Federated Detection of KNX Attacks with Network Flow Statistics in Industrial Environments

Christos Dalamagkas[†‡], Panagiotis Radoglou-Grammatikis[‡**], Antonios Sarigiannidis[‡], Evangelos K. Markakis[§], Michail-Alexandros Kourtis[††], Thomas Lagkas[†], Georgios Th. Papadopoulos[¶], Vasileios Argyriou[‖] and Panagiotis Sarigiannidis[**]

***Abstract*—The convergence of Information Technology (IT) and Operational Technology (OT) domains bring increased inter-connectivity for cyber-physical systems, allowing them to be remotely monitored and controlled. Moreover, modern requirements of energy efficiency encourage organizations and individuals to deploy smart building automation systems, leading not only to the development of new protocols but also to the resurface of older protocols, such as the KNX standard. However, considering the emergence of cyber-threats across the IT/OT domains, it is critical for infrastructure owners to monitor their assets effectively and detect potential malicious behaviors. In this work, we propose a Network Intrusion Detection System (NIDS) that is capable of detecting multiple malicious activities against KNX systems, which are carried out over the Internet Protocol (IP) network. We propose an analysis based on network flows that are enhanced with application-layer statistics, thus improving detection accuracy and allowing better multi-class classification. We compare two Machine Learning (ML) models trained with CICFlowMeter and our solution, KNXFlowMeter, showcasing around 20% improvement across classification performance metrics.**

***Index Terms*—KNX, anomaly detection, network flows, network intrusion detection system**

* This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No, 101225719 (SALUS - Strengthening law enforcement with advanced IoT forensic tools and enriched investigation schemes, realised by a Software Defined Network Security-as-a-Service architecture) - Disclaimer: Funded by the European Union. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

† C. Dalamagkas and T. Lagkas are with the Department of Informatics, Faculty of Science, Democritus University of Thrace, EX B14, Kavala Campus, Kavala, Greece (e-mail: cdalamagkas@cs.duth.gr, tlagkas@cs.duth.gr).

‡ C. Dalamagkas, P. Radoglou-Grammatikis and A. Sarigiannidis are with K3Y Ltd, Studentski District, Vitosha Quarter, Bl. 9, 1700 Sofia, Bulgaria - E-Mail: cdalamagkas@k3y.bg, pradoglou@k3y.bg, asarigia@k3y.bg

§ E. K. Markakis is with the Department of Electrical and Computer Engineering, Hellenic Mediterranean University, P.O. Box 1939, GR-71004 Heraklion, Crete, Greece - E-Mail: emarkakis@hmu.gr

†† M. A. Kourtis is with the Institute of Informatics and Telecommunications, National Centre for Scientific Research "Demokritos", P.O. Box 60037, 15310 Aghia Paraskevi, Greece - E-Mail: akis.kourtis@iit.demokritos.gr

¶ G. Th. Papadopoulos is with the Department of Informatics and Telematics, Harokopio University of Athens, 9 Omirou St., Tavros, GR-17778, Athens, Greece - E-Mail: g.th.papadopoulos@hua.gr

‖ V. Argyriou is with the Department of Networks and Digital Media, Kingston University, London, U.K. - E-Mail: vasileios.argyriou@kingston.ac.uk

** P. Radoglou-Grammatikis and P. Sarigiannidis are with the Department of Electrical and Computer Engineering, University of Western Macedonia, ZEP Campus, 50100, Kozani, Greece - E-Mail: pradoglou@uowm.gr, psarigiannidis@uowm.gr

I. Introduction

Driven by the modern society's needs for increased energy efficiency, reduced operational costs and enhanced occupant comfort, there has been increased deployment of building automation systems. Moreover, European policymaking initiatives have also supported the deployment of smart technologies in buildings, such as the Smart Readiness Indicator (SRI), introduced under the Energy Performance of Buildings Directive, which rates the capabilities of buildings to optimize energy efficiency and adapt not only to the needs of the occupants but also to signals from the grid [1].

To achieve building automation, the convergence of Information Technology (IT) and Operational Technology (OT) is a critical factor, allowing physical entities such as temperature sensors, actuators, Heating, Ventilation, and Air Conditioning (HVAC) systems to be accessed remotely through IT technologies, and to be integrated into cloud platforms and Internet Protocol (IP)-based networks. One of the dominant IT/OT protocols in the building automation sector is KNX. This protocol covers the entire networking stack, from the physical layer to the application layer, and is standardized as International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC) 14543. KNX can support several physical communication media, such as twisted pair (TP), power-line, radio and IP, known as KNXnet/IP. The KNXnet/IP medium is critical for enabling interoperability of the KNX devices with the IT domain, as it allows KNX telegrams, usually transmitted on the TP-based KNX bus, to be encapsulated into Ethernet network packets, through a KNX interface, and to be consumed by IoT platforms residing on the IT realm, such as OpenHAB [2].

