2024 IEEE 29th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)

2024 IEEE 29th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD) | 979-8-3503-7764-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/CAMAD62243.2024.10942818

A Generalized Multi-Layer IDS for Smart Buildings

Marco Ruta*, Pietro Giuseppe Giardina*, Giada Landi*, Rosario G. Garroppo†
* Nextworks s.r.l., Pisa, Italy
†University of Pisa, Dip. di Ingegneria dell'Informazione
Email: *m.ruta@nextworks.it, p.giardina@nextworks.it, g.landi@nextworks.it † rosario.garroppo@unipi.it

***Abstract*—Internet of Things (IoT) technology represents an important opportunity for future smart buildings. However, IoT systems require robust security measures to protect against threats that could compromise system functionalities, data privacy, and even the safety of inhabitants. To this end, this paper proposes a multi-layer intrusion detection system (ML-IDS) that operates at both the network and sensor layers making use of a unified data engineering strategy for network and sensor data. The proposed solution is based on a general-purpose system that can accommodate new types of data available at network layer or other information generated by new IoT devices added to the system. To reduce the amount of data, the method introduces a data compression strategy based on sampling, aggregation, and image conversion, which allows the use of Convolutional Neural Networks (CNN). Performance analysis demonstrates the system effectiveness in identifying attacks at the network layer, achieving a False Negative Rate (FNR) of 0.16% and a False Positive Rate (FPR) of 4.76%, and at the IoT layer, with strong performance on four out of six types of sensors. Additionally, the data compression approach efficiently manages to convert data of different nature while reducing the validation dataset size from 20GB to 2MB.**

I. Introduction

Smart buildings are bringing about a huge advancement in making environments more efficient, sustainable, and user-friendly. These evolving ecosystems integrate various devices, from lighting to heating, ventilation and air conditioning (HVAC) systems, in ways that enhance both comfort and efficiency. This integration indicates that in the future, buildings will not only adapt to our needs but also anticipate them [1][2]. However, the huge amount of technology employed to enable smart buildings introduces complex security vulnerabilities, thereby increasing risks to privacy, security, and safety. The integration, as well as increased dependence on interconnected systems, raises the possibility of cyber-physical attacks, which can disrupt essential services and, in extreme cases, endanger lives [3][4]. Smart buildings collect huge quantities of data which can disclose many details about the behavior and routines of the people living or working in them. The sensitivity of this information has been underscored in a prior study [3]. For these reasons, security measures that stop intruders from accessing the system without permission need to be deployed. Also, any such measure should abide by rules that look after individuals' personal information, e.g. General Data Protection Regulation (GDPR) among others [5].

II. Related Work

In an IoT-enabled context such as smart buildings, Machine Learning-based intrusion detection systems (IDS) are a widely adopted solution due to the complex and dynamic nature of the setting that generates huge amounts of data [6][7][8][9]. A general framework for Machine Learning (ML) applied to IDS is presented in [10] consists in four steps: *data collection*, *data preprocessing*, *learning* and *interpretation*. However, analyzed solutions do not focus on attack detection in smart environments at multiple layers of the architecture: [11] focuses on a novel technique for network data feature extraction and optimization to enhance detection capabilities, [12] explores the use of specific loss functions to optimize IDS performance for underrepresented classes, and [13] employs Principal Component Analysis (PCA) to efficiently manage large data volumes making the IDS more efficient. While these solutions are valuable, they focus exclusively on network data and often overlook attacks that do not use the newtork as attack vector (e.g., jamming of a sensors). Moreover, the data reduction and optimization techniques proposed in these works are tailored to network data and are not general purpose techniques that can be applied to other data types.

A. Paper contribution

This paper proposes a novel ML method for smart building IDS that operates at both the network and IoT layers to identify structured attacks at network layer and anomalies on IoT sensors. The method is based on a general-purpose data preprocessing and engineering system that uses the same methodology for both network and IoT data. This characteristic allows the entire system to adapt to new network features and new sensor

979-8-3503-7764-4/24/$31.00 ©2024 IEEE

data. The proposed ML-IDS is further characterized by compressing and transforming the resulting data in images to exploit the great advances of CNN in image analysis for extracting useful information.

