Delft University of Technology

Faculty of Electrical Engineering, Mathematics & Computer Science, Cyber Security Group

---

Decentralized Learning Towards Passport-Grade Anomaly Detection

---

Vlad-Mihai Constantinescu

Student number: 5216648

Thesis committee:

Dr. Ir. J.A. Pouwelse (Thesis supervisor)
Dr. H.J. Griffioen

To obtain the degree of Master of Science in Computer Science
Software Technology track with a specialization in Cyber Security.
To be defended publicly on July 8th, 2025 at 09:30 AM.

Decentralized Learning Towards Passport-Grade Anomaly Detection

1st Constantinescu Vlad-Mihai
Distributed Systems
Delft University of Technology

2nd Dr. ir. J.A. Pouwelse
Distributed Systems
Delft University of Technology

***Abstract*—In today's networks, the frequency of distributed cyber attacks made centrally based SIEM solutions vulnerable to bottlenecks, privacy invasions, and single points of failure. This thesis proposes a decentralized anomaly detection platform for autonomous agents, every server visualized as a node, operating independently without centralized control to identify coordinated attacks with a focus on compliance in the European Blockchain Services Infrastructure (EBSI).**

We construct an evolving subnet adjacency graph over destination IPs and use Deep Graph Infomax (DGI) to learn high-quality node embeddings that capture local traffic patterns as well as global information. The fine-tuned online embeddings are concatenated with raw flow features and fed into a universal LSTM-augmented reinforcement learning policy. The LSTM temporal memory enables both sudden and slow, sneaky attacks.

Evaluation metrics (ROC-AUC, F1, Accuracy, Precision and Recall) are computed at two granularity levels: per-agent for each node detection performance and system-level using a "union-of-alerts" decision rule for event-level detection. Experiments on the UNSW-NB15 flow data set demonstrate that our approach improves ROC-AUC from ≈ 0.936 to ≈ 0.95 and F1 from ≈ 0.89 to ≈ 0.913, outperforming the independent PPO-LSTM baseline. The system preserves data privacy, only exposing aggregated scores of anomalies. These results suggest that integrating self-supervised graph embeddings with recurrent multi-agent RL produces a robust, scalable, and privacy-preserving SIEM solution tailored for the EBSI federated environment. Further studies on window granularity, hyperparameters, and per-subnet policy specialization could potentially further validate design choices and offer a roadmap for deploying decentralized network defense systems.

***Index Terms*—Decentralized Anomaly Detection, Decentralized SIEM, Privacy-Preserving Monitoring, EBSI, Sovereign Security Agents**

I. Introduction

Today's networks face a flood of security events, ranging from user logins and file transfers to API requests. Traditional Security Information and Event Management (SIEM) solutions centralize these events for analysis, but as organizations embrace multicloud and edge computing architectures, transfering every log into a single repository creates serious bandwidth and storage bottlenecks. Moreover, edge filtering or sampling to reduce data volume can leave critical blind spots. [1] [2]

Centralized SIEM systems also present significant privacy and compliance challenges. Regulations like GDPR require controls on personal data, and even encrypted or pseudonymized logs can leak sensitive information through metadata. [3].

Furthermore, storing all security events in one place forms a high-value target, introducing a single point of failure that must be trusted by the other parties. If this central authority is misconfigured, overloaded, or compromised, the entire network loses visibility. In federated environments, such as public sector bodies under the European Blockchain Services Infrastructure (EBSI) [4], no participant will give full control of its logs to another party, creating gaps in collective defense and leaving coordinated multi-node attacks undetected [5]. In addition to the coordinated attacks, big network such as EBSI can present load balancers or NAT points, splitting the traffic, distributing targeted attacks to multiple nodes, resulting in undetected attacks.

Decentralized approaches offer a promising alternative. By empowering each server (to be called an agent from now on) to analyze its own logs locally and share compact aggregated indicators, organizations can avoid central bottlenecks, reduce the risk of a single point of failure, and better comply with privacy regulations.

Recent studies in edge security emphasize how distributing analysis tasks closer to data sources not only improves scalability but also enables finer-grained control over sensitive data, agents being able to apply local anonymization before sharing alerts. [6] [7]

Meanwhile, the EBSI initiative has demonstrated how federated governance and self-sovereign identity frameworks enable trust without centralized intermediaries. By issuing verifiable credentials [5] and maintaining tamper-resistant records, EBSI provides a blueprint for privacy-preserving collaboration across jurisdictions [8]. These principles inspire a SIEM model where independent nodes cooperate on threat detection without exposing raw logs to a central authority.

Advances in graph neural networks and self-supervised learning further enable this vision. By constructing graphs-like structures that capture relationships between network entities and applying contrastive objectives, it allows for development of methods to extract embeddings reflecting both local behaviors and global context.

This thesis integrates these technological advancements into a novel, unified, decentralized SIEM framework tailored for federated environments like EBSI. We develop a pipeline where each node aggregates local flow features, constructs subnet-based graphs, and learns dynamic graph embeddings online. Independent RL agents decide on alerts using both

local and embedding-based inputs.

By exchanging only minimal, privacy-preserving state vectors, our design scales to large federations, creating a compliant and resistant architecture, eliminating single points of failure, all while delivering robust anomaly detection.

The rest of this thesis is organized as follows: Chapter I presents the introduction, Chapter II defines the problem, Chapter III reviews related work, Chapter IV describes the methodology, Chapter V describes the dataset used in the experimental setup, Chapter VI details the experimental setup, Chapter VII reports the results gathered from the experiments, Chapter VIII discusses limitations and future work and Chapter IX offers the conclusion.

II. Problem Description

Traditional Security Information and Event Management (SIEM) systems rely on centralizing logs and events from across the network into a single repository for analysis. While this approach simplifies correlation and threat hunting, it struggles to keep pace with today's evolving, multi-cloud architectures.

As data volumes increase, transferring every log to a central authority drastically increase the bandwidth and storage costs. This most often forces organizations to sample data which can hide threats.