However, cybersecurity remains a significant concern for cyber-physical systems and building automation systems. Adversaries can exploit vulnerabilities and design flaws of the underlying OT protocols (e.g., lack of encryption and authentication or authorization) in order to take control of devices, cause disruption or conduct advanced cyber warfare. Considering the introduction of physical media such as the KNXnet/IP, the threat surface increases as well. Indicatively, M. Küppers et. al [3] performed a thorough security analysis on KNX, by performing a set of security tests against the Engineering Tool Software (ETS), the official device commissioning and project management software from the KNX association, and

against the KNX bus. In particular, the authors successfully performed reconnaissance attack, by eavesdropping the unencrypted KNX telegrams, uncovering the system topology, addresses and measurements. Next, by exploiting the inherent lack of authentication and authorization on KNX, the attackers were able to reply packets and perform packet injection to manipulate established automations. A data exfiltration attack was demonstrated by Graveto et. al [4], showcasing how physical attack vectors can be exploited (e.g., by using USB ports) to get access to sensitive data. Two exfiltration methods are described, a) injecting the sensitive data into the payload of KNX telegrams, and b) encoding sensitive data in binary encoding, e.g. Morse code, and manipulate the blinking of existing luminaries to transmit this data.

The cyber risk associated with insecure KNX infrastructures is highlighted through the risk assessment presented by J. Goltz et al [5]. The authors note the lack of important security controls in KNX, such as client authentication, data confidentiality / integrity / verification, and the ability to perform updates over the air (OTA). Through a static risk analysis, four categories of dangers are defined, and several classes for each danger that indicate the level of exposure. The static risk analysis gives an overall risk of about 67.19%. Considering the aforementioned, an open challenge remains to classify the network activity and timely detect potential suspicious activities. Motivated by the above, the contribution of this paper is summarized as follows:

- Inspired by the literature, we present eight attacks against KNXnet/IP, including a detailed description of the messages exchanged between the participating KNX assets. From this analysis, we obtain useful insights about the features that could assist in classification and anomaly detection.
- We introduce the KNXFlowMeter tool, a flow-based detection method that produces enhanced network flow statistics, including application layer information related to KNX.
- We utilize the dataset generated by KNXFlowMeter to train a neural network through Federated Learning (FL), and by comparing its performance with an identical model trained with CICFlowMeter data and different FL aggregation techniques, we showcase how the features of KNXFlowMeter lead to increased Artificial Intelligence (AI) related performance metrics.

II. Related Work

In this section, the relevant work is discussed in terms of methods developed to detect threats against KNX.

M. Peters et al. present a Machine Learning (ML)-based IDS that leverages multiple classification algorithms to detect outliers on KNX traffic, particularly the k-nearest neighbor, Local Outlier Factor (LOF), Support Vector Machines (SVM), and entropy [6]. To reduce the amount of data that needs to be processed, the authors aggregate KNX traffic into network flows, which are characterised by 7 KNX features (seconds of the week, source/destination KNX addresses, priority, hop count, payload length, and Application-layer Protocol Control Information / APCI code). The proposed IDS applies binary classification, and is used to detect various abnormal activities, such as network scanning, the onboarding of a new device, and Denial of Service (DoS) attacks. Concluding, the authors highlight the loss of information caused due to aggregation of KNX traffic into flows, considering that some simple intrusion attempts that depend on a small amount of telegrams were undetectable.

V. Graveto et al. present a hybrid Network Intrusion Detection System (NIDS) for KNX systems [7]. The proposed NIDS combines rule-based methods, statistical analysis and AI-based classification to detect potential attacks as well as to distinguish between normal and malicious network traffic. The authors developed four reference attack scenarios, particularly, a line scan attack with the `TConnect` and `DeviceDescription` request messages, a device info attack which utilizes the line scan attack messages followed by `MemoryRead` and `ADCRead` requests, a message injection attack with a high rate of valid `GroupValueWrite` messages, and an invalid context attack. For the AI-based detection, 9 features were considered, including inter-arrival time, number of messages per second, message size, and basic KNX attributes, which are extracted from deep packet inspection. The evaluation results showcased high performance of the AI methods, particularly the Multilayer Perceptron (MLP) neural network which achieved 100% accuracy.