III. The Dataset

This section presents a brief review of some of the most recent IoT security datasets available in the literature to identify the most suitable for developing and testing the smart building ML-IDS method. While [14] provides an extensive overview of available IoT network security datasets, this review focuses on a subset composed of three datasets:

1) ***CIC-IoT****:* the CIC-IoT dataset [15] includes approximately 105 IoT devices that participate either as attack vectors or targets. The dataset groups attacks into seven major types, providing a rich testing ground for IDS.

2) ***Aposemat IoT Dataset****:* the Aposemat IoT dataset [16] focuses on malware within IoT networks, including traffic from actual IoT devices such as Amazon Echo and Philips Hue.

3) ***ToN-IoT****:* the ToN-IoT dataset [17] provides data from three different sources: telemetry from IoT devices, operational system logs, and network traffic. The testbed simulates a real-world smart city network including a virtualized architecture of cloud services, a fog layer, and an edge layer in which different IoT devices are simulated.

Feature	**CIC-IoT**	**ToN-IoT**	**Aposemat**
IoT Devices	$\sim$ 100	$\sim$ 10	$\sim$ 10
Attacks	generic[1]	generic[2]	malware[3]
Data Format	.pcap, .csv	.pcap, .csv	.pcap
Available Data	nw[4]	nw[4], os[5], t[6]	nw[4]

TABLE I: Comparison of IoT Security Datasets.

[1] Brute Force, DoS, DDoS, Flooding, Injection, Mirai, Scanning, Spoofing.
[2] Backdoor, DoS, DDoS, Injection, MITM, Scanning, Ransomware, Password Guessing, XSS.
[3] Mirai, Torii, Trojan, Gagfyt, Kenjiro, Okiru, Hakai, IRCBot, Hajime, Muhstik, Hide & Seek.
[4] Network data.
[5] Operative Systems logs.
[6] Telemetry data from sensors.

The datasets are evaluated based on their ability to represent characteristics relevant to the smart building scenario, including multiple IoT devices and network traffic patterns that reflect real-world smart building environments. Table I provides a summary of the characteristics considered in the dataset selection process. While each dataset has distinct strengths, the ToN-IoT dataset was selected because of its realistic smart city testbed, that provides a robust environment for simulating smart building networks. Furthermore, this dataset is the only one that provides both network and sensor telemetry data, allowing for the development of a multi-layer IDS. However, the use of simulated IoT devices (Thermostat, Fridge, GPS, Garage Door, Motion Light, and Weather station) represents a limitation: the dataset provides only device captures, not including information about the sensors' state (e.g. power consumption, connectivity state, battery state, etc..), that could be useful to add another layer of detection to the proposed method. The attacks present in this dataset are carried out using complete systems (Kali machines) and, for each type of attack, the tools and the steps are meticulously reported, making both the testbed and the attacks fully replicable. However, two over nine of the available attacks were removed from the analysis after an examination of the dataset, which revealed significant differences in the number of packets across the various attack categories. In fact, in Table II, it can be observed that the instances of man-in-the-middle (MITM) and Ransomware are significantly lower than those of other attacks, making the dataset extremely unbalanced.

Traffic type	**Total data records**
Backdoor	508,116
DDoS	6,165,008
DoS	3,375,328
Injection	452,659
MITM	1052
Password	1,718,569
Ransomware	72,805
Scanning	7,140,161
XSS	2,108,944
Normal	796,380