Furthermore, storing all sensitive logs in one place creates a high-value target for attackers and raises serious privacy concerns. Misconfiguration, downtime, or compromise of the analytics hub blinds the entire organization. This gap makes it difficult to detect more advanced, coordinated attacks that span multiple targets, or attacks that are distributed to multiple nodes via internal load balancers or NAT.

Moreover, personal identifiers and metadata must be pseudonymized and strictly controlled to comply with GDPR, yet even encrypted or masked records can leak information through metadata.

Meanwhile, decentralized approaches present significant advancements in adjacent domains. Self-sovereign identity frameworks demonstrate how individuals can fully own and selectively share verifiable credentials without entrusting raw personal data to a central authority. Federated learning shows that models can be trained collaboratively across many nodes, each keeping its data locally to preserve privacy. However, these ideas have seen little application in SIEM technology.

This being said, the core challenge is to rethink SIEM as a truly distributed system: one where each server analyzes its own traffic, shares only compact signals rather than raw logs, and collectively learns to spot both local anomalies and coordinated, multi-node attacks. We must discover how to represent security signals in a bandwidth-efficient, privacy-preserving way, and how to adapt continuously to evolving threats without central bottlenecks. Addressing these questions is essential to building a SIEM framework that scales with modern networks, respects privacy regulations, and withstands single points of failure.

III. Related Work

Over the past decade, researchers have explored various ways to distribute security monitoring and analysis in order to overcome the limitations of centralized SIEMs. One early tentative, BlockSIEM [6], proposed using a permissioned blockchain to record security events in a tamper-proof ledger, allowing multiple "SIEM miners" to verify alerts without relying on a single log repository. Although this design improved integrity and non-repudiation, its reliance on smart contracts and full-chain replication introduced substantial latency for high-throughput environments.

More recently, the concept of a cybersecurity mesh has emerged as a guiding vision for decentralized threat detection. In their survey, Ramos-Cruz et al. [9] argue that by treating each network node as an independent security sensor cooperating through federated learning and blockchain. They can achieve both scalability and resilient coordination. This mesh approach shares our goal of local analysis combined with peer collaboration, but practical implementations remain sparse, particularly when it comes to real-time anomaly detection.

With increased data protection regulations, modern SIEM designs must also preserve privacy. Menges et al. [10] examine the impact of the EU GDPR on SIEM data handling and propose an architecture to process security telemetry in compliance with privacy requirements. An important technique to be mentioned is pseudonymization of personally identifiable information in logs.

In practice, Vazão et al. [11] built and evaluated an open-source SIEM prototype with inline data pseudonymization to ensure GDPR compliance. Their works illustrate that the future of SIEM solutions must balance security analytics with data confidentiality controls.

In the field of collaborative machine learning, federated learning (FL) has shown it is possible to train models on decentralized datasets without exchanging raw records.

As a concrete example, Lazzarini et al. [12] evaluate FL for IoT intrusion detection using a lightweight neural network shared among clients. Their results show that a collaborative federated IDS can approach the accuracy of a centralized model while each client keeps its network logs local.

Parallel to federated approaches, graph-based and self-supervised methods have gained traction for cybersecurity tasks. Caville et al. introduce Anomal-E [13], a system that constructs communication graphs of hosts and applies Deep Graph Infomax (DGI) to learn node embeddings that reflect both local flow patterns and global topology. This method uncovers complex attack structures without labeled examples, paving the way for more generalizable anomaly detectors. In their approach, network entities (hosts, IP addresses) form nodes in a graph, and communication flows form edges. A Graph Neural Network encoder (based on GraphSAGE [14]) is trained using DGI to maximize correlated information between local and global representations, resulting in powerful embeddings for anomaly detection.

Although Anomal-E focuses on offline analysis, its use of DGI inspires our choice of an online, fine-tuned graph encoder for streaming threats.

Another emerging direction is the use of Multi-Agent systems and Reinforcement Learning for distributed anomaly detection. Instead of a single monolithic detector, multiple intelligent agents can be deployed across a network, each learning to identify local abnormalities and collaborate to flag global threats. One approach is to have agents collaboratively learn expected behaviors and then detect deviations. For example Kazari et al. [15] propose a Multi-Agent-Reinforcement-Learning (MARL) based system where agents predict each other's actions to recognize an insider with abnormal behaviour. These multi-agent and reinforcement learning techniques align with the trend of autonomous cyber defense, where distributed intelligent agents collectively guard large, complex infrastructures.

In parallel, recent advances in federated graph learning demonstrate the power of combining privacy-preserving model training with attention-based graph neural networks. Wu et al. introduce a Federated Graph Attention Network (FedGAT) [16] that organizes traffic logs into chronological graphs and leverages node-to-node attention to detect network attacks across decentralized clients without sharing raw data. Additionally, Liu et al [17] propose an Adaptive Multi-channel Bayesian Graph Attention Network (AMBGAT) for detecting anomalies in blockchain transaction graphs, showcasing how Bayesian attention mechanisms can adaptively weight heterogeneous channels for improved classification in privacy-sensitive applications [2].

In summary, prior work has demonstrated the feasibility of decentralized logging, federated training, self-sovereign identity, graph-based anomaly scoring and multi-agent defense coordination.

However, a unified framework that combines online, self-supervised graph embeddings with self-sovereign agents remains unexplored.

This thesis aims to fill that gap by integrating these strands into a cohesive, privacy-preserving, and scalable SIEM architecture suitable for federated, compliance-driven environments.

IV. Methodology

In this chapter, it is described each step of our decentralized SIEM pipeline, from local data processing to graph embedding, multi-agent environment design, and online learning. We break the workflow into four main stages: **Local Feature Aggregation, Subnet Graph Construction, Online Graph Embedding via Deep Graph Infomax, Multi-Agent Reinforcement Learning Environment** .

A. Local Feature Aggregation

Each node (destination IP) collects raw flow records from its local network traffic. We aggregate these records into fixed-length time windows (3 seconds) and compute a summary: packet counts, byte volumes, flow durations, and protocol distributions. This produces a feature vector $x_i^t = \in^d$ where d is the number of features used, per node at t time window.