Finally, Cash et al. focus on detecting a false injection attack that alters temperature sensor data injected into a building automation system, impacting the system in terms of energy cost [8]. In particular, the authors adopt the Jensen Shannon Divergence (JSD) method to measure the similarity of interarrival time distributions between KNX telegrams, in attack and benign conditions. Experimentation results indicate 100% detection accuracy with an SVM-based ML model.

III. KNX Testbed

For the experiments and attacks designed in this work, we assumed the existence of the KNX testbed depicted in 1. The infrastructure consists of a KNX bus that is powered by a KNX power supply. Moreover, an Air Condition (AC) gateway allows an indoor AC unit to be accessed through the KNX bus, allowing the reading of variables such as room temperature. On the Ethernet network, the OpenHAB software obtains measurements from the KNX bus in real-time. The KNX/IP interface is the critical component that bridges the KNX and the Ethernet networks, by creating virtual tunnels for encapsulating KNX and common External Message Format (cEMI) traffic into KNXnet/IP packets that are routable in the Transmission Control Protocol (TCP)/IP network. The attack scenarios assume that an attacker is able to access the local Ethernet network, having direct network access to the KNX/IP interface. Through network scanning, the attacker can discover the KNX/IP interface as well as the AC gateway that resides on the KNX bus.

IV. Threats against KNXnet/IP

In this work, we introduce a network IDS that is able to distinguish among different attacks. This section is dedicated

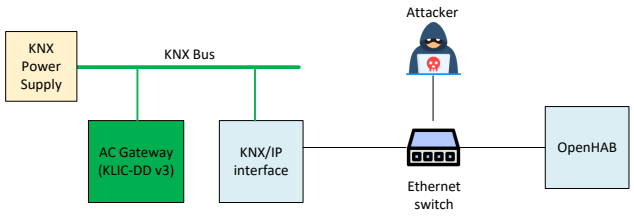


Fig. 1. The KNX Testbed

into analyzing each of the attacks considered in the training process. Each attack is characterized by a unique identifier along with a descriptive title. For each attack, we break down the steps that the attacker follows, giving us insights on what KNX-related features could be leveraged to identify the malicious activities. All the attacks can be found in our Gitlab repository [9].

A. A.KNX.01 - M_PropRead.req Fuzzing

C. Vacherot [10] implements a fuzzing attack, in which the attacker sends mutated `M_PropRead.req` cEMI packets, with randomized values on randomly chosen fields. In particular:

1) The attacker initializes the `M_PropRead.req` message to be mutated during the fuzzing process.
2) The attacker sends a KNXnet/IP Tunnel Connection Request for Device Management (i.e., connection type 0x03) to the KNX/IP interface.
3) The KNX/IP interface accepts the connection request.
4) The attacker sends a malformed `M_PropRead.req`, by setting random bytes to the values of all cEMI fields of `M_PropRead.req`, excluding `message_code` and the cEMI payload.
5) The KNX/IP interface replies with a KNXnet/IP Configuration Acknowledgment (ConfigAck), confirming that it received the `M_PropRead.req`.
6) Subsequently, the KNX/IP interface also replies with an `M_PropRead.con`, which includes the value of the memory address requested.
7) The attacker sends another malformed `M_PropRead.req`, repeating steps 4-7.

Observations: The traffic pattern indicates an unusually high number of M_PropRead messages, as well as a variety of different object type and object instances that the attacker tries in the M_PropRead messages. This pattern could indicate a reconnaissance, fuzzing or DoS attempt.

B. A.KNX.02 - Device Reset

In this simple attack, the attacker sends a `M_Reset.req` message to the KNX/IP interface each 125 msec. The message forces the KNX/IP gateway to close all KNX transport layer and KNXnet/IP connections. Despite its disruptive effect, it is noteworthy that no authentication or authorization is applied by the KNX/IP gateway upon receiving and applying the reset command. **Observations**: The number of `M_Reset.req` messages in a given flow may indicate a suspicious activity, considering that these messages are not expected during normal operation. A high amount of those messages may also indicate a DoS attempt.

C. A.KNX.03 - Network scanning and identification

The attacker sends multicast and description request messages to the TCP/IP network in order to discover KNX/IP interfaces. The information can be leveraged by the attacker in order to further penetrate into the underlying KNX buses. The attacker follows the steps bellow:

1) The attacker sends a KNX/IP Search Request at 224.0.23.12, which is the multicast address on which all KNX/IP interfaces are listening to.
2) The targeted KNX/IP interface, that listens to this address, responds with a KNX Search Response.
3) Following the response from the KNX/IP interface, the attacker sends a Device Management Connection Request directly to the device, and the KNX/IP interface responds positively.
4) The attacker sends a KNX Description Request message to the KNX/IP interface.
5) The KNX/IP interface responds with a KNX Description Response, and the attacker sends a KNX Disconnect Request to the device.