TABLE II: Network attacks distribution

IV. Data preprocessing and engineering unit

This section details the design and development of the general-purpose data preprocessing and engineering system designed to transform raw data into a format compatible with ML, thus making it ready for training and inference. Throughout the design and development phases, the following principle was followed: all decisions regarding the selection and design of tools and frameworks were made with the specific goal of building a general-purpose data engineering chain capable of processing any kind of raw network and sensor data. Therefore, this system can theoretically convert any .pcap file generated by network traffic measurement tools and .csv data generated by sensors into a ML-compatible format. Thus, the system is not strictly tailored to the ToN-IoT dataset. The network and sensor data are processed using the same pipeline, which consists of *encoding*, *sampling*, and *image conversion*. The architecture of the Data Processing and Engineering Unit, which is described later

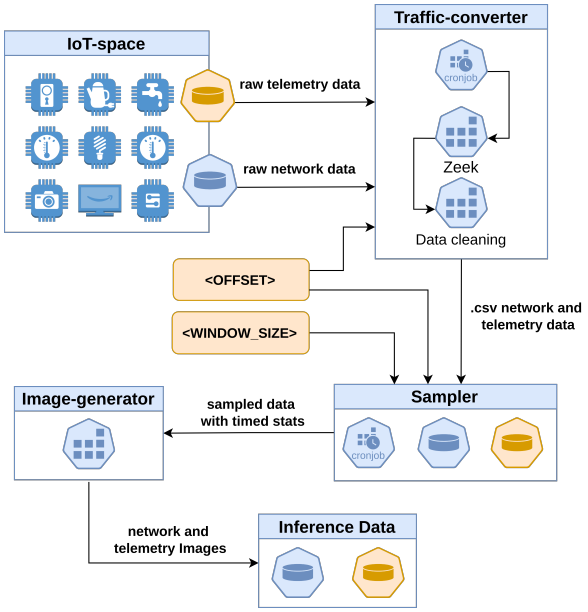


Fig. 1: Data preprocessing and engineering unit

in this section, is shown in Figure 1. Note that the sensor data are in .csv format and are ready to be pre-processed, while the network data requires an extra step: extracting network features from the raw .pcap format. The Zeek (ex Bro) [18] tool was used to extract the relevant network features for this purpose, first extracting nine different log files related to different protocols (conn.log, dns.log, files.log, http.log, ntp.log, packet_filter.log, ssl.log, weird.log, X509.log) from the .pcap. Then, these .log files are merged into a single .csv file containing all the information from the different protocols for each packet. This approach, similar to the one described in [17], was chosen because it can be automated and its execution does not require direct human intervention. On the contrary, alternative approaches, such as the one proposed in [15] and based on the use of the DPKT [19] tool, may require human intervention.

A. Data preprocessing

The data preprocessing step is minimal, primarily focused on data formatting and the dropping of unformattable instances. This process is carried out ensuring that the preprocessing stage remains flexible enough to accommodate diverse datasets without requiring human intervention. The preprocessing module takes as input the .csv files of the different sensors and the .csv file of the network features extracted by Zeek. The operations performed by the module are:

1) ***Sensor data****:* these data are numerical (e.g., the temperature of the fridge) or boolean (e.g., the state of the garage door). The only data preprocessing step is spotting and formatting malformed data. For boolean features, malformed values such as “False”, “false”, “f”, “True”, “true”, “t”, are converted into a common binary representation (0,1). For numerical features, malformed values (e.g., strings) are dropped.

2) ***Network data****:* network data are both categorical (e.g., HTTP methods) and numerical (e.g., connection duration). Categorical features are converted into integer features using a custom *Label Encoder*, which iterates and collects all the possible values for a particular feature to create a `string:int` mapping, used for conversion. The different mappings are kept in a file that can be accessed to load the Label Encoders and use them in the future to convert other data, while keeping the same mapping. With this type of data, there are two issues to deal with for numeric features: malformed values and large values. In the first case, as with sensor data, malformed values are discarded. In the second case, wide-range values (e.g., duration, number of bytes or packets) are converted to logarithmic scale to be efficiently represented in a narrower range of values. A common preprocessing phase is the conversion of the `date:time` pair in the sensor data and the timestamp (UNIX) of the network data into a common timestamp `YYYY/MM/DD hh:mm:ss` format. After these steps, all the data are scaled with a *Robust Scaler*, which normalizes the data by removing the median and scaling according to the quantile range, preserving the outliers. The ML-IDS uses binary labels for the sensors. This approach allows the IDS to act as an *anomaly detection system* at the sensor layer, focusing on identifying deviations from normal behavior within the sensor data streams. In contrast, the network-based component of the ML-IDS has a broader scope, covering different types of network cyber-attacks.

