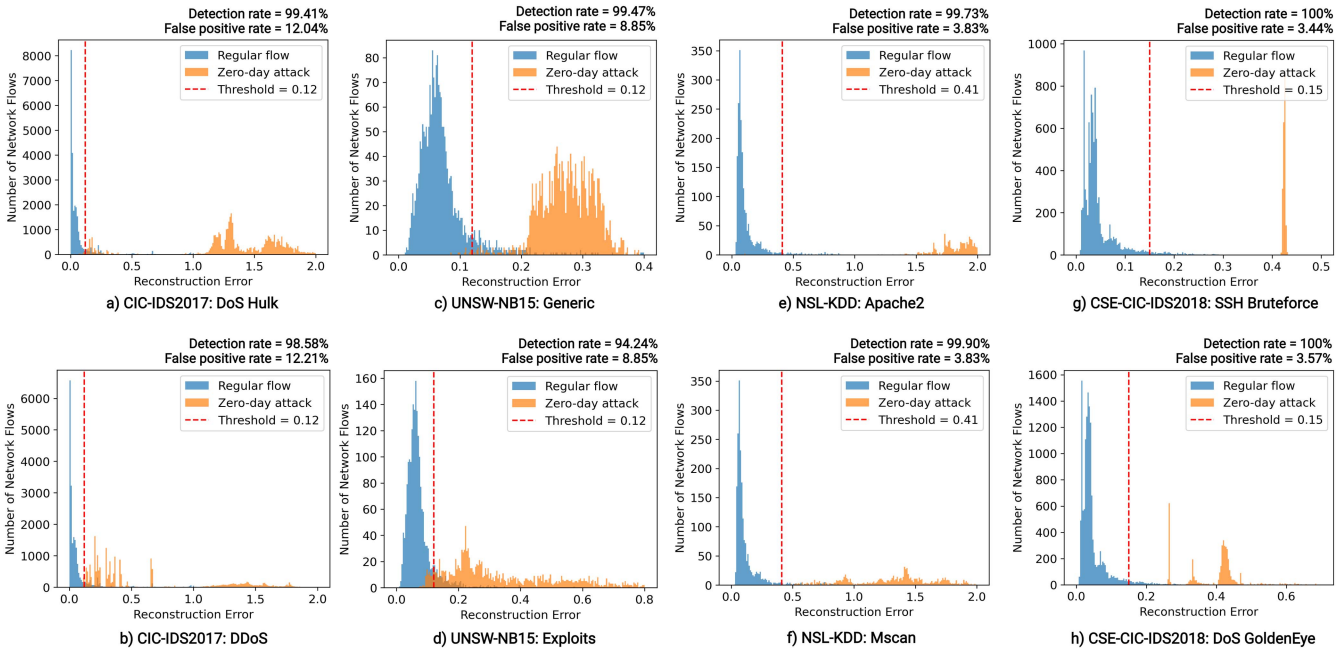

FIGURE 6. Zero-day attack detection on four different datasets. Network flows with reconstruction error greater than the threshold are considered malicious.

errors of zero-day attack flows were clearly distinguishable from the regular flows. However, in certain cases such as DDoS and DoS Hulk attacks on the CIC-IDS2017 dataset, the false positive rate was still significant, ranging from 8-12%.

D. BLOCKCHAIN PERFORMANCE

When the committee size increases, it takes more time for the committee to reach consensus due to additional communication. According to Fig. 7(a), with a transaction arrival rate of 600 TPS, the blockchain's throughput decreases from nearly 600 transactions per second (TPS) to just 100 TPS when the committee size increases from 2 to 50, while the latency increase from 4 seconds to 6 seconds. MetaCIDS selects a committee size of about 20–30 aggregators for a balanced trade-off between performance and decentralization.