Observations: Similarly to A.KNX.02, the number of KNX/IP Search Requests and Description Requests are indicators of potential malicious activity, particularly a reconnaissance attempt.

D. A.KNX.04 - KNX bus scanning and identification

After discovering the KNX/IP interfaces through A.KNX.03, the attacker implements line scanning on the identified KNX/IP interfaces, aiming to discover all KNX devices connected to the KNX buses. For this purpose, the attacker follow the steps bellow:

1) The attacker generates a pool of random KNX Individual Addresses (IA) (i.e., unique address of the KNX device in the form 1.1.X), which will be used later to probe for KNX devices.
2) The attacker sends a Tunnel Connection Request (`ConnectReq`) to the KNX/IP interface. The KNX/IP interface responds with a `ConnectResp` message, assigning an available IA to the attacker.
3) The attacker chooses an IA from the pool, and follows the steps bellow:
   a) The attacker sends a tunneled cEMI connection request, in order to establish a transport-layer session (i.e, `TunnelReq` that encapsulates a cEMI `L_Data.req` specifying a Transport Layer Control Information (TPCI) Connect service).
   b) The KNX/IP gateway replies with a corresponding `L_Data.con` message, and the attacker replies with a tunneling acknowledgment (`TunnelAck`) message.
   c) Next, using the selected IA, the attacker sends a tunneled cEMI device description read request

(i.e., a TunnelReq that encapsulates a cEMI L_Data.req specifying the DevDescrRead ACPI service).
   d) The KNX/IP gateway accepts the request, by replying with a L_Data.con message, which the attacker confirms with a TunnelAck message.
   e) The attacker waits to receive a cEMI acknowledgment from the probed IA (i.e., to receive a cEMI L_Data.ind specifying the TPCI ACK service). If this happens, then the relevant IA corresponds to an actual KNX device. Concurrently, the attacker also receives the Device Description Response (i.e., cEMI L_Data.ind specifying the DevDescrResp APCI service), thus getting more information about the KNX device residing on the bus. Both messages are acknowledged by TunnelAck messages.
   f) Next, the attacker sends a tunneled cEMI acknowledgment message (i.e., a TunnelReq that encapsulates a cEMI L_Data.req specifying the TPCI ACK service). The KNX/IP gateway replies with the respective L_Data.con message, and the attacker confirms with a TunnelAck message.
   g) Finally, the attacker sends a tunneled cEMI disconnection request to terminate the transport-layer tunneling session (i.e., a TunnelReq that encapsulates a cEMI L_Data.req specifying a TPCI Disconnect service). The KNX/IP gateway replies with the respective L_Data.req message, and the attacker confirms with a TunnelAck message.
   h) The steps of this block are repeated for each IA of the pool.
4) When all individual addresses are scanned, the attacker sends a KNX/IP Tunnel Disconnect request to the KNX/IP interface, in order to terminate the KNXnet/IP tunnel.

Observations: The activity of this attack is characterized by multiple requests of DevDescrRead ACPI service type in a short amount of time, which are destined to multiple IAs. Useful observations are the number of DevDescrRead cEMI packets, as well as the number of different IAs observed in a given timeframe.

E. A.KNX.05 - Flooding DoS with valid packets

After identifying the targeted KNX devices from A.KNX.04, the attacker aims to overwhelm the victim KNX device by sending multiple valid KNX messages in the form of Heatbeats, aiming to overwhelm the KNX/IP interface, render it unresponsive and cause potential disruption. In our reference KNX infrastructure, the Heartbeat messages are in the form of GroupValueWrite messages at the 0/0/2 group address. The attack is implemented by following the steps bellow:

1) The attacker sends a Tunnel Connection Request to the KNX/IP interface.
2) Given a positive reply, the attacker sends a Tunneling Request (TunnelReq) that encapsulates a cEMI GroupValueWrite message. The message is addressed to 0/0/2 with the value of 1, to indicate a heartbeat message.
3) Upon receiving an L_Data.con confirmation, the attacker sends a tunneling acknowledgment (TunnelAck).
4) The attacker sends a Tunneling Disconnect request, to terminate the tunneling session.
5) Immediately, the attacker starts the same procedure by going to step 1.

Observations: Given that this attack does not generate any errors or specific responses, observing the number of GroupValueWrite messages in a specific time window would indicate a potential flooding DoS attack.