B. Subnet Graph Construction

We construct an undirected adjacency graph based on /24 subnets: nodes i and j share an edge if they belong to the same subnet. This coarse grouping captures the likely coordination or exposure to shared threats among hosts. In real deployments, particularly under EBSI, load balancers, NAT gateways, or virtualized endpoints may distribute traffic across multiple internal servers that share a public /24 address. By defining edges at the subnet level, we implicitly link these distributed instances, allowing the graph encoder to learn patterns that span behind a single public IP. The static edge list is stored as an index tensor E, which remains fixed throughout the training.

C. Online Graph Embedding with Deep Graph Infomax

We employ Deep Graph Infomax (DGI) to extract robust node representations that capture both local traffic characteristics and global network structure without requiring labeled data.

DGI's self-supervised contrastive objective encourages embeddings to align with the overall graph summary, making it well suited for evolving threat patterns. By fine-tuning online at each RL iteration, the encoder adapts continuously to new behaviors, ensuring that agents receive up-to-date graph context alongside raw features. We implement the encoder f_θ as a two-layer Graph Attention Network (GAT).

1) **GAT encoder:** Two successive `GATConv` layers aggregate neighbor features via learned attention, interleaved with ELU activations and dropout.
2) **Summary vector:**

$$s^{(t)} = \sigma\Big(\frac{1}{N}\sum_{i=1}^{N} z_i^{(t)}\Big),$$

where $z_i^{(t)} = f_\theta(x_i^{(t)}, E)$ and σ denotes the sigmoid function.
3) **Corruption:** We generate negative samples by randomly permuting rows of $X^{(t)}$, yielding corrupted embeddings $\tilde{z}_i^{(t)}$.
4) **Contrastive loss:**

$$\mathcal{L}\text{DGI} = -\sum i = 1^N \left[\log \sigma\big(z_i^T s\big) + \log\big(1 - \sigma(\tilde{z}_i^T s)\big)\right].$$

5) **Online fine-tuning:** At each RL iteration, we sample mini-batches of past graphs (batch size 4), compute $\mathcal{L}$DGI, and update θ via backpropagation.
6) **Feature concatenation:** For the current window, compute $Z^{(t)} = f\theta(X^{(t)}, E)$ and form each agent's observation:

$$o_i^{(t)} = \big[x_i^{(t)} \| z_i^{(t)}\big] \in \mathbb{R}^{49+d}.$$

D. Multi-Agent Reinforcement Learning Environment

We model each node as an independent agent in a synchronous `MultiAgentEnv`. At time t, agents receive observations $o_i^{(t)}$ and choose actions $a_i^{(t)} \in 0, 1$. The instantaneous reward is:

$$r_i^{(t)} = \begin{cases} +1, & \text{if } a_i^{(t)} = 1 \text{ and } y_i^{(t)} = 1, \\ -0.5, & \text{if } a_i^{(t)} = 1 \text{ and } y_i^{(t)} = 0, \\ -0.2, & \text{if } a_i^{(t)} = 0 \text{ and } y_i^{(t)} = 1, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Episodes span all windows in the training split (first 80%), truncated into segments of length $L = 20$ for LSTM backpropagation. Each agent's LSTM hidden state resets only at the start of each episode

E. Individual per-agent LSTM-Enhanced Policy via PPO

We adopt an individual per-agent policy Proximal Policy Optimization (PPO) with an LSTM backbone.

F. Threat Model

In order to support the applicability of our implementation, we define the following threat model, specifying the adversaries considered, assumptions and guarantees upon the system.

As a first **assumption**, agents are considered accesible to the internet and are assumed to run an unmodified version of the detection software. Furthermore, the communication between agents is assumed to be authenticated and integral.

In our threat model the system is considered to not rely on a central authority for detection or policy distribution, and raw logs are not shared between nodes.

As far as the adversarie's capabilities goes, we consider the system designed against external adversaries, which can interact with any or multiple of the agents and can launch reconnaissance activities or various types of attacks, both targeted and distributed, upon the agent/agents. We assume that adversaries cannot directly tamper with the detection logic or the LSTM policy weights.

As far as the security goals goes, our system is designed to achieve privacy preservation, meaning no raw logs are exchanged between the agents, aligning with various requirements such as GDPR and EBSI. Furthermore, the system must detect both localized and distributed attacks using a combination of local observations and global context.

The system must also be resistant to partial network failure and the anomaly detection must continue even if a subset of agents are unavailable.

Lastly, any agent must make their own decision independently in a scalable and fully decentralized fashion.

Regarding **limitations**, the current implementation assumes a static subnet topology and does not explicitly defend against long-term embedding poisoning. More limitations are discussed in Section VIII.

V. Dataset

In this chapter, one can find a description of the network flow data used in the experimental setup, how it was processed, and some initial insights that motivated the design choices.

A. Dataset Overview

The dataset used in the experiments is UNSW-NB15, which contains raw flow records captured over a 24hours time period in late 2015.

It containss approximantely 10 million flows in total, each record representing an unidirection TCP/UDP transaction.

The dataset under discussion comes in the form of multiple headrless CSV files, so a custom python script was used to parse, load, and merge the data into one big data table with 49 features per flow.

For simplicity and explainability reasons, we split the features into categories with some example:

- Basic fields: source Ip (srcip), destination IP (dstip), ports, protocol, state.
- Trafic volume and timing: bytes sent/received (sbytes, dbytes), duration (dur), inter-packet times (Sinpkt, Dinpkt)
- Behaviour flags: HTTP depth, FPT login ,same-IP-port indicati (is_sm_ips_ports)
- Aggregated counters: counts of recent connections per service or host (ct_srv_src, ct_dst_lmt, etc.)
- Timestamps: timestamp of the first packet (Stime), timestamp of the last packet (Ltime)

The records of the dataset are aggregated into non-overlapping 3s windows and labeled per IP, resulting in a per-actor, time-sensitive dataset. After constructing the summary table and the /24-subnet graph, the wondows are chronologically sorted, then split into **Training set** (first 80 % of windows) used for RL training and DGI online fine-tuning, and **Evaluation set** (final 20% of windows), used only for metric computation.

This time-aware split prevents look-ahead leakage and mimics a streaming, online deployment.

B. Participant (IP) Analysis

We start by analyzing the unique destination IPs in the dataset and identifying how many experienced at least one attack. This analysis reveals whether malicious traffic is restricted to a few hosts or spread across the network.

As can be seen in Table I, the data set presented a total number of 47 independent hosts, of which 10 presented attack windows.

The distribution of attacks per IP host can be found in Figure 1.

TABLE I
SUMMARY OF DESTINATION IPS

Metric	*Count*	*Share (%)*
All destination IPs	47	100%
IPs ever under attack	10	21.27%

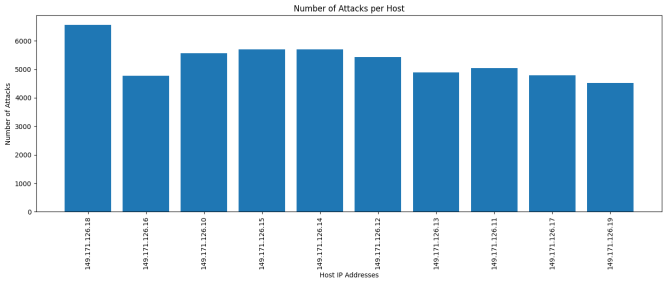

Fig. 1. Number of attacks per host.

From this we can see that about one in five hosts in the network experienced malicious activity during the captured period. Most IPs remain clean, suggesting that a detection approach must balance sensitivity with the noise of benign flows.

C. Flow-level analysis

Understanding the overall balance between normal and attack flows is also key. In the full dataset:

- Normal flows windows:12,861,31 (3.95%)
- Attack flows windows: 52,993 (96.04%)

So about 4% of all windows are malicious. This moderate imbalance is manageable with standard techniques such as weight-loss functions.

VI. Experimental Setup

To rigorously evaluate our decentralized SIEM pipeline, we propose the following experimental setup. The goal is to demonstrate (a) that our online, individual-policy, LSTM-enhanced approach outperforms static and feed-forward baselines, (b) the impact of fully independent agents versus policy sharing, and (c) how detection quality varies at the per-agent and system-wide levels.

A. Configurations

We compare three main configurations:

- Independent LSTM-PPO Policy
  - Baseline
  - Each agent trains its own LSTM policy (same architecture) on only its local observations and rewards.
  - No parameter sharing across IPs.
- Independent LSTM-PPO Policy + Online DGI
  - Each agent trains its own LSTM policy (same architecture) on only its local observations and rewards.
  - No parameter sharing across IPs.
  - Online DGI fine-tuning per training iteration.
- Shared LSTM+PPO Policy
  - Act as a upper bound in our experiments.
  - One policy over all agents.
  - Each agent has full view over the network.

For simplicity, in this paper we consider one **episode** as one run through the training dataset and one **step** as one 3s aggregated window. We run each configuraton until the ROC-AUC converges, which for our experimental setup was after 3 episodes. We test our approach against the Independent LSTM-PPO Policy to showcase the gains that comes with the addition of global-context awareness.

Furthermore, the shared LSTM-PPO policy acts as an upper bound, demonstrating the room for improvement and validating that the our implementation **is closing the gap between Isolation and Perfect Information**

B. Metrics and Evaluation

At the same time, both per-agent and system-wide metrics are collected.

As the per-agent metrics, **ROC-AUC, F1, Accuracy, Precision and Recall** scores are computed by comparing all agent-window predictions with the real action in the evaluation part of the dataset. This measures each model's ability to differentiate between mallicious and normal trafic from the point of view of a individual nodes.

Furthermore, as for many security use-cases an alert anywhere in the network results in a global-wide investigation, during the experiment, are also computed the ROC-AUC and F1-scores under a union-of-alers rule: for each window, if any agent raises an alert, count the window as "detected".

This system-level view captures how well a decentralized ensemble of agents can catch coordinated or distributed attacks even if some nodes miss them.

C. Scalability Benchmark Setup

In addition to model evaluation, a scalability microbenchmark was conducted to assess how the system behaves as the number of participating agents increases.

To see how well this scales in practice, we ran a microbenchmark starting from a base setup of 25 agents trained on UNSW-NB15 traffic. We then increased the number of agents by scaling up to 50, 125, and 250 agents. In order to keep the traffic look realistic while increasing the number of agents, we create replicas of already existing agents, assign them other IP addresses and then duplicate the existing traffic from the existing agent to the replica. This way we are simulating larger organizations or networks.

We measured how long it takes per training step to compute graph embeddings with DGI and to exchange those embeddings via a gossip protocol. The selected gossip protocol works as the following: each agent communicates with up to 3 randomly selected neighbors per gossip round, following a fanout-3 model.

We consider the following formula: $gossip\_time = rounds \cdot (SIM\_LANTECY + overhead)$, where the number of gossip rounds is estimated as $rounds = \log_2(N)$, and each round's overhead is composed of the network latency (SIM_LATENCY) (which for our experiment was capped on 2ms) and serialization + deserialization overhead.

We do not count for packet queue, packet drop, real network connection, transfer or congestion.

The gossip topology is defined by the /24 IP-prefix proximity graph.

VII. Experimental Results

In this chapter, one can find a comprehensive evaluation of the framework under discussion against the disputed baseline as described in Section 6.

We first describe the quantitative performance in terms of ROC-AUC ,F_1, Accuracy, Precision and Recall scores at both per-agent and system-level granularity.

A. System-Wide "Union-of-Alerts" Analysis

We apply the union-of-alerts rule: a window is flagged if *any* host raises an alert. ROC-AUC system level, F_1, precision, precision and recall metrics are reported in Table III, presented in the Appendix.

From the beginning, we observe that global context aggregation amplifies gains: the independent policy reaches 92% ROC-AUC on average, while the DGI-enhanced version hits 95%. This demonstrates that even if some agents miss an attack, others compensate, as long as each maintains fresh graph embeddings.

Specifically, as shown in Figure 3 (left), our system achieves a higher precision, averaging around 84.6% versus 80.7%, generating fewer false alarms than the No-GAT model and achieving the same precision as our 'perfect information' shared policy (84.9%).

Figure 3 (center) illustrates the recall metric. Although shared PPO remain at highest recall ($\geq$97%) and both our and baseline models present high detection rates ($\geq$97%), the GAT model presents itself with an average of 98.7% compared to 97.5% for the baseline.

In terms of overall accuracy, presented in Figure 3 (right), our implementation presents a gain of about 2 percentage points, from 91.5% (baseline) to 93.5% correct decisions, only 0.1% away from the shared policy with 93.6%.

The F1-score is presented in Figure 4 (left) and confirms that our approach balances Precision and Recall more effectively, improving from 88.7% up to 91.1%, while the shared policy stands at 91.4%.

Finally, Figure 4 (right) shows ROC-AUC rising from approximately 93.0% without GAT to 94.7% with GAT, showing that it differentiates better between classes when using dynamic graph embeddings.

B. Independent Agent Analysis

While, according to the system-wide analysis above, the DGI-enhanced system, by useing global relationships, outperformed the baseline, a more granular, per-agent analysis might reveal more information to support our claim.

Figure 5 and Figure 6 plot the metric described above per agent under the three policies under discussion, revealing how local detectors, DGI-enhanced agents, and a fully centralized policy compare at the agent level.

One can find the evolution of per-agent ROC-AUC, F_1, Precision, Recall and Accuracy metrics in Tables V IV VI.

One observation to be made is about the high variability across agents, as precision ranges from as low as 62.7% (for 149.171.126.13) to as high as 92.7% (for 149.171.126.10), while the shared policy presents a smaller range, from 77.6% to 91.9%. This hardens the idea of some of the agents encountering noisy data and sparse attack signals when training.

Another observation is the consistent weak links, specifically IPs 149.171.126.12 and 149.171.126.13. These agents presents consistently one of the lowest F1-scores and ROC-AUC, indicating that highly imbalanced data struggles to provide reliable policies without cross-agent context.

Futhermore, one can observe the consistency of strong performing agents, specifically IPs 149.171.126.14 and 149.171.126.17, which achieve F1-scsores ≥ 0.92 and ROC-AUC ≥ 0.93. This is a clear signal that when local data is meaningfull enough, independent training can approach optimal classification. However, even for such strong performers, the addition of global context awareness increases the ability to classify traffic. For example, in the case of 149.171.126.17, the ROC-AUC increased from 0.961 to 0.973 and F1 score from 0.924 to 0.942.

The patterns described above confirm the hypothesis that a subset of IPs can become robust detectors on their own, while a significant fraction remain vulnerable in isolation. This further hardens the motivation for out DGI-based approach as by sharing embeddings, individual agents can elevate the weaker ones and maintain a high system-wide performance.

Overall, DGI-enhanced agents generally sit, performance-wise, between the fully independent approach (lower bound) and the Shared-PPO, roughly at 80–90% of that gap.

In the case of some of the agents, the GAT encoder even outperforms the shared policy, showcasing the ability to specialize detection to local graph structure while still using global context.

C. Scalability Benchmark

The results reveals that the DGI encoder presents near-linear growth with the numbe of agents, however, it reamins wihtin sub-second latency for up to 125 agents. Communication overhead from the gossip algorithm, remains negligible throughout with a maximum of 6 miliseconds observed.

Overall, these results is hardening the idea that the presented system is suitable for real-time operation across moderately sized decentralized networks and can be used as foundation for scaling to larger enterprise-level environments.

TABLE II
Scalability Benchmark

Agent Count (N)	***DGI Time (seconds)***	***Gossip Time (seconds)***
25	0.185	0.002
50	0.233	0.003
125	0.308	0.006

VIII. Limitations and Future work

A. Static Graph Topology

Although the presented framework demonstrates strong detection performance and practical resource requirements, several limitations and open challenges can be observed.

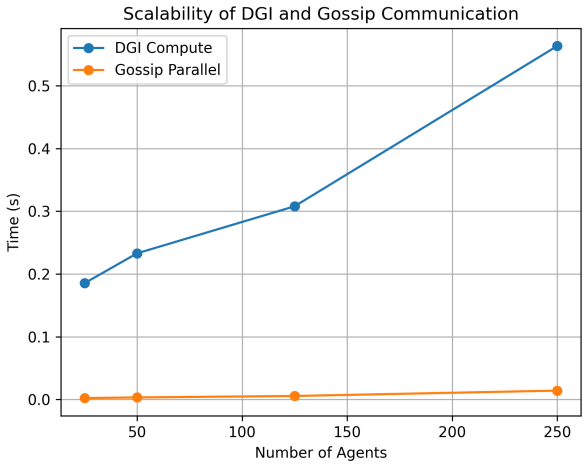

Fig. 2. Scalability Benchmark

As a first limitation to be brought into discussion, a fixed /24-subnet adjencecy graph is built at deployment time. This choice successfully captures patterns between nodes behind load balancers of NAT, however, it lacks the ability to adapt if network address changes. For example, new subnets brought online, dynamic IP assignment, virtual IPs or if the attack pivot across subnets.

With this limitation in mind, one can research the effectiveness of a system that periodically rebuilds or augments the graph from recent traffic statistics.

B. Scalability to Very Large Networks

As part of the evaluation of this thesis, 47 hosts were detected and used to train on a single CPU machine.

However, a network part of a larger enterprises may consist of thousands of monitored hosts, raising concerns about exceeding real-time level processing speeds.

As the the number of hosts increases, the cost of GAT message-passing and the per-iteration cost of the PPO rollouts increase.

As a potential fix, we propose the usage of mini-batch graph sampling algorithms such as GraphSAGE-style neighborhood sampling.

Additionally, one can create try to create a hierarchical partitioning system where the IP space is partitioned into clusters and distinct encoders and policies and trained with occasional cross-cluster syncronization.

C. Cold-Start for New Agents

When a new, previously unseen IP first appears, it has no historical embedding or policy experience. In the current implementation, new agents simply inherit the global DGI encoder's parameters. This approach lacks any fine-tuning specific to the node until the policy learns about node's trafic profile. In this period of 'uncertainty' detection accuracy can suffer as the encoder may produce embeddings that do not reflect the node's true behaviour and the policy may miss genuine attacks or label as annomaly normal traffic.

This behaviour can be a potential vulnerability in fast-moving environments, where nodes spin on and off rapidly.

As for future improvements, we propose the use of a hybrid meta-learning strategy such as Model-Agnostic Meta-Learning (MAML) [18].

An alternative solution could potentially be to temporarily increase exploration for new agents. For example, increasing the entropy coefficient or ϵ for a period of time, could lead to faster policy tuning.

IX. Conclusion

In this work, it was designed and evaluated a novel, fully decentralized Security Information and Event Management (SIEM) framework that brings together self-supervised graph embeddings and multi-agent reinforcement learning.

Rather than sending every log into a central repository, our approach treats each server as an independent "agent" that processes its own traffic, communicates only compact anomaly signals, and learns collaboratively to spot coordinated attacks. By aligning with EBSI compliance guidelines, our proposed design preserves privacy and data sovereignty by avoiding centralized data collection.

Our contribution consist of two main blocks.

On one hand, instead of moving every single network packet to one big database, we build a small "subnet graph" that links together IPs in the same /24 block and run a self-supervised graph encoder (DGI) on it. That means each server **only ever shares compact node summaries, not raw logs**, so privacy stays intact and we still capture the bigger picture of what's happening across the network.

Furthermore, each independent agent **is enhanced with a lightweight LSTM policy trained with PPO**. As a result, each agent can learn and detect patterns on its own traffic. These per-agent models see both their local features and the graph embeddings, letting them spot unusual patterns that only make sense in context. In our tests on UNSW-NB15, this combination of personalized, recurrent policies plus shared graph signals showcased an increase in ROC-AUC, F_1 score, Accuracy, Precision and Recall metrics.

By exchanging only low-dimensional embeddings and binary alerts, never raw flows or metadata, we align with GDPR and EBSI's self-sovereign identity rules.

At the same time, we recognize several limitations that point to future work such as consolidate cold-start procedures, support dynamic topologies, and refine inter-agent protocols.

Looking ahead, this thesis lays down the foundation for a truly decentralized, privacy-preserving SIEM architecture. By combining online attention mechanisms with specialized multi-agent learning, we illustrate a path toward resilient, scalable network defense that fits perfectly into federated infrastructures like EBSI.

Building on these results, future efforts will loosen the limitations and bring us closer to a next-generation SIEM that scales out, not up.

REFERENCES

[1] R. Zuech, T. M. Khoshgoftaar, and R. Wald, "Intrusion detection and big heterogeneous data: a survey," *Journal of Big Data*, vol. 2, no. 1, p. 3, Feb. 2015. [Online]. Available: https://doi.org/10.1186/s40537-015-0013-4

[2] A. Oliner, A. Ganapathi, and W. Xu, "Advances and challenges in log analysis," *Commun. ACM*, vol. 55, pp. 55–61, 02 2012.

[3] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," 06 2008, pp. 111–125.

[4] D. Kasimatis, W. Buchanan, M. Abubakar, O. Lo, C. Chrysoulas, N. Pitropakis, P. Papadopoulos, S. Sayeed, and M. Sel, *Transforming EU Governance: The Digital Integration Through EBSI and GLASS*, 07 2024, pp. 250–263.

[5] E. Tan, E. Lerouge, J. Caju, and D. Seuil, "Verification of education credentials on european blockchain services infrastructure (ebsi): Action research in a cross-border use case between belgium and italy," *Big Data and Cognitive Computing*, vol. 7, p. 79, 04 2023.

[6] J. Velandia, A. Mesa, F. Rodríguez, D. Díaz-López, P. Nespoli, and F. Gomez Marmol, "Blocksiem: Protecting smart city services through a blockchain-based and distributed siem," *Sensors*, vol. 20, p. 4636, 08 2020.

[7] H. Bahsi and A. Levi, "Preserving organizational privacy in intrusion detection log sharing," in *2011 3rd International Conference on Cyber Conflict*, 2011, pp. 1–14.

[8] E. Tan and D. Seuil, *European Digital Infrastructure Consortium (EDIC): A New Governance Framework for the European Blockchain Services Infrastructure (EBSI)*, 03 2025, pp. 83–101.

[9] B. Ramos-Cruz, J. Andreu-Perez, and L. Martínez, "The cybersecurity mesh: A comprehensive survey of involved artificial intelligence methods, cryptographic protocols and challenges for future research," *Neurocomputing*, vol. 581, p. 127427, 2024.

[10] F. Menges, T. Latzo, M. Vielberth, S. Sobola, H.-C. Pöhls, B. Taubmann, and et al., "Towards GDPR-compliant data processing in modern SIEM systems," *Computers & Security*, vol. 103, p. 102165, 2021.

[11] A. P. V. ao, L. Santos, R. L. Costa, and C. R. ao, "Implementing and evaluating a GDPR-compliant open-source SIEM solution," *Journal of Information Security and Applications*, vol. 75, p. 103509, 2023.

[12] R. Lazzarini, H. Tianfield, and V. Charissis, "Federated Learning for IoT Intrusion Detection," *AI*, vol. 4, no. 3, pp. 509–530, 2023.

[13] E. Caville, W. W. Lo, S. Layeghy, and M. Portmann, "Anomal-E: A self-supervised network intrusion detection system based on graph neural networks," *Knowledge-Based Systems*, vol. 258, p. 110030, 2022.

[14] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[15] K. Kazari, E. Shereen, and G. Dan, "Decentralized Anomaly Detection in Cooperative Multi-Agent Reinforcement Learning," in *Proc. IJCAI*, 2023, pp. 162–170.

[16] J. Wu, G. Qiu, C. Wu, W. Jiang, and J. Jin, "Federated learning for network attack detection using attention-based graph neural networks," *Scientific Reports*, vol. 14, p. 19088, 2024.

[17] Z. Liu, D. Yang, S. Wang, and H. Su, "Adaptive multi-channel bayesian graph attention network for iot transaction security," *Digital Communications and Networks*, 12 2022.

[18] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *CoRR*, vol. abs/1703.03400, 2017. [Online]. Available: http://arxiv.org/abs/1703.03400

APPENDIX

TABLE III
SYSTEM-WIDE METRICS PER EPISODE

Episode	Setting	Precision	Recall	Accuracy	F1-score	ROC-AUC
1	With GAT	0.847	0.990	0.936	0.913	0.949
	Independent	0.793	0.977	0.906	0.876	0.923
	Shared	0.849	0.977	0.933	0.909	0.944
2	With GAT	0.845	0.984	0.933	0.909	0.946
	Independent	0.820	0.971	0.918	0.889	0.931
	Shared	0.846	0.988	0.935	0.912	0.948
3	With GAT	0.846	0.986	0.935	0.911	0.947
	Independent	0.827	0.976	0.923	0.895	0.936
	Shared	0.846	0.994	0.936	0.914	0.950

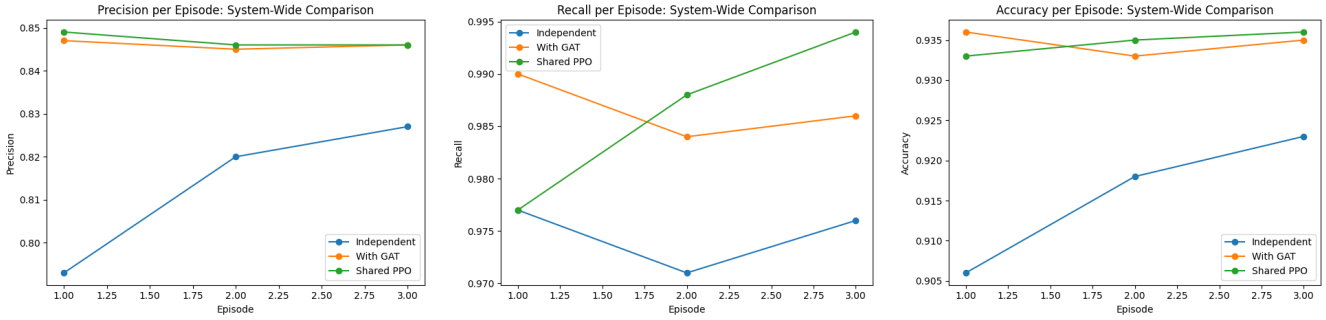

Fig. 3. Precision, Recall, and Accuracy over Episodes for independent and context-aware approaches.

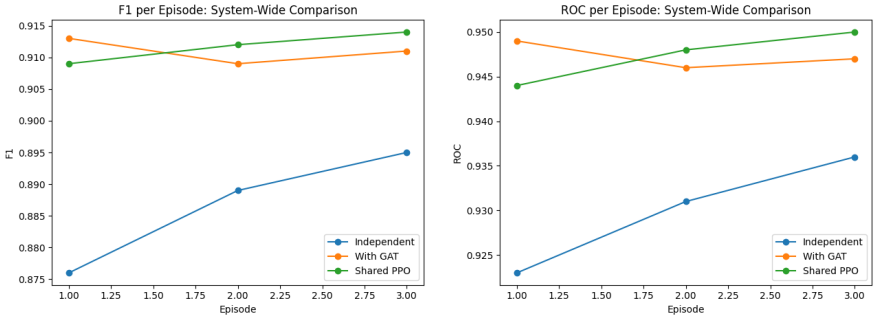

Fig. 4. F1-score and ROC-AUC over Episodes for both independent and context-aware approaches.

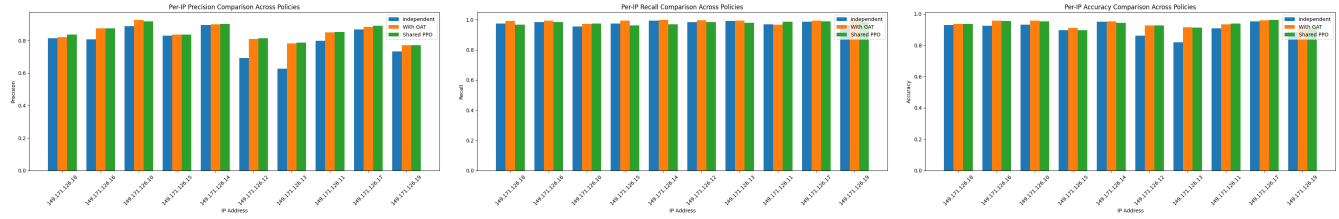

Fig. 5. Precision, Recall, and Accuracy per agent for both independent and context-aware approaches.

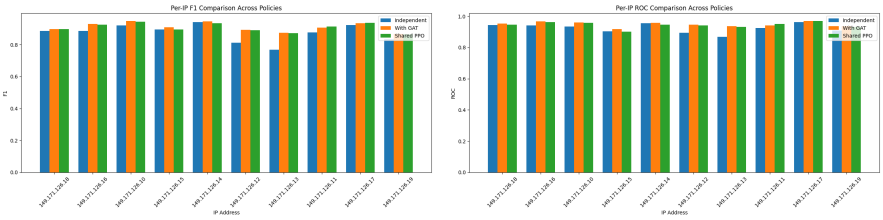

Fig. 6. F1-score and ROC-AUC per agent for both independent and context-aware approaches.

TABLE IV
PER-IP METRICS FOR INDEPENDENT IPS WITH ONLINE DGI (GAT) ACROSS EPISODES

	Episode 1					Episode 2					Episode 3				
IP (addr)	P	R	A	F_1	ROC	P	R	A	F_1	ROC	P	R	A	F_1	ROC
IP 0 (149.171.126.18)	0.820	0.992	0.936	0.898	0.953	0.817	0.976	0.931	0.890	0.945	0.820	0.986	0.935	0.895	0.950
IP 1 (149.171.126.16)	0.874	0.995	0.956	0.930	0.967	0.875	0.991	0.956	0.930	0.966	0.874	0.994	0.956	0.930	0.967
IP 2 (149.171.126.10)	0.927	0.973	0.957	0.949	0.960	0.919	0.990	0.960	0.953	0.965	0.923	0.994	0.964	0.958	0.969
IP 3 (149.171.126.15)	0.837	0.995	0.910	0.909	0.917	0.837	0.995	0.909	0.909	0.917	0.836	0.993	0.908	0.908	0.915
IP 4 (149.171.126.14)	0.899	0.999	0.952	0.946	0.958	0.897	0.947	0.932	0.921	0.934	0.897	0.938	0.928	0.917	0.930
IP 5 (149.171.126.12)	0.810	0.997	0.927	0.894	0.947	0.808	0.981	0.922	0.886	0.939	0.810	0.998	0.928	0.894	0.947
IP 6 (149.171.126.13)	0.782	0.994	0.914	0.875	0.937	0.774	0.987	0.909	0.867	0.931	0.775	0.995	0.911	0.871	0.935
IP 7 (149.171.126.11)	0.851	0.968	0.933	0.906	0.942	0.853	0.991	0.941	0.917	0.953	0.855	0.986	0.940	0.916	0.952
IP 8 (149.171.126.17)	0.884	0.993	0.960	0.935	0.970	0.887	0.996	0.962	0.939	0.972	0.896	0.993	0.965	0.942	0.973
IP 9 (149.171.126.19)	0.770	0.998	0.911	0.869	0.936	0.770	0.996	0.911	0.869	0.936	0.770	0.996	0.911	0.869	0.936

TABLE V
PER-IP METRICS FOR INDEPENDENT IPS (BASELINE) ACROSS EPISODES

	Episode 1					Episode 2					Episode 3				
IP (addr)	P	R	A	F_1	ROC	P	R	A	F_1	ROC	P	R	A	F_1	ROC
149.171.126.18	0.814	0.975	0.930	0.887	0.943	0.795	0.983	0.923	0.879	0.941	0.809	0.976	0.928	0.885	0.943
149.171.126.16	0.807	0.983	0.925	0.886	0.942	0.793	0.982	0.919	0.878	0.937	0.867	0.967	0.946	0.914	0.952
149.171.126.10	0.888	0.954	0.932	0.920	0.935	0.892	0.934	0.927	0.913	0.928	0.906	0.970	0.947	0.937	0.950
149.171.126.15	0.829	0.975	0.897	0.896	0.904	0.826	0.984	0.898	0.898	0.905	0.837	0.986	0.906	0.906	0.913
149.171.126.14	0.895	0.995	0.949	0.942	0.955	0.894	0.995	0.948	0.942	0.954	0.874	0.992	0.936	0.929	0.944
149.171.126.12	0.692	0.985	0.861	0.813	0.895	0.806	0.916	0.907	0.858	0.909	0.784	0.965	0.908	0.865	0.924
149.171.126.13	0.627	0.991	0.819	0.768	0.868	0.772	0.970	0.904	0.860	0.923	0.779	0.970	0.908	0.864	0.925
149.171.126.11	0.799	0.969	0.909	0.876	0.924	0.800	0.981	0.913	0.881	0.930	0.778	0.975	0.900	0.865	0.919
149.171.126.17	0.868	0.987	0.952	0.924	0.963	0.847	0.980	0.943	0.909	0.954	0.858	0.988	0.949	0.919	0.961
149.171.126.19	0.732	0.964	0.885	0.833	0.908	0.745	0.985	0.896	0.848	0.921	0.759	0.965	0.899	0.850	0.918

TABLE VI
PER-IP METRICS FOR SHARED PPO POLICY ACROSS EPISODES

	Episode 1					Episode 2					Episode 3				
IP (addr)	P	R	A	F_1	ROC	P	R	A	F_1	ROC	P	R	A	F_1	ROC
149.171.126.18	0.836	0.967	0.937	0.897	0.946	0.824	0.978	0.935	0.895	0.948	0.821	0.989	0.936	0.897	0.952
149.171.126.16	0.875	0.983	0.954	0.926	0.962	0.876	0.985	0.954	0.927	0.963	0.875	0.994	0.956	0.931	0.967
149.171.126.10	0.917	0.975	0.953	0.945	0.957	0.919	0.993	0.961	0.954	0.966	0.919	0.996	0.962	0.956	0.967
149.171.126.15	0.836	0.963	0.897	0.895	0.902	0.836	0.990	0.907	0.907	0.914	0.837	0.996	0.910	0.909	0.917
149.171.126.14	0.903	0.970	0.943	0.935	0.947	0.902	0.990	0.951	0.944	0.956	0.901	0.995	0.952	0.946	0.958
149.171.126.12	0.813	0.983	0.926	0.890	0.942	0.810	0.989	0.926	0.891	0.943	0.810	0.994	0.927	0.893	0.945
149.171.126.13	0.786	0.980	0.913	0.872	0.932	0.778	0.983	0.910	0.869	0.931	0.776	0.991	0.911	0.870	0.933
149.171.126.11	0.852	0.987	0.939	0.915	0.951	0.850	0.986	0.938	0.913	0.950	0.850	0.989	0.939	0.914	0.952
149.171.126.17	0.891	0.989	0.962	0.937	0.970	0.885	0.992	0.960	0.935	0.969	0.884	0.995	0.961	0.936	0.971
149.171.126.19	0.770	0.987	0.909	0.865	0.932	0.770	0.993	0.910	0.867	0.934	0.770	0.996	0.911	0.869	0.936