Fig. 7(b) illustrates the blockchain's performance when the transaction arrival rate varies from a low workload of less than 100 TPS to a high or even extreme workload of more than 1,000 TPS. The general trend shows that the transaction processing rate decrease while the transaction average latency

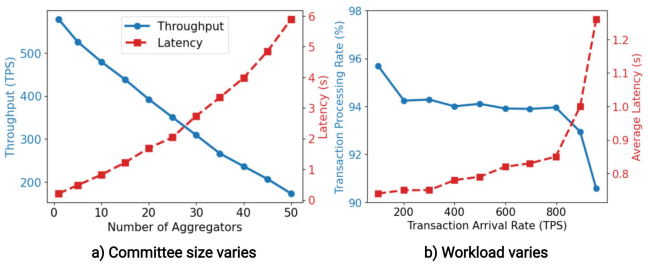

FIGURE 7. Performance corresponding to committee sizes and workloads.

climbs up for higher workload. This trend becomes more significant when there are more than 800 transactions submitted every second. Besides, another experiment is carried out to investigate the impact of block size to the blockchain's processing performance. In theory, a larger block can store more transactions submitted by metaverse users. According to Fig. 8, when the workload is low, there is almost no significant difference in blockchain performance for different block sizes.

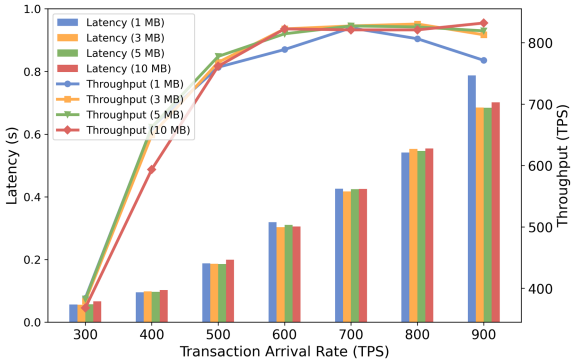

FIGURE 8. Blockchain performance corresponding to different block sizes.

However, when the workload increases to more than 800 TPS, the blockchain with 1 MB block size reaches its limitation and becomes less efficient than the others.

E. FL CONVERGENCE AGAINST POISONING ATTACKS

This experiment investigates the convergence of the FL training process in MetaCIDS under different poisoning conditions. The training process is monitored over 50 blockchain rounds, while each round corresponds to 16 SGD epochs. In each round, trainers collect a new batch of local data which is derived from the CIC-IDS2017 dataset. In terms of poisoning attacks, 33% of trainers are designated as poisoners, while the poisoning types include the following strategies:

Back Gradient: The poisoners reverse the direction of the updates before submitting them to the committee.

Gradient Scaling: The poisoners randomly scale up/down the gradient updates by a random value $\in (0, \mu]$. In this experiment, μ is set to 3.

Random Gradient: Instead of training the IDS model, the poisoners just randomly generate the gradient updates by certain values $\in [\alpha, \beta]$. We select $\alpha = -\beta = -0.5$.

Label Reversal: The poisoners distort the label of every network flow before training the IDS model on it.

As depicted in Fig. 9, MetaCIDS can effectively converge within rounds 10 to 15 for all tested poisoning attack strategies. In contrast, the FL baseline that utilizes FedAvg [4] for aggregation fails to reach the same level of loss achieved by MetaCIDS under the same poisoning conditions.

V. CONCLUSION

In this paper, we propose MetaCIDS, a novel CIDS framework for the metaverse based on blockchain and online FL using an attention mechanism and semi-supervised learning with privacy preservation. Metaverse users can take part in the system to collaboratively train a ML model for intrusion detection, then use such the model to detect attackers over the global network, thereby earning metaverse tokens and reputation scores. The detection model in MetaCIDS is efficient for both multi-class and zero-day attack detection with 95–99% accuracy and detection rate in most test cases,