F. A.KNX.06 - Flooding DoS with partially invalid packets

As a variation of A.KNX.05, the attacker aims to overwhelm the victim KNX device by sending partially invalid KNX messages. These messages are in the form of Heatbeats (i.e., GroupValueWrite messages with a random payload), however, in this case the attacker randomly alters both the source IA address and destination group address in each attempt. The attack is implemented by following the steps bellow:

1) The attacker chooses a random source IA and destination group address.
2) The attacker sends a Tunnel Connection Request to the KNX/IP interface.
3) Given a positive reply, the attacker sends a Tunneling Request (TunnelReq) that encapsulates a cEMI GroupValueWrite message. The message is addressed to the random destination group address with a random payload.
4) Upon receiving an L_Data.con confirmation, the attacker sends a tunneling acknowledgment (TunnelAck).
5) The attacker sends a tunneling disconnect request, to terminate the tunneling session.
6) Immediately, the attacker starts the same procedure by going to step 1.

Observations: Similar to A.KNX.05, the attack generates an unusual amount of GroupValueWrite messages, however, in this case, we can also observe the amount of unique IAs addressed in the KNX messages. A high amount of unique IAs further indicates a malicious activity.

G. A.KNX.07 - Flooding DoS with GroupValueRead

As a variation of A.KNX.06, the attacker aims to overwhelm the victim KNX device by sending partially invalid KNX messages. In this case, the attacker keeps the same strategy of altering source and destination addresses, but they send GroupValueRead messages instead, aiming at extracting information from the KNX devices (e.g., state information or measurements). The same observations from A.KNX.06 are also applicable in A.KNX.07.

H. A.KNX.08 - Unauthorized Access

Most of the operations inside a KNX system is performed through `GroupValueWrite` messages. During the programming phase of a KNX network with the ETS tool, the KNX programmer assigns communication objects of the KNX device to particular group addresses. For example, a group address, in the form of 0/0/X, can be programmed for reporting the room temperature, while another in the form of 0/0/Y, can be used for changing the target temperature or opening/closing the device.

Given there is lack of authentication/authorization on the KNX messages, an attacker can send `GroupValueWrite` messages to sensitive group addresses that are related to critical operations of the automation infrastructure (e.g., to raise the target temperature of the indoor AC unit, thus endangering the systems of a computer room).

In this scenario, the attacker sends `GroupValueWrite` to the group address 0/0/54, which is programmed to close the AC indoor unit of the computer room.

Observations: In contrast to the aforementioned attacks, no significant statistical observations can be made for A.KNX.08. In this case, a rule-based approach would be more appropriate, as in a given KNX setup, it is common to know before-hand all the KNX group addresses and whether they are expected to appear in the network.

V. Proposed Intrusion Detection System

From the attacks mentioned in section IV, useful statistical observations were made, which can guide the development of an AI-based detection method that ”learns” the normal behavior as well as the malicious patterns. Given that, the architecture of the proposed IDS is presented in Fig. 2 as a C4 component diagram. The architecture is based on our previous work [11], and consists of two main containers, namely the Network IDS, that also acts as a local FL client, and the FL server that aggregates the local contributions from multiple FL clients to construct a global model. The components of each container are discussed in the following subsections.

A. Network IDS / FL client

1) Network Traffic Capture: This component undertakes to capture the raw network traffic from the network. Assuming that there is a port mirroring configuration that copies network traffic from the KNX/IP interface to this component, `tcpdump` is used to capture the mirrored network traffic in a periodic manner, according to the predefined flow timeout (i.e., the time duration of a network flow).

2) Network Flow Generation: This component is responsible for calculating network flow statistics from the captured PCAP files. In particular, network packets are aggregated into network flows, and the packets of each flow are parsed, from the network layer up to the application layer, in order to calculate relevant statistics. The component uses two alternative solutions for the flow generation: CICFlowMeter [12] and KNXFlowMeter. CICFlowMeter is a baseline tool that generates statistics limited to the network and transmission layers, whereas KNXFlowMeter - introduced in this work - extends the calculation of statistics up to the application layer of KNXnet/IP and cEMI. Given that the KNXnet/IP traffic is encapsulated in User Datagram Protocol (UDP) datagrams, KNXFlowMeter covers the network layer, transport layer, KNXnet/IP layer, and cEMI layer. All the 104 features of KNXFlowMeter are documented in the KNXnet/IP Intrusion Detection Dataset [13]. A critical parameter for both flow generation solutions is the flow timeout, which is a static parameter that is given by the user once when launching the IDS. The flows are determined from the IP and UDP attributes, and given that UDP is a stateless protocol, the flow timeout is the main parameter that determines the first and last packet of a network flow, along with the source and destination network sockets. Output of this component are the network flows contained within the given PCAP file.

3) Inference Module: Receiving the flows from the Network Flow Generator, this component applies a pre-trained ML classification model in order to classify the flows as malicious or normal. The model is an MLP neural network that is pre-trained with benign traffic as well as with labeled network flows from the attacks described in section IV.

4) Local Training Module: This component trains the ML classification model according to the instructions and hyper-parameters received from the FL server. Moreover, it provides feedback to the FL server regarding the model updates, accompanied with training metadata and overall status.

5) Security Event Normalizer & Dispatcher: The labeled flows are transformed to a common schema indicating a cyber security event, and the event is transmitted through the appropriate means (e.g., Syslog, Kafka etc).

B. FL Server

Finally, the FL server is responsible for orchestrating the training process, by passing instructions and metadata to the local FL clients. Moreover, the server is responsible for constructing the global ML model, by considering the contributions of the local FL clients and applying an aggregation strategy. In this work, we present performance results from different FL aggregation strategies.

VI. Experimental Results & Discussion

To study the performance of the proposed IDS, the attacks described in section IV were implemented against a real KNX/IP infrastructure, depicted in Fig. 1, which was realized in laboratory environment.

Two ML models were trained for this experiment. The first one was trained with a dataset from the CICFlowMeter tool, which was used as a baseline, and the second ML model was trained with a dataset generated by the KNXFlowMeter. Both CICFlowMeter and KNXFlowMeter received the same input in terms of network traffic, particularly a mix of benign and malicious network traffic, as it was obtained from the infrastructure of Fig. 1 while executing the attacks described in section IV. The purpose of developing two ML models is to study how the features provided by our implementation improves the detection performance.

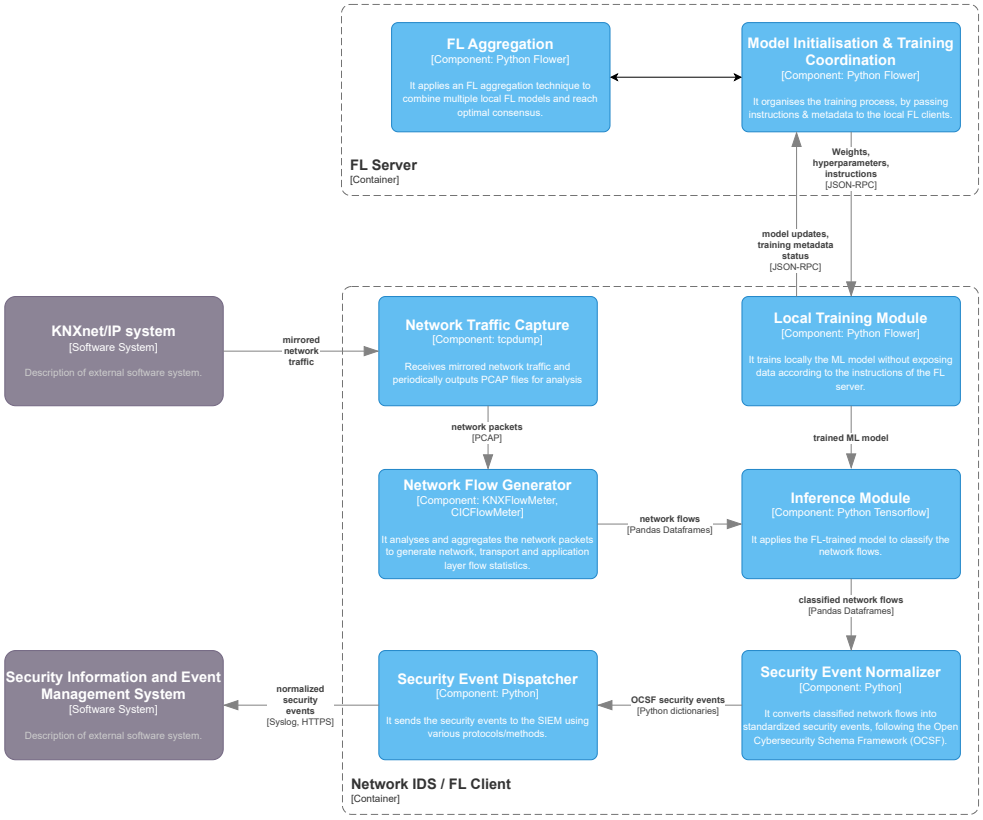


Fig. 2. The C4 Component Diagram of the Proposed IDS

For both cases, multiple FL aggregation techniques were tested, namely FedAvg [14], FedProx [15], FedAdam [16], FedAdagrad [16], and FedYogi [17]. For each FL aggregation technique, typical evaluation metrics of classification performance were obtained, particularly Accuracy (ACC), True Positive Rate (TPR), False Positive Rate (FPR), F1 score, and Area Under Curve (AUC). The metrics are defined in equations 1-5.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$TPR = \frac{TP}{TP + FN} \tag{2}$$

$$FPR = \frac{FP}{FP + TN} \tag{3}$$

$$F1 = \frac{2 * TP}{2 * TP + FP + FN} \tag{4}$$

$$AUC = \int_0^1 TPR(FPR)dFPR \tag{5}$$

Table I presents the evaluation results obtained from the classifier that was pre-trained with CICFlowMeter. It can be observed that FedAvg performs better in all metrics, however, the model still underperforms in general, by reaching at maximum 75,53% of accuracy. The low performance can be attributed to the fact that the footprint of the KNXnet/IP-related attacks is focused on the application layer, that CICFlowMeter has no visibility on, while there is a limited footprint on the IP and UDP layers. Given the connection-less nature of UDP, we cannot leverage additional indicators that could indicate an attack (e.g., short-lived TCP flows, frequent FIN or RST packets, etc). On the contrary, table II presents the results obtained from a classifier that was pre-trained with the KNXFlowMeter data. We can observe that the metrics were improved across all FL aggregation techniques; still the best performance is achieved by FedAvg, reaching 88,12% accuracy. The improved metrics confirm that KNXFlowMeter introduces features with more predictive power, allowing more accurate classification results.

Aggregation Strategy	ACC	TPR	FPR	F1	AUC
FedAvg	75,53%	75,53%	4,07%	72,34%	93,99%
FedProx	74,47%	74,47%	4,25%	73,67%	93,94%
FedAdam	53,43%	53,43%	7,76%	48,78%	79,05%
FedAdagrad	72,75%	72,75%	4,54%	70,16%	93,55%
FedYogi	74,07%	74,07%	4,32%	70,68%	93,08%

TABLE I
EVALUATION RESULTS FROM CICFLOWMETER

VII. CONCLUSIONS

In this work, we presented a network IDS that analyses network traffic into network flows, by leveraging application-layer statistics, and is able to detect various attacks against

Aggregation Strategy	ACC	TPR	FPR	F1	AUC
FedAvg	88,12%	88,09%	1,97%	85,76%	97,06%
FedProx	87,95%	87,89%	2,01%	87,54%	97,03%
FedAdam	71,46%	71,42%	4,75%	64,33%	92,97%
FedAdagrad	79,41%	79,36%	3,42%	75,69%	96,08%
FedYogi	78,89%	78,83%	3,51%	75,25%	96,71%

TABLE II
EVALUATION RESULTS FROM KNXFLOWMETER

KNXnet/IP. To develop the proposed IDS, first, eight different attacks against KNXnet/IP were developed and thoroughly described in this work. Useful observations were obtained from each attack, highlighting protocol-related features that could be used for observing suspicious behaviour. Next, the KNXFlowMeter was developed, which aggregates the network traffic into application-layer aware network flows, including features that could indicate suspicious activities. The evaluation results confirmed that KNXFlowMeter produced improved results compared to CICFlowMeter, by providing features with stronger predictive power. It is noteworthy that most of the attacks were detectable through statistic analysis, however, A.KNX.08 was a challenging case, since it did not produce useful statistic results. Combining both rule-based and AI-based detection techniques would be the most effective way to completely cover all attack scenarios. As next steps, we plan to employ AI explainability methods (XAI) to further understand the predictive power of the introduced features, and to further improve the feature engineering process applied during the training phase.

REFERENCES

[1] BUILD UP. "Overview article - smart buildings and smart technologies in europe: State of play and perspectives." (2023), [Online]. Available: https://build-up.ec.europa.eu/en/resources-and-tools/articles/overview-article-smart-buildings-and-smart-technologies-europe-state (visited on 03/23/2026).

[2] C. Portalés, S. Casas, and K. Kreuzer, "Challenges and trends in home automation: Addressing the interoperability problem with the open-source platform openhab," in *Research Anthology on Usage and Development of Open Source Software*. IGI Global, 2021, pp. 539–565, ISBN: 9781799891598. DOI: 10.4018/978-1-7998-9158-1.ch030. [Online]. Available: http://dx.doi.org/10.4018/978-1-7998-9158-1.ch030.

[3] M. Küppers, M. Schuba, G. Neugebauer, T. Hoener, and S. Hack, "Security analysis of the knx smart building protocol," in *Proceedings of the 18th International Conference on Availability, Reliability and Security*, ser. ARES 2023, ACM, Aug. 2023, pp. 1–7. DOI: 10.1145/3600160.3605167. [Online]. Available: http://dx.doi.org/10.1145/3600160.3605167.

[4] V. Graveto, T. Cruz, and P. Simões, "Using knx-based building automation and control systems for data exfiltration," *IEEE Internet of Things Journal*, vol. 10, no. 15, pp. 13 727–13 741, Aug. 2023, ISSN: 2372-2541. DOI: 10.1109/jiot.2023.3262873. [Online]. Available: http://dx.doi.org/10.1109/JIOT.2023.3262873.

[5] G. Johannes, M. Thomas, and S. Wiedenmann, "Risk analysis in fieldbus networks using the example of knx," in *2019 International Conference on Information Networking (ICOIN)*, IEEE, Jan. 2019, pp. 310–315. DOI: 10.1109/icoin.2019.8718149. [Online]. Available: http://dx.doi.org/10.1109/ICOIN.2019.8718149.

[6] M. Peters, J. Goltz, S. Wiedenmann, and T. Mundt, "Using machine learning to find anomalies in field bus network traffic," in *Security, Privacy, and Anonymity in Computation, Communication, and Storage*. Springer International Publishing, 2019, pp. 336–353, ISBN: 9783030249076. DOI: 10.1007/978-3-030-24907-6_26. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-24907-6_26.

[7] V. Graveto, T. Cruz, and P. Simões, "A network intrusion detection system for building automation and control systems," *IEEE Access*, vol. 11, pp. 7968–7983, 2023, ISSN: 2169-3536. DOI: 10.1109/access.2023.3238874. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2023.3238874.

[8] M. Cash, C. Morales-Gonzalez, S. Wang, *et al.*, "On false data injection attack against building automation systems," in *2023 International Conference on Computing, Networking and Communications (ICNC)*, IEEE, Feb. 2023, pp. 35–41. DOI: 10.1109/icnc57223.2023.10074353. [Online]. Available: http://dx.doi.org/10.1109/ICNC57223.2023.10074353.

[9] C. Dalamagkas, *Knxsmartfuzzer: Collection of knxnet/ip attacks*, https://gitlab.ithaca.ece.uowm.gr/ai4cyber/KNXSmartFuzzer, 2024.

[10] C. Vacherot, "Sneak into buildings with KNXnet/IP," in *Sneak into buildings with KNXnet/IP*, Présenté aux conférences :- GreHack 2020- Defcon 2021, Lyon, France, Nov. 2020. [Online]. Available: https://hal.science/hal-03022310.

[11] C. Dalamagkas, P. Radoglou-Grammatikis, P. Bouzinis, *et al.*, "Federated detection of open charge point protocol 1.6 cyberattacks," *Complex Engineering Systems*, vol. 5, no. 2, 2025, ISSN: 2770-6249. DOI: 10.20517/ces.2025.04. [Online]. Available: https://www.oaepublish.com/articles/ces.2025.04.

[12] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an intrusion detection dataset: The cicids2017 case study," in *2021 IEEE Security and Privacy Workshops (SPW)*, IEEE, May 2021, pp. 7–12. DOI: 10.1109/spw53761.2021.00009. [Online]. Available: http://dx.doi.org/10.1109/SPW53761.2021.00009.

[13] C. Dalamagkas, P. Radoglou-Grammatikis, P. Bouzinis, T. Lagkas, V. Argyriou, and P. Sarigiannidis, *Knxnet/ip intrusion detection dataset*, version 1.0.0, Zenodo, Aug. 2025. DOI: 10.5281/zenodo.16957517. [Online]. Available: https://doi.org/10.5281/zenodo.16957517.

[14] Y. Zhou, Q. Ye, and J. Lv, "Communication-efficient federated learning with compensated overlap-fedavg," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 1, pp. 192–205, Jan. 2022, ISSN: 2161-9883. DOI: 10.1109/tpds.2021.3090331. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2021.3090331.

[15] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, *Federated optimization in heterogeneous networks*, 2018. DOI: 10.48550/ARXIV.1812.06127. [Online]. Available: https://arxiv.org/abs/1812.06127.

[16] S. Reddi, Z. Charles, M. Zaheer, *et al.*, *Adaptive federated optimization*, 2020. DOI: 10.48550/ARXIV.2003.00295. [Online]. Available: https://arxiv.org/abs/2003.00295.

[17] S. Reddi, Z. Charles, M. Zaheer, *et al.*, *Adaptive federated optimization*, 2020. DOI: 10.48550/ARXIV.2003.00295. [Online]. Available: https://arxiv.org/abs/2003.00295.