InDeKX: A High Efficiency Intrusion Detection and Knowledge Exchange Architecture for Secure Edge Computing Services

Author 1, Author 2, Author 3, Author 4
Institute Line 1
Institute Line 2
email-1, email-2, email-3, email-4

***Abstract*—The recent proliferation of Internet-of-Things (IoT) enabled devices in all forms of service deployments. Additionally, the edge computing paradigm brings short-term data and computation closer to end devices, allowing services to have lower latency and faster computations. However, such IoT-enabled edge services suffer from lack of security control, due to the highly distributed nature of such architectures. Moreover, incorporating centralized security controls limits the scalability of the intended decentralized nature of IoT-enabled services. In this paper, we present InDeKX, a novel hierarchical architecture for secure IoT-based edge services. The architecture incorporates a built-in Intrusion Detection System (IDS) along with knowledge propagation for security alerts within the system. We also present a comprehensive prototype implementation for InDeKX, built on top of EdgeX Foundry, an open-source platform for IoT device management, enhanced with microservices to run rule-based intrusion detection using Snort IDS. Henceforth, the prototype is utilized to perform exhaustive empirical evaluation using open-source dataset for IoT device service requests and a large array of varying network and intrusion conditions. The results illustrate high efficiency of functionality and security enforcement, depicting the feasibility of the proposed system in the real-world.**

***Keywords*-Edge Computing, Security, Intrusion Detection Systems (IDS), Internet-of-Things (IoT), Hierarchical Architecture, EdgeX Foundry, Snort**

I. Introduction

The Internet of Things (IoT) is a vast network of interconnected devices, seamlessly integrated into our environment. This enables smart objects to interact using standardized data formats[1, 2]. IoT applications span home automation, healthcare, smart cities, and industrial management, introducing new software development paradigms [3–5].

The rise of IoT devices in edge computing environments leverages enhanced performance and scalability. Edge computing brings processing closer to data sources, reducing latency and bandwidth use, and improving real-time analysis and response times [1, 6]. However, this decentralization introduces security challenges, including expanded attack surfaces, limited security resources, and the complexity of protecting diverse, distributed devices, all while ensuring data integrity, confidentiality, and availability [7, 8].

In this paper, we present InDeKX *(**In**trusion **De**tection and **K**nowledge e**X**change)*, a novel systematic framework for enabling secure and high-efficiency IoT-enabled edge computing services. The InDeKX architecture incorporates a hierarchical organization of edge computing elements, ensuring data flow and processing efficiency, with an integrated dynamic and run-time intrusion detection service (IDS) component, ensuring the distributed and decentralized security enforcement, within the framework. The hierarchical organization optimizes data flow and processing efficiency while ensuring the scalability of edge networks.

The presented work includes a fully-functional implementation of the proposed InDeKX architecture, by adopting EdgeX Foundry, an open-source edge computing IoT device management platform [9]. Additionally, a robust Intrusion Detection System (IDS) is integrated within EdgeX Foundry with Snort IDS to enhance real-time anomaly detection and allow enforcement of security controls [10]. InDeKX focuses not only on detecting common threats, such as Denial of Service (DoS) attacks and unauthorized access, but also, emphasizes the importance knowledge exchange among other components within the framework, while maintaining a balance between security, performance, and system manageability.

Contributions: The contributions of this paper are as follows:

1) We propose InDeKX, a hierarchical architecture, to allow run-time intrusion detection and dynamic knowledge exchange and propagation of intrusion alerts within the framework to ensure secure, yet efficient, IoT-enabled edge computing services, addressing the unique challenges posed by such highly distributed and decentralized architectures.
2) We illustrate a microservice-oriented integration for the proposed InDeKX architecture with EdgeX Foundry [9] and Snort IDS [10], to demonstrate the flexibility for adopting InDeKX towards practical applications.
3) We present exhaustive empirical evaluation and feasibility analysis of the InDeKX framework utilizing the fully-functional prototype implementation to validate the system's aptitude towards advancing safety in real-life circumstances for IoT-enabled edge computing services.

The rest of the paper is organized as follows. Section II presents the background and motivation for the proposed work. Section III presents the details of the proposed InDeKX architecture. Details of the prototype implementation and test-

bed is presented in Section IV, followed by the elaborations on the experimentation and analysis in Section V. The related work is presented in Section VI, and finally, the conclusion and future work are presented in Section VII.

II. Background and Motivation

Edge computing has emerged as a crucial paradigm in the IoT ecosystem, bringing computation and data storage closer to the sources of data generation [11]. This approach offers numerous advantages, that includes, reduced latency in data processing and decision-making, decreased bandwidth usage and associated costs, enhanced privacy through localized data processing, and improved reliability in scenarios with intermittent network connectivity [12, 13].

However, such environments are introduced with several security challenges [7, 8]. The nature of deployment results in an increased attack surface due to multiple entry points. Moreover, the heterogeneity of the terminal IoT devices make it difficult to implement uniform security policies across the framework. Overall resource constraints on edge devices limits the implementation of complex security measures, with potential for cascading failures if edge nodes in case of a compromised terminal device within the service network [12, 13].

These challenges create a pressing need for robust, decentralized security solutions that can effectively protect edge computing environments without compromising their inherent benefits. The problem statement addressing the proposed solution in our work is thus presented as follows:

Problem Statement: *The increase in integration of IoT devices into edge computing architecture has presented critical security vulnerabilities and performance challenges that has demanded a scalable and efficient security framework which can dynamically safeguard data while optimizing resource use across the network.*

The motivation behind the presented research in this paper therefore addresses the following critical security and performance efficiency concerns for edge computing services:

- *Increasing Attack Surface*: As the number of edge devices grows, so does the potential for security breaches. Each device represents a potential entry point for attackers, necessitating a distributed approach to the security.
- *Resource Constraints*: Many edge devices have limited computational power and memory, making it challenging to implement traditional security measures. There is a need for lightweight yet effective security solutions.
- *Real-time Requirements*: Many edge computing applications require real-time data processing and decision-making. Security measures must be able to detect and respond to threats quickly without introducing significant latency.
- *Heterogeneity of Devices*: Edge computing environments often consist of a wide variety of devices with different capabilities and operating systems. A flexible security framework that can accommodate this diversity is essential.
- *Data Privacy Concerns*: With increasing regulations around data privacy, there is a need for security solutions that can protect sensitive data at the edge, reducing the need to transmit such data to centralized servers.

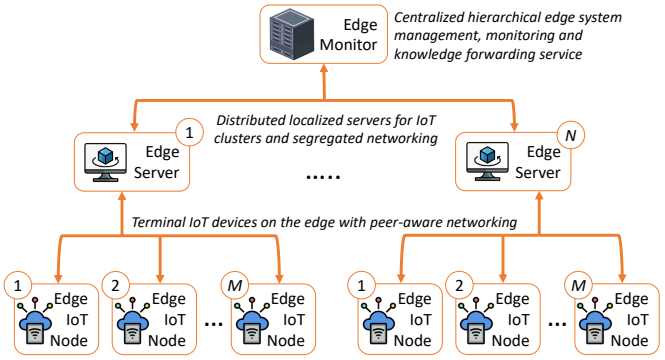

Fig. 1: InDeKX System Overview

Our research aims to bridge these gaps by developing a comprehensive security framework tailored to the unique challenges of edge computing. By focusing on early detection and prevention of unauthorized access and intrusions, we seek to enhance the overall security posture of IoT-driven systems while maintaining the performance benefits of edge computing.

III. InDeKX System Model

The InDeKX system architecture is comprised of multiple components, data elements, and communication protocols, and are explained in the following sub-sections.

A. System Overview

The proposed system model integrates various IoT edge computing architecture components to enhance security and operational efficiency. Our approach utilized a microservice-oriented design for various communication protocols to manage, secure, and circulate alert notifications across the framework. The work is presented through a structured explanation of each system component, their functions, interconnections, and the data flow between these entities.

The overview of the InDeKX architecture is illustrated in Figure 1. The system consists of three layers: Edge Nodes, Edge Servers, and an Edge Monitor. At the bottom layer, multiple edge IoT terminal nodes are connected to a dedicated Edge Server, which sends and receives data from the nodes within specific the segmented network cluster. In the middle layer, multiple such Edge Servers exist in the system, managing clusters of edge nodes. At the top layer, the Edge Monitor serves as the central component for overall system operations, handling the multiple Edge Servers. The Edge Servers are deployed in a localized manner. Moreover, the Edge Servers handle intrusion detection for the dedicated Edge Node cluster, identifying any malicious IoT terminal at run-time. If detected, the Edge Server then utilizes the integrated alert management microservices to handle alert knowledge propagation to the other Edge Nodes (non-malicious peers), as well as to the Edge Monitor for alert broadcasting to other Edge Servers.

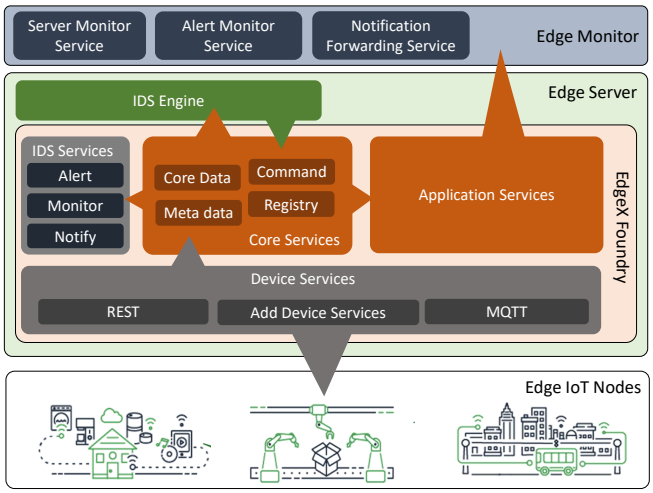

Fig. 2: InDeKX System Architecture

B. System Entities

The entities included in the InDeKX architecture are illustrated in Figure 2, and are described as follows.

Edge IoT Nodes: These devices are the primary data collection points in diverse environments such as homes, workplaces and community areas. They are crucial for capturing real-time operational data. They are essential for the distributed nature of edge computing, enabling localized data processing and reducing latency, which is vial for real-time application.

Edge Server: This act as a local processing hub, aggregating data from IoT devices, performing preliminary data analysis, and hosting the IDS engine for threat detection. This minimizes data transit to centralized clouds reducing bandwidth usage and latency, which enhances response times and operational efficiency in threat management.

Edge Monitor: This serves as the central oversight for the entire edge network monitoring system. The Edge Monitor ensures health, security, data integrity, and performance-efficiency for the overall InDeKX framework.

The Edge Server, in the middle layer, is responsible for maintaining communication flows with the Edge Nodes, as well as the intrusion detection and knowledge exchange operations. As such, we adopted EdgeX Foundry, an open source platform for IoT device management in edge computing systems [9]. The following elements describe the internal details of the Edge Server, including the relevant functional components of EdgeX Foundry.

IDS Engine: Integrated within the Edge Server, the IDS engine performs real-time intrusion detection by analyzing continuous traffic flows for unusual activities or known threat patterns. The modularity of the IDS engine allows a plug-and-play approach to integrate existing IDS platforms, which may be selected based on robustness and flexibility, making it ideal for adapting to the varying domain of IoT-enabled edge computing services. The IDS engine interacts with the EdgeX Foundry using microservice APIs.

Core Services: These are fundamental services provided by the EdgeX Foundry framework, including: a. Core Data: Responsible for data persistence and distribution within the system. b. Metadata: Manages information about the devices and sensors in the system. c. Command: Facilitates sending commands to devices and actuators.

Device Services: These components act as intermediaries between physical devices (nodes) and the core services. They translate device protocols into a common EdgeX Foundry data format.

Application Services: These services process and transform data, applying business logic and preparing data for export to external systems or for use by other services integrated with EdgeX Foundry.

Supporting Services: These include components that provide additional functionality to the system, such as, registry and configuration, scheduling, and activity logging. This EdgeX Foundry module is enhanced for functionality for our proposed InDeKX architecture, to enable notification and alerting mechanisms within the framework.

Security Services: These components manage the security aspects of the connected IoT devices for EdgeX Foundry, including a secret store to manage secrets and credentials, and an API gateway to provide a single entry point for external access to microservices.

System Management: This entity oversees the operation of other microservices for EdgeX Foundry, providing capabilities like health checking, metric collection, and remote management.

C. System Artifacts

The system artifacts for the InDeKX model are based on the entities described above and illustrated in Figure 2. Our system generates and utilizes several types of artifacts, and are described as follows:

Collected Data: Raw data gathered by IoT devices, which include sensor readings, user inputs, or environmental data.

Processed Data: Data that has undergone initial processing at the edge servers, including aggregated statistics or filtered information.

Network Traffic Metadata: Information about the patterns and characteristics of data flow within the network, used for anomaly detection.

Alerts and Notifications: Generated by the IDS Engine when potential intrusions or anomalies are detected.

Predefined Rule Sets: A collection of rules used by the IDS Engine to identify known attack patterns or suspicious activities.

System Logs: Detailed records of system activities, used for auditing and forensic analysis in case of security incidents.

D. System Architecture

Our proposed framework is designed to optimize data transmission and processing efficiency while maintaining a high level of security. The system architecture for InDeKX is outlined with respect to the following aspects:

Hierarchical Organization: Nodes are grouped and connected to dedicated edge servers, which in turn communicate with the central Edge Monitor. This structure allows for efficient data aggregation and localized processing.

Direct Communication: Nodes send data directly to their assigned Edge Servers, reducing network congestion, speeding up processing, and ensure network segregation.

Centralized Monitoring: The Edge Monitor receives processed data from all Edge Servers, providing a comprehensive view of the entire network.

Data Provenance: Network activity records are recorded to track the source and path of each piece of data, aiding in threat detection, enhancing accountability, and allowing auditability of post-event investigations.

Scalability: The layered organization of system components allow easy expansion of the service framework to accommodate the addition of new Edge Nodes or Edge Servers without reconfiguration requirements for the existing components.

Bidirectional Data Flow: The architecture supports two-way communication between Edge Nodes, Edge Servers, and the Edge Monitor, allowing seamless data collection, command and control, as well as intrusion knowledge exchange when applicable.

Integrated IDS Service: The IDS Engine is integrated into the Edge Server, allowing for segregated network threat detection, but, network-wide alert knowledge propagation. The modular design allows interchangeable IDS platforms to be placed, as per requirement for the overall edge service framework.

E. Registration and Initialization

Upon the introduction of a localized Edge Node to the InDeKX network, it establishes a connection with the assigned Edge Server. The Edge Server authenticates the Edge Node and registers it with the central Edge Monitor. Following successful registration, the node is provisioned with configuration parameters, encompassing security settings and communication protocols. Concurrently, the Edge Monitor updates the network topology and, if required, adjusts the IDS Engine rules to maintain optimal security.

F. Service Registration and Recording

When a new service (e.g., a device service or application service) comes online, the Edge Node registers with the EdgeX Foundry Registry and Configuration service. This registration process involves the metadata logging service on EdgeX Foundry, and logs essential metadata, including service type, version, capabilities, and network location. Registered services are integrated into regular health checks to ensure ongoing operational status. EdgeX Foundry enables dynamic service discovery, allowing other components to query the registry for available services. Additionally, the registry manages service versions, supporting updates and backward compatibility. Access control is enforced by EdgeX Foundry during the registration process to prevent unregistered Edge Nodes from interacting with the framework.

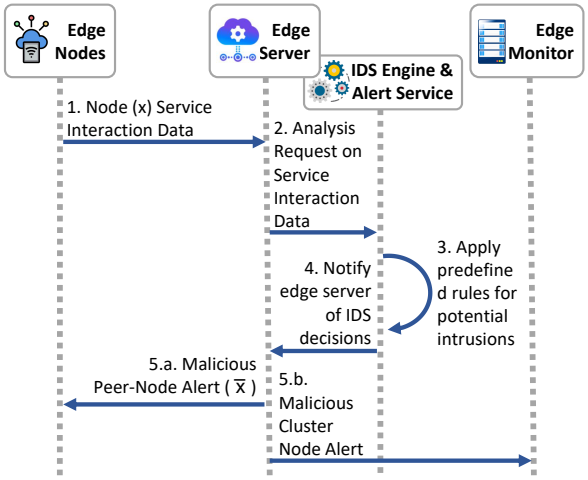

Fig. 3: InDeKX Communication Protocol Sequence

G. IDS Load balancing

To ensure the IDS Engine and other critical services stay responsive, InDeKX incorporates features for load balancing and resource allocation across the edge computing environment. Traffic analysis tools monitor and track the workload on each IDS instance, allowing the system to adjust traffic flow dynamically to avoid bottlenecks in any single instance. This approach ensures that the IDS Engine can keep high performance even during periods of high network traffic.

The IDS Engine implements continuous traffic analysis, monitoring network traffic patterns and processing loads across the dedicated Edge Server cluster. IDS Engine rules may be dynamically distributed and updated across instances to optimize detection capabilities and resource utilization. Load balancing thresholds are adaptively adjusted in response to overall system load and threat levels. Additionally, specific IDS instances may be allocated for specialized analysis tasks, such as deep packet inspection, depending on hardware capabilities. The system also maintains and reports key performance metrics, facilitating ongoing optimization of load balancing and overall system efficiency.

H. Data Collection and Transmission

The communication protocol sequence for data collection, transmission, and real-time analysis is illustrated in Figure 3.

The Edge Nodes continuously collect localized data from their environment or connected sensors, with the collected data undergoing pre-processing to eliminate noise and irrelevant information. The processed data is transmitted to the designated Edge Server via secure communication protocols. At the Edge Server, further processing on data aggregation and feature extraction is performed. The resulting data is

eventually forwarded to the central Edge Monitor for longer-term comprehensive analysis.

I. System Updates and Maintenance

The Edge Monitor periodically refreshes the IDS Engine rule-set based on the latest threat intelligence, which helps keep InDeKX's effectiveness against new threats. Edge Servers receive these updates and automatically apply patches to the IDS Engine to address significant vulnerabilities. Additionally, InDeKX conducts routine self-diagnostics using EdgeX Foundry management services to verify that all Edge Nodes are running correctly, allowing to promptly address any functionality and connectivity issues.

IV. Prototype Implementation and Test-Bed

To validate our proposed system model, we have developed a prototype implementation that integrates the key components of our security framework with the EdgeX Foundry platform. This section details the implementation specifics, the technologies used, and the experimental setup. Subsequently, we utilized the prototype to perform experimental evaluation of the proposed scheme, to justify the feasibility of InDeKX in the real world.

A. Prototype Implementation

The proposed system model for InDeKX has been used to develop a fully functional prototype implementation. The InDeKX framework consists of a networked system, interconnecting multiple simulated components. The developed entities were based on the system architecture, as illustrated in Figure 2, and explained in Section III. The InDeKX prototype for a comprehensive IoT security framework was built on the EdgeX Foundry platform [9], an open-source software platform, integrated with the Snort Intrusion Detection System (IDS) [10], and enhanced with custom security microservices. Our framework is designed to deliver real-time threat detection, streamline data processing, and ensure seamless device integration while keeping low latency and efficient resource use. All required entities and internal components were implemented in Java.

1) Adoption of EdgeX Foundry: We use EdgeX Foundry as the base edge computing platform due to its open architecture, modularity, and strong community support [9]. There are four service layers of EdgeX Foundry with two underlying system services as a part of the collection of microservices, security and system management. In our proposed framework, the core services provided by EdgeX foundry have been expanded with security components with the help of an expanded system service found within core services.

The Core Services work at the heart of the EdgeX Foundry implementation, which provide essential services like Core Data, Command, and Metadata, which are used by the integrated Snort-based IDS Engine. We utilize EdgeX Foundry's Device Services to enable communication between IoT devices and the Edge Server. These services translate various device protocols into a interoperable common data format for use by the core service layer and other microservices. Additionally, the Application Services for EdgeX Foundry are used to handling of real-time data processing and transformation within the InDeKX framework. We integrated our InDeKX microservices for anomaly detection, alert management, and knowledge exchange in the real-time generated by the Snort-based IDS Engine. The services are designed to be lightweight and efficient, ensuring that they don't introduce significant latency into the system.

2) Integration of Snort IDS: We integrated Snort IDS into the EdgeX Foundry framework, specifically within the Edge Server's IDS Engine, to perform real-time network traffic analysis [10]. Custom Snort rules along with existing rules have been created to detect edge-specific attack patterns, including data injection, denial-of-service (DoS) attacks, out-of-range data values, and unauthorized access attempts. With Snort, the system can quickly name and respond to threats, minimizing the potential impact on the network. With this addition, we were able effectively watch and flag the Edge Nodes that run outside their expected parameters.

3) Integration of Alert Manager: The Alert Manager handles alerts, ensuring that notifications for both critical and less critical threats are propagated throughout the InDeKX framework. We integrated this custom microservice within EdgeX Foundry to handle and forward alert knowledge generated by Snort-based IDS Engine within the InDeKX components in all layers of the hierarchical architecture. This service ensures that alert notifications are routed to Edge Monitor and other Edge Nodes within the given Edge Server's cluster.

4) Assurance of Secure Communication: All network communications for data transmission among Edge Nodes, Edge Servers, and the Edge Monitor are encrypted using TLS/SSL protocols. Secure communication channels are especially important in IoT environments, where data often travels over public or less secure networks.

5) Service and Data Flow Recording: We integrated a comprehensive provenance checking mechanism within the InDeKX framework into EdgeX Foundry to check the origin and source of every packet as it moves through the system. This tracking of data provenance is crucial for keeping accountability and improving the system's capability to detect and address anomalies. By keeping transparent documentation of data sources, the system is better equipped to recognize potentially malicious activities and respond accordingly. A data pre-processing module is deployed on each Edge Node to perform initial data cleaning and formatting before it is transmitted to the Edge Server. This step reduces the volume of data that is transmitted to the Edge Server, improving overall system efficiency.

6) Edge Server Analytics: Custom analytics modules are deployed on the Edge Servers to perform real-time data analysis and feature extraction. The data processing component aggregates data from multiple Edge Nodes, ignoring unusual fields and name patterns that may write down potential threats. The local processing capabilities of the Edge Servers ensure that the system can respond quickly to threats without relying

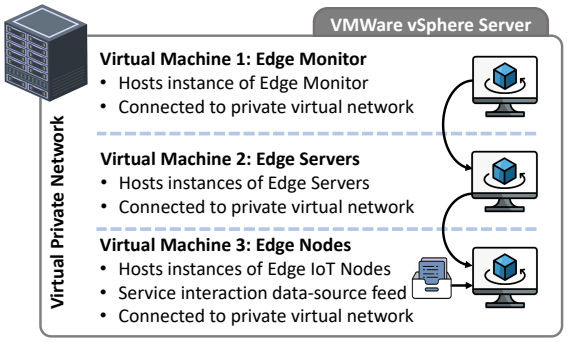

Fig. 4: InDeKX Test-Bed Experimental Setup

on centralized cloud resources. The analytics module can scale up with multiplying Edge Servers, depending on the number of connected Edge Nodes and cluster segmentation, ensuring that the system can handle increasing amounts of data without a significant drop in performance. The hierarchical deployment allows the system to manage larger volumes of data while supporting performance levels. Furthermore, these components are fine-tuned for efficient operation of Edge Servers, reducing their demand on centralized system resources and guaranteeing that the IDS Engine functions effectively, even in environments with limited resources.

B. Test-Bed Setup

To test our prototype, we created a controlled environment that simulates a realistic edge computing deployment. The test-bed architecture for the InDeKX prototype deployment is illustrated in Figure 4.

The virtual machine (VM) configuration utilized in this setup includes 2 CPU cores, 16GB of RAM, a 40GB HDD, and the Ubuntu 24.04 LTS operating system. We utilized three such VM instances, as shown in Figure 4. The first VM (VM-01), hosts the Edge Monitor, which is integrated as part of the application service in EdgeX Foundry. Its primary role is to receive notifications from the Edge Servers (VM-02), and display alerts accordingly. The second virtual machine (VM 02), serves as the Edge Server, incorporating essential components such as the EdgeX Foundry platform and the Snort IDS Engine. This VM acts as the central processing unit for edge computing tasks, leveraging EdgeX Foundry's core services to process incoming data, detect anomalies using Snort IDS, and generate alerts for any detected intrusions. Snort is configured to run on a dedicated port within VM 02. Custom rules are developed to detect edge-specific attack patterns. The third virtual machine (VM 03), simulates various IoT devices by hosting multiple Edge Node instances. We created software-simulated Edge Nodes that generate realistic IoT data streams. These entities are programmed to exhibit various behaviors, including normal operation and anomalous activities All communication uses the TCP/IP protocol suite.

V. Experimentation and Evaluation

We aimed to evaluate the performance of InDeKX with respect to the IDS Engine and knowledge propagation services, focusing on detection accuracy, notification timeliness, network traffic management, and overall system consistency. To achieve this, we devised 16 distinct scenarios, each standing for a combination of variable node counts and rate control, to ensure the system's scalability and reliability across varying conditions. The testbed setup is utilized to examine InDeKX's functionalities under normal conditions (continuous data flow without intrusions), simulated attacks (including data injection), while dynamically adding and removing Edge Nodes to assess scalability and performance efficiency of the proposed architecture. VM and EdgeX Foundry's system monitoring tools we used to track resource utilization, network traffic, and response times across all components.

A. Experimentation Dataset

We utilized the VARIoT Dataset of Legitimate IoT Data to simulate various types of sensor data. The dataset is modified to contain the desired number of Edge Nodes, based on unique device identifiers. Additionally, we also inject variable percentages of randomly placed anomalous data patterns to perform experimentation on the desired InDeKX functionalities.

B. Test Case Scenarios

The experimental design assessed how InDeKX manages varying complexities in edge computing, using different device counts and malicious traffic levels to simulate both simple and complex real-world scenarios. It aimed to evaluate InDeKX's detection capabilities, knowledge exchange speed, and impact on performance, including response time and resource usage.

Node Count (25, 50, 75, 100): The selected node counts, ranging from 25 to 100, reflect typical edge computing network sizes. Testing with these numbers helps assess InDeKX's scalability and performance from minimal to more substantial loads, providing insights into its robustness across varying network sizes.

Percentage of Malicious Devices (2%, 5%, 10%, 15%): The percentages of malicious devices, ranging from 2% to 15%, represent varying security threats in the edge network. Testing different levels allows us to evaluate InDeKX's detection capabilities under low (2% or 5%) and high (10% or 15%) threat intensities, assessing its sensitivity, accuracy, and performance under different attack pressures.

C. Measurements

We measured: the following parameters for the 16 scenarios (4 node counts × 4 malicious traffic percentages) to evaluate InDeKX's performance across different network complexities and security threats:

- Detection Capabilities: Snort's accuracy in identifying malicious and benign packets across different scenarios.
- Notification Timeliness: Latency of IDS Engine in notifying the Edge Monitor and affected Edge Server cluster.
- Network Traffic Management: Efficiency in managing data flow and mitigating bottlenecks with varying node counts and malicious traffic.
- Overall System Consistency: Stability in performance, including response times and resource usage, with varying system load.

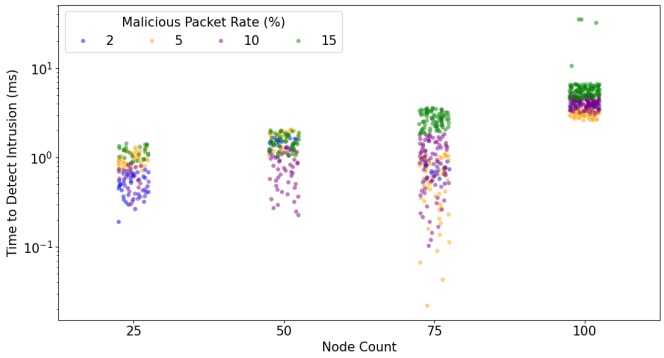

Fig. 5: Time To Detect Intrusion

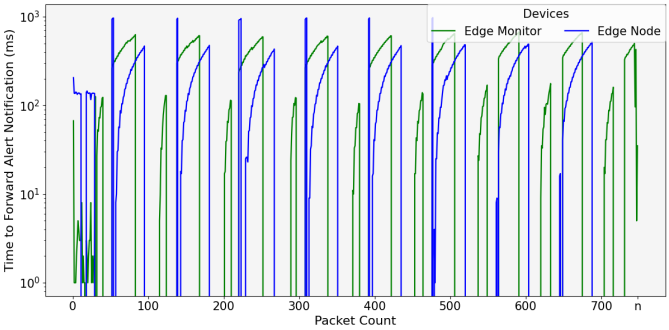

Fig. 6: Time to Propagate Intrusion Alert

D. Results and Analysis

1) Detection Capabilities: Figure 5 illustrates the 'Time to Detect Intrusion' in milliseconds as a function of the "Node Count." It features data points distinguished by four attack rates (2%, 5%, 10%, and 15%). The detection times are calculated based on the duration between the time when a particular packet is received by the Edge Server from an Edge Node, till when the Snort-based IDS engine API responds to the Edge Server's request for intrusion detection.

We observed most detections occur within 10 and 100 milliseconds for all cases. However, the spread of values suggests that Snort's performance may fluctuate more as the number of Edge Nodes increases. However, the effect of attack rates (2%, 5%, 10%, and 15%) is less remarkable. The results demonstrate the overall effectiveness of InDeKX in detecting intrusions across various node counts and attack rates. However, the variability in detection times, especially at higher node counts, suggests potential performance degradation under heavier loads. Nonetheless, the proposed InDeKX architecture of is designed to divide into smaller clusters whenever particular Edge Servers are growing in size without the additional overhead of system configuration, making the system easily scalable.

2) Timeliness of Notifications: Figure 6 shows the 'Time to Forward Alert Notifications' in milliseconds with respect to packet counts. The knowledge exchange for alert notifications occur when an intrusion is detected, and the notification is forwarded to the connected Edge Nodes and the Edge Monitor.

The graph reveals a pattern in notification times, characterized by peaks followed by drops to lower values as the packet count rises. This pattern likely signifies batch processing within the system, where a set of packets is processed together, leading to periodic spikes in notification time as the system handles the increased load. Notification times to the Edge Nodes are consistently higher compared to those for the Edge Monitor, due to the non-singular number of Edge Nodes within the cluster, in comparison to the single Edge Monitor. Despite the periodic spikes in notification times, the system shows an ability to recover and return to lower latency levels. This recovery is crucial for maintaining real-time responsiveness, especially in the context of intrusion detection. However, the presence of these spikes points to areas where the system could be optimized, particularly in communication and processing pathways, to minimize delays.

VI. Related Work

The paradigm of IoT-enabled services offer various opportunities to adopt edge computing systems towards enhancing operational performance and novel experiences [14]. Recent research has extensively explored various approaches to enhance security in edge computing and IoT frameworks [7, 8]. However, various approaches have been proposed to utilize edge computing for IoT-enabled services [15]. Wang et al. proposed physical layer techniques to bolster information security in heterogeneous IoT systems integrated with mobile edge computing [16]. Cazacu provided a comprehensive survey on IoT frameworks tailored for low-powered devices, highlighting the need for lightweight yet secure solutions [12]. Hagan et al. addressed security and privacy concerns in next-generation edge computing [17], whereas John et al. focused on Domain-Specific Languages (DSLs) for developing secure, interoperable systems [18]. Kovala explored edge computing platforms' potential for IoT [11], while Han et al. introduced an open framework for gateway monitoring in edge computing environments [13]. Ray et al. conducted a survey on edge computing in IoT, emphasizing future directions in e-healthcare [5]. Sharma et al. discussed data management for IoT edge-cloud architecture [19]. However, a unified approach addressing high-efficiency intrusion detection and knowledge exchange within secure edge computing remains underexplored. Our research contributes significantly by integrating comprehensive security measures across all layers of the network. Moreover, our research extends these concepts by not only enhancing edge computing capabilities but also incorporating adaptive security protocols that scale with device capabilities and threat levels, thus offering a more holistic and security-focused approach.

VII. Conclusion and Future Work

The proliferation of edge computing systems allows enhanced performance and scalability for IoT-driven services. However, security concerns arise from expanded attack surfaces, limited resources, and the complexity of protecting distributed devices while ensuring data integrity, confidentiality, and low latency.

The proposed InDeKX framework features a hierarchical architecture that optimizes data flow and functional efficiency. It includes a dynamic, run-time IDS for decentralized security. Utilizing EdgeX Foundry, the system integrates Snort for real-time anomaly detection and security control. InDeKX handles common threats, such as DoS attacks, and emphasizes knowledge exchange among components. The effectiveness of the proposed system is confirmed through extensive empirical evaluation utilizing the prototype implementation along with real-life IoT service dataset and varying network and threat scenarios.

Future research includes refinement of IDS rule sets for improved detection accuracy and exploration of machine learning techniques for adaptive security. Testing in larger, more varied IoT environments will be essential to confirm scalability, robustness, and optimization for extensive environments.

REFERENCES

[1] O. Uviase and G. Kotonya, "IoT architectural framework: connection and integration framework for IoT systems," *arXiv preprint arXiv:1803.04780*, 2018.

[2] A. Souri, A. Hussien, M. Hoseyninezhad, and M. Norouzi, "A systematic review of IoT communication strategies for an efficient smart environment," *Transactions on Emerging Telecommunications Technologies*, 2019.

[3] Business Intelligence, "Here comes the Internet of Things," Online at https://intelligence.businessinsider.com/the-internet-of-things-2013-10, 2013.

[4] S. N. Han, I. Khan, G. M. Lee, N. Crespi, and R. H. Glitho, "Service composition for IP smart object using realtime web protocols: Concept and research challenges," *Computer Standards & Interfaces*, vol. 43, pp. 79–90, 2016.

[5] P. P. Ray, D. Dash, and D. De, "Edge computing for internet of things: A survey, e-healthcare case study and future direction," *Journal of Network and Computer Applications*, vol. 140, 2019.

[6] C. Hu, X. Wu, and B. Li, "A framework for trustworthy web service composition and optimization," *IEEE Access*, vol. 8, pp. 73 508–73 522, 2020.

[7] H. Zeyu, X. Geming, W. Zhaohang, and Y. Sen, "Survey on edge computing security," in *Proceedings of the International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering*, ser. ICBAIE. IEEE, 2020.

[8] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data security and privacy-preserving in edge computing paradigm: Survey and open issues," *IEEE access*, vol. 6, 2018.

[9] "EdgeX Foundry: The Preferred Open Source Edge Platform," Online at https://www.edgexfoundry.org/.

[10] "SNORT: Open Source Intrusion Prevention System (IPS)," Online at https://www.snort.org/.

[11] J. Kovala, "Edge computing platforms for internet of things," Master's thesis, University of Helsinki, Helsinki, Finland, 2020.

[12] A.-R. Cazacu, "A survey of iot frameworks for low-powered devices," *Informatica Economica*, vol. 28, no. 2, 2024.

[13] K. Han, Y. Duan, R. Jin, Z. Ma, H. Rong, and X. Cai, "Open framework of gateway monitoring system for internet of things in edge computing," in *Proceedings of the International Performance Computing and Communications Conference*, ser. IPCCC. IEEE, 2020.

[14] R. Khan and R. Hasan, "DExaS: Delegated experience as a service for mobile and wearable devices," in *Proceedings of the 14th IEEE Annual Consumer Communications & Networking Conference*, ser. CCNC. IEEE, 2017, pp. 383–388.

[15] B. Gooder, R. Khan, P.-R. Adrian, and K. Sossoe, "CSRaaS: Composite service rendezvous as a service for iot-based smart environments," in *Proceedings of the Annual Computing and Communication Workshop and Conference*, ser. CCWC. IEEE, 2021.

[16] D. Wang, B. Bai, K. Lei, W. Zhao, Y. Yang, and Z. Han, "Enhancing information security via physical layer approaches in heterogeneous iot with multiple access mobile edge computing in smart city," *IEEE Access*, vol. 7, 2019.

[17] M. Hagan, F. Siddiqui, and S. Sezer, "Enhancing security and privacy of next-generation edge computing technologies," in *Proceedings of the International Conference on Privacy, Security and Trust*, ser. PST. IEEE, 2019.

[18] J. John, A. Ghosal, T. Margaria, and D. Pesch, "DSLs for model driven development of secure interoperable automation systems with edgex foundry," in *Forum on specification & Design Languages*, ser. FDL. IEEE, 2021, pp. 1–8.

[19] G. Sharma, N. Hemrajani, S. Sharma, A. Upadhyay, Y. Bhardwaj, and A. Kumar, "Data management framework for iot edge-cloud architecture for resource-constrained iot application," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 25, no. 4, 2022.