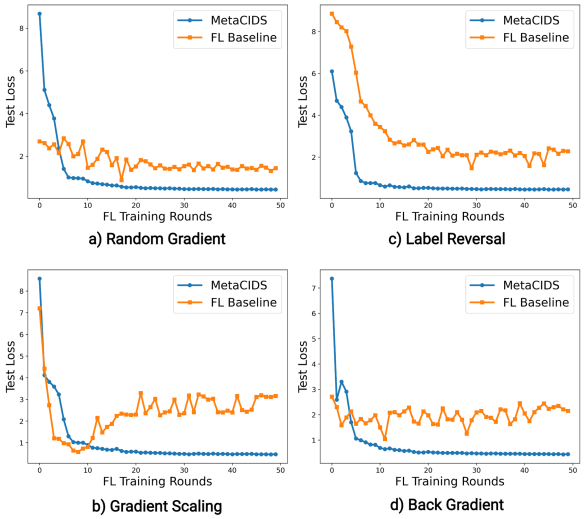

FIGURE 9. Convergence of MetaCIDS compared to a FL baseline on the CIC-IDS2017 dataset against different poisoning attacks, with 33% of poisoners.

which outperforms various ML models such as LightGBM and Tabnet. Security analysis shows that MetaCIDS's training process is robust against data leakage, model tampering, SPoF, poisoning, and inference attack, while the collaborative detection task is highly scalable and resistant to fake alerts and other trust-related issues. However, in practice, the attacks to different types of devices might be different, which potentially leads to a lower detection accuracy of the model. Therefore, our future work will investigate this impact to improve the performance of the IDS model accordingly.

REFERENCES

[1] V. T. Truong, L. Le, and D. Niyato, "Blockchain meets metaverse and digital asset management: A comprehensive survey," *IEEE Access*, vol. 11, pp. 26258–26288, 2023.
[2] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE Commun. Surv. Tuts.*, vol. 16, no. 1, pp. 266–282, First Quarter 2014.
[3] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," *IEEE Commun. Surv. Tut.*, vol. 21, no. 1, pp. 686–728, Firstquarter 2019.
[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[5] A. Vaswani et al., "Attention is all you need," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[6] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang, and J. Han, "When intrusion detection meets blockchain technology: A review," *IEEE Access*, vol. 6, pp. 10179–10188, 2018.
[7] M. Alazab, S. P. RM, P. M, P. K. R. Maddikunta, T. R. Gadekallu, and Q.-V. Pham, "Federated learning for cybersecurity: Concepts, challenges, and future directions," *IEEE Trans. Ind. Informat.*, vol. 18, no. 5, pp. 3501–3509, May 2022.
[8] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of Things intrusion detection: Centralized, on-device, or federated learning?," *IEEE Netw.*, vol. 34, no. 6, pp. 310–317, Nov./Dec. 2020.
[9] S. Ding, L. Kou, and T. Wu, "A GAN-based intrusion detection model for 5G enabled future metaverse," *Mobile Netw. Appl.*, vol. 27, no. 6, pp. 2596–2610, Jan. 2023.