B. Data sampling

The purpose of the data sampling step is to reduce the training and inference overhead by reducing the amount of data. This approach differs from standard evaluations made on the ToN-IoT dataset, where each packet, or a series of packets, represents an instance for training or prediction. Also, sampling data within a time window allows the extraction of new descriptive metrics for each feature in the selected time window. Sampling and aggregation are carried out using the python `pandas` tool `resample and aggregate` within a 10-second rolling window with a 5-second offset. The feature values of each time window are aggregated using custom aggregation methods for each data type, as follows:

- *Numeric* features are represented by three new metrics extracted from the feature values in the time window: minimum, maximum, and average.
- *(ex) Categorical* features are represented by three new metrics extracted from the feature values

Source	Num. original samples	Num. of original features	Num. samples after s.	Num. of features after s.
Network	24,000,000	42	45,000	126
Fridge	39,944	2	5,371	6
Weather	39,260	3	4,937	9
Thermos.	32,774	2	4,399	6
GPS	38,960	2	6,193	6
Motion	39,488	2	6,851	6
Garage	39,587	2	5,177	6

TABLE III: Effect of the data sampling on time space and features space - Considered dataset.

in the time window: first, second, and last most frequent value.

- *Binary label* are aggregated using `any` logic. Therefore, if even one capture is marked as malicious, the entire window of data is marked as malicious.
- *Multi-class labels* are aggregated using the `most frequent` logic. Therefore, the window label is assigned as the most frequent value. However, if the most frequent value is "normal" but there is still at least one attack, the second most frequent value is assigned.

This step transforms each original feature into three new features. In summary, the method reduces the data in the time space, but expands it in the feature space. Referring to the considered data set, Table III shows how this procedure reduces the number of samples while increase the number of features.

C. Image conversion

After the data sampling process, each feature within the time window is now described by three metrics: the minimum, the maximum, and the average values for numerical features and the first, the second, and the last most frequent values for categorical features. This representation provides a comprehensive overview of the value distribution within the time window and can potentially be applied to any new type of network observations and sensor data. To further reduce the amount of data to be processed, a 3:1 display ratio is used. In particular, the feature metrics are assigned to the RGB intensity of pixels in an image. There are two main reasons for this choice: representing each of the three features as an RGB pixel reduces the amount of data by one-third, minimizes the inference overhead, and allows the use of consolidated deep neural network (DNN) technologies such as CNNs, which have been widely used and proven in various image-related tasks [20]. To perform this task, data are re-shaped into 3-dimensional matrices, as represented in Figure 2, to be later converted to images. The sensor data is transformed into a 6×3×3 matrix. Each row of the matrix represents a sensor, and each cell contains a

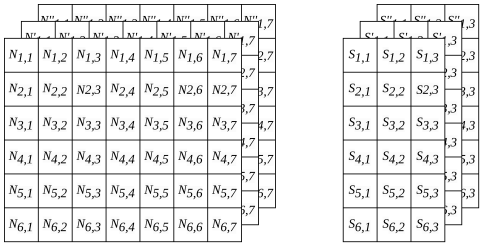


Fig. 2: Network and Sensors features matrices

Fig. 3: Network images with separated (left) and unified (right) channels

metric describing a feature. If a sensor has only two measures, `NaN` values are used to fill the last cell of the row. For example, the weather sensor has three columns filled by the values observed by its features (see Table III), i.e. temperature, pressure, and humidity. On the contrary, GPS has only longitude and latitude; hence the third column is filled by `NaN`. Network data is transformed into a 6×7×3 matrix, where each cell contains a metric describing a feature. As shown in Table III the selected network features are 42 and this number has been tripled by the sampling and expansion procedure previously presented. `NaN` values are used to fill the remaining positions in the matrix. In both matrices, values are converted in the range `0:255` to be represented as pixel RGB intensity and NaN values are converted to zeros. The matrices are now converted into images using the Python library `PIL` by mapping each sub-value of the matrix to the pixel RGB intensity. Figure 3 shows the network matrix transformed into a 2D matrix with (top) separated channels (R, G, B) and (bottom) unified color channel (RGB). Figure 4 shows the same for the sensor data.

D. Remarks

The developed system, shown in Figure 1, efficiently converts raw data into an ML-friendly format while respecting the key design principle: the data sampling and the methodology used to extract relevant metrics in the sampled space, such as maximum, minimum, average, and most frequent values, are flexible. The presented approach is not tailored to the considered dataset

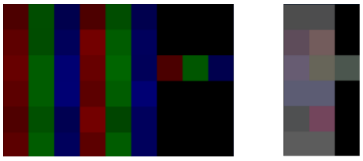


Fig. 4: Telemetry images with separated (left) and unified (right) channels

and lays the foundation for the integration of additional sensors and network features. The design principles of the system ensure that it can easily adapt to other data sources, making it a general-purpose solution for a wide range of IoT and network data engineering challenges. Moreover, the used data sampling, aggregation, and image conversion techniques, allowed the reduction of the ToN-IoT dataset from 20GB of raw data to 2 MB of images.

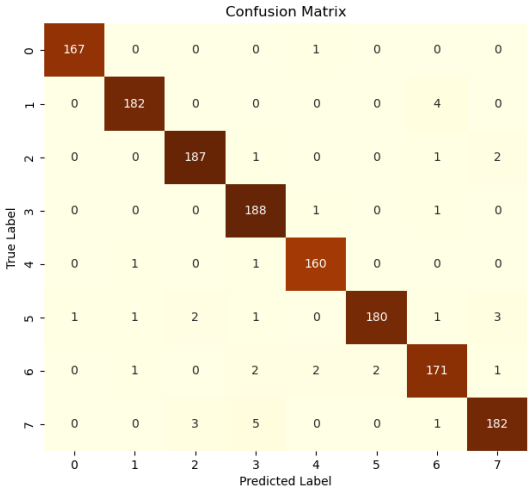
V. ML Models

As described in Section IV, the data preprocessing and engineering system generates images representing the sensors and network state in a time window. This motivation led to the development of two distinct CNNs networks composed of standard CNN layers as *Input*, *Conv2D*, *MaxPooling2D*, *Flatten*, *Dense*, *Dropout*, and *Output*. While the network and sensor architectures adhere to standard CNN layerization, there are some differences between the two:

- **Input layers:** the input layers shapes correspond to the input images shapes of 6×7 pixels for the network and 6×3 pixels for the sensors. The input shapes are three-dimensional due to the RGB format of both images.
- **Output layers:** the output layers shapes are designed to address the different CNNs tasks. The output layer of the sensors CNN has a shape of 6×3, as it provides a prediction for each of the 6 sensors. The prediction can have a value of 0 (normal traffic), 1 (attack), or 2 (sensor not available). The output layer of the network data has a shape of 1×8, as it predicts the presence or not of each of the seven different attacks considered in the analysis. The attacks are those reported in Table II, with the exception of MITM and Ransomware.
- **Reshape layer:** an additional layer is required in the sensor CNN to convert the output shape from 1×18 to a 6×3 matrix where the activation function is applied row by row, allowing for a separate prediction for each of the 6 sensors.

VI. Validation

The data processing system and the CNNs are tested by splitting network and sensor data into training, test, and validation sets using a 70-20-10 stratified split, preserving the label distribution. After training the network and sensor CNNs until the validation loss stop to decrease (200 and 100 epochs, respectively), confusion matrices were computed over the validation set and *accuracy*, *recall*, *precision*, *F1*, *false positive rate*, and *false negative rate* metrics are calculated. To do this, the value "5" in the network's confusion matrix is considered negative, representing the "normal" class, while all other classes representing attacks are considered positive. In the sensor's confusion matrix, class "1" is considered positive and "0" is considered negative, while class "2", which represents a sensor that is off, has been removed. The network's confusion matrix is shown in Figure 5 while the the network's training curve is shown in Figure 6. From the training curve, it can be observed that the model does not exhibit overfitting to the training data until 200 epochs, at which point the validation accuracy becomes constant and the gap between the loss and validation loss begins to increase. For brevity, the confusion matrices and the training curve of the sensors are not shown. The metrics extracted from the matrices are shown in Table IV. Across all metrics, the performance of the CNN network is excellent, with an Accuracy of 99.23%, meaning that it can predict correctly most of the time. When it comes to Precision, this model scores 99.28% while the Recall level is at 99.84%. Therefore, it achieves both high levels in detecting positive samples (FNR 0.16%) at the expense of increasing the FPR (4.76%), meaning that there would be more cases with false alarms than missing some attacks. However, individual sensor CNNs perform differently from others: GPS, Motion Light, and Garage Door reach perfect scores on all metrics; others, such as Fridge and Weather, have high FPR and perfect Recall, while the Thermostat, with with 16.83% FPR has suboptimal Recall score.



0:backdoor, 1:dos, 2:xss, 3:password, 4:ddos, 5:normal, 6:scanning, 7:injection

Fig. 5: Network Confusion Matrix of ML-IDS

VII. Conclusion

Non-tailored data preprocessing and engineering combined with CNNs have shown promising performance, especially on network data and certain sensor data. However, the sensors CNN showed some performance

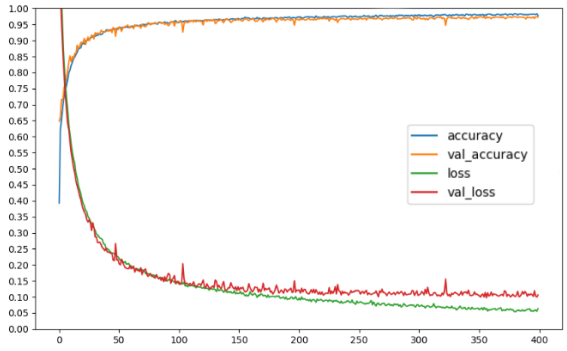


Fig. 6: Training and validation accuracy and loss over Epochs

CNN	Acc.	Prec.	Rec.	F1	FPR	FNR
Netw.	99.23	99.28	99.84	99.56	4.76	0.16
GPS	100	100	100	100	0	0
Fridge	93.25	90.95	100	95.26	20.94	0
Garage	100	100	100	100	0	0
Weather	93.98	93.60	100	96.69	50.0	0
Thermos.	91.23	100	83.17	90.81	0	16.83
Motion	100	100	100	100	0	0

TABLE IV: CNNs Performance Metrics (%)

inconsistencies, indicating that work on data preprocessing and model optimization needs to be improved. One of the main strengths of such an approach is that it can transform different forms of information into images to extract relevant temporal features from the sampling interval, without the need for customized extraction processes. This integrated feature extraction approach is highly flexible, ensuring smooth adaptability to different new data collections. It is worth noting that the sensor data used in this study was fictitious and the next step necessitate the use of reals sensors data, which includes not only captures, but also sensor state details, for a more comprehensive analysis that can help resolve the observed discrepancies. Furthermore, the significance of these results lies not only in the successful application across different data types, but also in the fact that basic CNNs were used. This underscores the potential for significant improvements with more advanced DNN models.

ACKNOWLEDGMENT

This work has been funded by the European Commission through the Horizon Europe JU SNS project ROBUST-6G (Grant Agreement no. 101139068) and supported by TUBITAK through the 1515 Program under Project 5169902.

REFERENCES

[1] A. Latifah, S. Supangkat, and A. Ramelan, "Smart building: A literature review," *2020 International Conference on ICT for Smart Society (ICISS)*, vol. CFP2013V-ART, pp. 1–6, 2020.

[2] A. K. Kumar, S. Sharma, N. Goyal, A. Singh, X. Cheng, and P. Singh, "Secure and energy-efficient smart building architecture with emerging technology iot," *Comput. Commun.*, vol. 176, pp. 207–217, 2021.

[3] I. M. Runge, B. Akinci, and M. Bergés, "Challenges in cyber-physical attack detection for building automation systems," in *Proceedings of the 10th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, ser. BuildSys '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 236–239. [Online]. Available: https://doi.org/10.1145/3600100.3623738

[4] D. Meyer, J. Haase, M. Eckert, and B. Klauer, "A threat-model for building and home automation," in *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, 2016, pp. 860–866.

[5] European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council. [Online]. Available: https://data.europa.eu/eli/reg/2016/679/oj

[6] D. Musleh, M. Alotaibi, F. Alhaidari, A. Rahman, and R. Mohammad, "Intrusion detection system using feature extraction with machine learning algorithms in iot," *J. Sens. Actuator Networks*, vol. 12, p. 29, 2023.

[7] S. R. Manzano, N. Goel, M. Zaman, R. Joshi, and S. Naik, "Design of a machine learning based intrusion detection framework and methodology for iot networks," *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0191–0198, 2022.

[8] M. Alani, "Iotprotect: A machine-learning based iot intrusion detection system," *2022 6th International Conference on Cryptography, Security and Privacy (CSP)*, pp. 61–65, 2022.

[9] B. Xu, L. Sun, X. Mao, R. Ding, and C. Liu, "Iot intrusion detection system based on machine learning," *Electronics*, 2023.

[10] D. Djenouri, R. Laidi, Y. Djenouri, and I. Balasingham, "Machine learning for smart building applications," *ACM Computing Surveys (CSUR)*, vol. 52, pp. 1–36, 2019.

[11] A. A. Malibari, S. S. Alotaibi, R. Alshahrani, S. Dhahbi, R. Alabdan, F. N. Al-wesabi, and A. M. Hilal, "A novel metaheuristics with deep learning enabled intrusion detection system for secured smart environment," *Sustainable Energy Technologies and Assessments*, vol. 52, p. 102312, 2022.

[12] A. S. Dina, A. Siddique, and D. Manivannan, "A deep learning approach for intrusion detection in internet of things using focal loss function," *Internet of Things*, vol. 22, p. 100699, 2023.

[13] Y. Kayode Saheed, A. Idris Abiodun, S. Misra, M. Kristiansen Holone, and R. Colomo-Palacios, "A machine learning-based intrusion detection for detecting internet of things network attacks," *Alexandria Engineering Journal*, vol. 61, no. 12, pp. 9395–9409, 2022.

[14] F. De Keersmaeker, Y. Cao, G. K. Ndonda, and R. Sadre, "A survey of public iot datasets for network security research," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 3, pp. 1808–1840, 2023.

[15] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment," *Sensors*, vol. 23, no. 13, p. 5941, 2023.

[16] S. Garcia, A. Parmisano, and M. J. Erquiaga, "Iot-23: A labeled dataset with malicious and benign iot network traffic," 2020.

[17] N. Moustafa, "A new distributed architecture for evaluating ai-based security systems at the edge: Network ton-iot datasets," *Sustainable Cities and Society*, vol. 72, p. 102994, 2021.

[18] V. Paxson, "Bro: a System for Detecting Network Intruders in Real-Time," *Computer Networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999. [Online]. Available: http://www.icir.org/vern/papers/bro-CN99.pdf

[19] D. Kiefer and A. Graupner, *dpkt Documentation*, dpkt Project, 2024. [Online]. Available: https://dpkt.readthedocs.io/en/latest/

[20] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision," *Artificial Intelligence Review*, vol. 57, no. 4, p. 99, Mar 2024.