[10] S.-Y. Kuo, F.-H. Tseng, and Y.-H. Chou, "Metaverse intrusion detection of wormhole attacks based on a novel statistical mechanism," *Future Gener. Comput. Syst.*, vol. 143, pp. 179–190, Jun. 2023.
[11] V. T. Truong, V. T. Nguyen, and L. Le, "MetaCIDS: A metaverse collaborative intrusion detection system based on blockchain and federated learning," *TechRxiv*, May 2023, doi: 10.36227/techrxiv.22816568.v1.
[12] A. Heidari, N. J. Navimipour, and M. Unal, "A secure intrusion detection platform using blockchain and radial basis function neural networks for Internet of Drones," *IEEE Internet Things J.*, vol. 10, no. 10, pp. 8445–8454, May 2023.
[13] O. Alkadi, N. Moustafa, B. Turnbull, and K.-K. R. Choo, "A deep blockchain framework-enabled collaborative intrusion detection for protecting IoT and cloud networks," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9463–9472, Jun. 2021.
[14] B. Hu, C. Zhou, Y.-C. Tian, Y. Qin, and X. Junping, "A collaborative intrusion detection approach using blockchain for multimicrogrid systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 8, pp. 1720–1730, Aug. 2019.
[15] N. Alexopoulos, E. Vasilomanolakis, N. R. Ivánkó, and M. Mühlhäuser, "Towards blockchain-based collaborative intrusion detection systems," in *Proc. Int. Conf. Crit. Inf. Infrastructures Secur.*, 2018, pp. 107–118.
[16] Y. Liu et al., "Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6348–6358, Apr. 2021.
[17] 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 Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5615–5624, Aug. 2021.
[18] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh, and O. Jogunola, "Federated deep learning for zero-day botnet attack detection in IoT-edge devices," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3930–3944, Mar. 2022.
[19] O. Aouedi, K. Piamrat, G. Muller, and K. Singh, "Federated semisupervised learning for attack detection in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 286–295, Jan. 2023.
[20] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated-learning-based anomaly detection for IoT security attacks," *IEEE Internet Things J.*, vol. 9, no. 4, pp. 2545–2554, Feb. 2022.
[21] P. Ruzafa-Alcázar et al., "Intrusion detection based on privacy-preserving federated learning for the industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1145–1154, Feb. 2023.
[22] R. Zhao, Y. Wang, Z. Xue, T. Ohtsuki, B. Adebisi, and G. Gui, "Semi-supervised federated learning based intrusion detection method for Internet of Things," *IEEE Internet Things J.*, vol. 10, no. 10, pp. 8645–8657, May 2023.
[23] H. Liu et al., "Blockchain and federated learning for collaborative intrusion detection in vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 6073–6084, Jun. 2021.
[24] X. He, Q. Chen, L. Tang, W. Wang, and T. Liu, "CGAN-based collaborative intrusion detection for UAV networks: A blockchain-empowered distributed federated learning approach," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 120–132, Jan. 2023.
[25] M. Abdel-Basset, N. Moustafa, H. Hawash, I. Razzak, K. M. Sallam, and O. M. Elkomy, "Federated intrusion detection in blockchain-based smart transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2523–2537, Mar. 2022.
[26] M. Abadi et al., "Deep learning with differential privacy," in *Proc. Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.
[27] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, Nov. 2002.
[28] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 118–128.
[29] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Proc. IEEE 40th Annu. Symp. Found. Comput. Sci.*, 1999, pp. 120–130.
[30] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3149–3157.
[31] S. Ö. Arik and T. Pfister, "TABNet: Attentive interpretable tabular learning," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 6679–6687.
[32] E. Androulaki et al., "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. ACM EuroSys Conf.*, 2018, pp. 1–15.
[33] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116.
[34] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD cup 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, 2009, pp. 1–6.
[35] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf.*, 2015, pp. 1–6.

VU TUAN TRUONG received the B.Eng. degree in electrical and computer engineering from the Hanoi University of Technology and Technology, Hanoi, Vietnam, in 2021. He is currently a Graduate Student with the Institut National de la Recherche Scientifique, University of Quebec, Montreal, QC, Canada. His research interests include cybersecurity, blockchain, machine learning, and enabling technologies for the metaverse.

LONG BAO LE received the B.Eng. degree in electrical engineering from the Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam, in 1999, the M.Eng. degree in telecommunications from the Asian Institute of Technology, Bangkok, Thailand, in 2002, and the Ph.D. degree in electrical engineering from the University of Manitoba, Winnipeg, MB, Canada, in 2007. He was a Postdoctoral Researcher with the Massachusetts Institute of Technology, Cambridge, MA, USA, from 2008 to 2010 and University of Waterloo, Waterloo, ON, Canada, from 2007 to 2008. Since 2010, he has been with the Institut National de la Recherche Scientifique, University of Quebec, Montreal, QC, Canada, where he is currently a Full Professor. He is the co-author of books *Radio Resource Management in Multi-Tier Cellular Wireless Networks* (Wiley, 2013) and *Radio Resource Management in Wireless Networks: An Engineering Approach* (Cambridge University Press, 2017). His research interests include smartgrids, radio resource management, network control and optimization, and emerging enabling technologies for 5G-and-beyond wireless systems and the metaverse. Dr. Le was a Member of the editorial board of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and IEEE COMMUNICATIONS SURVEYS AND TUTORIALS. He is the Editor of IEEE TRANSACTIONS ON COMMUNICATIONS and IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING.