IoT Threat Detection Using Process Conformance and ML Classification

Adam Burke[1,*], Mashal Ashraf[1], Marimbay Kadirov[1] and Andrzej Janusz[1,2,*]

[1]School of Information Systems, Queensland University of Technology, Brisbane, Australia
[2]Institute of Informatics, The University of Warsaw, Warsaw, Poland

Abstract

The Internet-of-Things (IoT) cybersecurity landscape is ever-evolving and requires continuous efforts to deal with new threats. We explore using process mining to extract features from IoT network traffic data for use with a machine learning classifier to detect cybersecurity anomalies. Our data pipeline calculates process models and process conformance metrics that, in turn, are used to train machine learning models crafted for network data evaluation. This paves the way for automatic threat detection systems in the area of IoT device security. We assess our results experimentally, paying attention to data selection and information leakage in model training and experimental design. The resulting model is successful in differentiating between malicious and benign traces, with process conformance metric features being of particular importance.

Keywords

Internet of Things, Process Mining, Machine Learning, Intrusion Detection, Anomaly Detection, IoT Security, Cybersecurity

1. Introduction

Forty billion Internet of Things (IoT) devices are expected to be in use by 2034 [1]. IoT devices are being adopted in industries such as agriculture, healthcare, logistics, home automation, and environmental monitoring [2]. Due to the interconnectedness of IoT devices, the huge amount of information they deal with, and their limited computing power, they are prone to being targets of cyber attacks [2, 3]. We can see IoT security incidents ranging from hijacking and denial-of-service to ransomware attacks [4, 5]. There is great value in detecting such attacks quickly and accurately.

This paper uses process mining and machine learning for network anomaly detection. Process mining is a family of techniques for automatically discovering, monitoring and improving real processes based on the information extracted from organisational data in the form of event logs, which collect sequential events [6, p31]. Process mining has great potential for insights into cybersecurity because many security-relevant phenomena are inherently sequential [7]. Process conformance metrics can provide quantitative information on the similarity between a particular sample of network activity and normal behaviour.

To investigate process mining and machine learning techniques for intrusion detection, we explore:

i) *Can process conformance metrics capture malign behavioural deviations in network traffic?*
ii) *Do such features improve classification performance compared to directly observed network features?*

Our solution is a *process-informed* machine learning pipeline that transforms IoT network traffic into event logs, discovers a reference process model from benign behaviour, computes conformance metrics for each training trace, and uses these metrics as input features for training a machine learning model. Then, at runtime, inference is performed by calculating metrics for individual network traces and performing classification with the ML model. We evaluate the approach on a public dataset of

Pre-Print - accepted at - ASPAI 2026, 2nd Asia-Pacific Symposium on Process and Artificial Intelligence
at.burke@qut.edu.au (A. Burke); mashal.ashraf@connect.qut.edu.au (M. Ashraf); marimbay.kadirov@connect.qut.edu.au (M. Kadirov); andrzej.janusz@qut.edu.au (A. Janusz)
0000-0003-4407-2199 (A. Burke); 0009-0002-7175-7783 (M. Ashraf); 0009-0003-7175-0837 (M. Kadirov); 0000-0002-9763-1399 (A. Janusz)

cybersecurity attacks [8] and carefully examine experimental design choices, particularly data-splitting strategies that may introduce information leakage.

The remainder of the paper is structured as follows. Section 2 discusses existing literature and key concepts of the Process Mining and Machine Learning fields. Section 3 explains our data pipeline. Section 4 describes experiments and results. Section 5 concludes the paper.

2. Related Work

This section discusses related work in machine learning and process mining as applied to the cybersecurity domain. It also critically reviews existing machine learning solutions for attack detection.

2.1. Machine Learning for Cybersecurity

There is much existing work on machine learning techniques in IoT security. According to [9], machine learning can be used for intrusion detection, anomaly detection, threat intelligence and prediction, firmware and software vulnerability analysis, data privacy and encryption, and behaviour-based authentication in IoT system security. Similarly, [10] notes the use of machine learning models in authentication, access control, malware detection, and secure offloading services. These reviews note limitations that come with using ML for IoT security, such as the large overhead produced by the models [10], the limited processing power of IoT devices [9], the model's inability to handle high-velocity streams of data [9], and the risk of models learning incorrect behaviours and policies [10].

Existing ML approaches often do not model the sequential structure of this incoming traffic, whereas the process mining techniques in this paper explicitly model it. In [11], the authors analysed traces of system processes on an IoT device and used their embeddings to extend the set of features extracted from system audit logs. They demonstrated that such features can significantly improve the performance of ML models for the cyberattack detection task. That approach relied on ML techniques to capture the sequential nature of system events in multidimensional vectors. The current paper uses explicit process models based on well-known process mining formalisms.

2.2. Process Mining for Cybersecurity

Process mining techniques use sequential event data to check compliance, analyse bottlenecks, compare process variants, and suggest improvements to existing processes. Event data with timestamps, activity labels, and case identifiers are organised into an input data structure called an *event log*. Standard techniques discover control-flow structures from these event logs including choice, loops and concurrency [6, 12], and output the result as a *process model*. Process mining discovery is a way of identifying patterns in sequential events of modal data. In cybersecurity network interactions, authentication attempts, and protocol exchanges all occur as ordered sequences of events. Conformance checking is a branch of process mining that is used to identify discrepancies between process models and observed behaviours [6, p243]. Process conformance metrics provide quantitative measures of how well an observed event log aligns with a reference process model.

Petri nets [6] are one important notation for representing processes in a concise way with simple semantics. In a Petri net, each activity is modelled by a rectangle-shaped transition, and each transition is connected via a circular place, which represents a possible state of the process. A transition can be enabled when all input places hold a token, after which it will fire an output token for each output place. An example Petri net is in Figure 2. In our research, we used a stochastic Petri net extension for our reference model to represent network behaviour, as explained in Section 3.

Process mining has been previously used in a cybersecurity context and yielded positive results. Different process mining techniques have been used to investigate simulated Man-in-the-Middle (MITM) attacks [13], with results showing detection of ARP spoofing attacks. An attack triage system was introduced in [14] that used process mining to model botnet command log behaviour against IoT devices, showing that process mining can be an effective tool for the automated discovery of new IoT attacks.

Process mining has also been used for anomaly detection in intelligent automation systems [15], insider threat detection using audit logs and conformance checking [16], and process-aware intrusion detection systems for industrial internet-of-things (IIoT) devices [17].

Zhong et al [12] use process mining techniques to create process models from event logs, and then identify anomalies based on conformance checking results [12]. In their approach, the defined a complete flow as one having an ACK flag at the beginning and an RST or FIN flag at the end, all other cases were discarded. In the paper, they use both inductive mining [18] and fuzzy mining [19] techniques and compare the results. The paper argues that process mining is ineffective for network-based anomaly detection as models mined from real-world, complex network traffic data will be too general, and that all flows, whether benign or malicious, being routed on a network will comply with the TCP protocol, hence making anomaly detection based on TCP data ineffective. They conclude that the frequency distribution of transitions is more effective in anomaly detection than the change in transitions themselves. Later work introduces an event pre-processing algorithm for online intrusion detection [20]. This algorithm processes network packet data before it is fed into a machine learning classifier. Both papers [12, 20] claim that inductive mining cannot be used for intrusion detection, based on their investigation of process mining models used as a standalone anomaly detector. By contrast, Section 4 shows successful results when inductive miner models are used as a feature extraction mechanism within a machine learning pipeline.

2.3. IoT-23 Dataset and Machine Learning

The IoT-23 dataset [8] is a large public dataset with rich examples of IoT network traffic under normal and attack conditions. It records the traffic when a Raspberry Pi device was used to execute malware on IoT gadgets such as a Philips HUE smart LED lamp, an Amazon Echo home intelligent personal assistant, and a Somfy smart door lock.

In computer networks, application data is segmented into packets at the transport layer, with each packet receiving an identifier containing the source IP address, source port number, destination IP address, and destination port number [21]. All packets sharing the same identifier form a session, which is a time-ordered exchange between two endpoints. Network monitoring tools such as Zeek process raw packet capture (pcap) files and aggregate each session into a single connection record stored in `conn.log` files [8]. In the IoT-23 dataset [8], each of its 23 network capture files corresponds to a distinct IoT network traffic capturing scenario. Rows in these files correspond to connections that were labelled by the dataset creators. Connections within a single scenario were captured for a single IoT device. In 20 scenarios, that device was deliberately infected with malware, and prior knowledge about the malware behaviour pattern was used in the labelling process. This means that most connections in each `conn.log` file are correlated and represent communication between the same devices sharing a common network context.

Two relevant studies [22, 23] evaluated ML-based intrusion detection specifically on the IoT-23 dataset. These report accuracy, precision, recall and F1 scores above 99.9%. We argue that these results do not reflect realistic production outcomes for those models. Section 4.2 presents this analysis, followed by reproductions of the experiments, and benchmarking against the new technique in this paper.

3. Process-Informed Machine Learning Intrusion Detection

Our approach follows the established machine learning structure of a data pipeline for constructing models, followed by the use of those artifacts for runtime inference. This section explains that pipeline, with more specific details on the use of process conformance features.

3.1. Intrusion Detection Pipeline

Figure 1 gives an overview of the model construction pipeline and its potential operational use. During model construction, recorded network traffic in the IoT-23 dataset is pre-processed and prepared as a

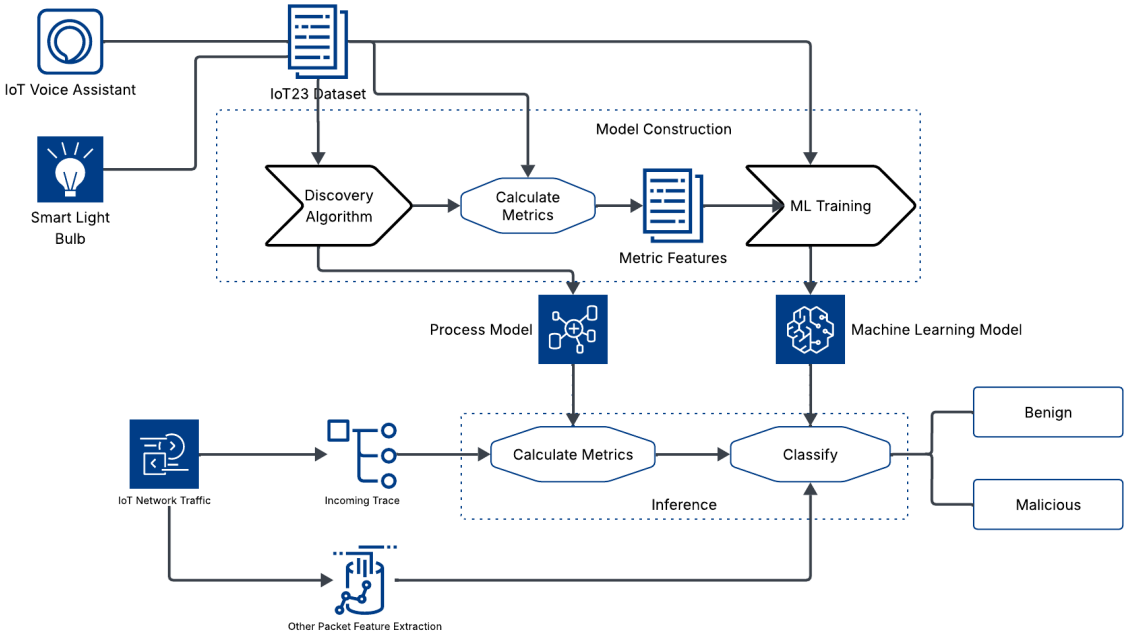

Figure 1: Overview of the proposed IoT cyber-threat monitoring pipeline, and operational use.

process event log. This is then input to a process discovery algorithm, which produces a process model. For each case in the event log, process conformance metrics are calculated using the process model and case as inputs. The machine learning model is then trained based on the directly observed features in the IoT-23 dataset and the additional process mining features. At runtime, intrusion detection inference consists of two steps. First, an incoming network trace has process conformance metrics calculated using the reference process model. Second, the machine learning model is used to classify the trace as benign or malicious.

While this general architecture provides a number of general slots where alternatives may be substituted, the specific algorithm choices used in this paper are also relevant. The process model was built using only benign network traces, simplifying training data preparation. The model then represents normal network behaviour without anomalies. As a case concept, we chose TCP connections, which are easily identifiable in network data, and as an activity notion, we used TCP packet types. The process discovery algorithm for producing the process reference model consisted of two stages. First the Inductive Miner [18] was used to discover a Petri net, followed by the alignment weight estimator [24] to add weights, making a Stochastic Labelled Petri Net (SLPN). Techniques from stochastic process mining provide additional information on probability for learning network behaviour. This then allowed the use of stochastic process conformance metrics to capture different aspects of behavioural deviation. We used Earth Movers' Stochastic Conformance (EMSC) [25], entropic relevance [26], Chi-squared [27], and trace probability [28].

Earth Movers' Stochastic Conformance (EMSC) uses Earth Movers' and string edit (Levenshtein) distance measures on probability distributions of traces in the log and the model. A higher value indicates a greater conformance with the expected behaviour. Entropic relevance gives the informational cost of describing a process model in the context of a particular log. Chi-squared is a measure based on the χ^2 statistic, using the event log as observed data and the process model as a hypothesised distribution. Trace probability uses a Markov chain to calculate the probability of observing a given trace under the model. This set could be extended in the future to capture other aspects of conformance to reference models. Unlike raw network features, these metrics encode global process-level information, making them more robust to noise and local variations. As we primarily want to distinguish traces according to different features, we construct an event log with a single trace for all measure calculations.

The machine learning technique and model used for classification can also be varied or tuned somewhat independently of the other components.

3.2. Process Conformance Metric Calculation

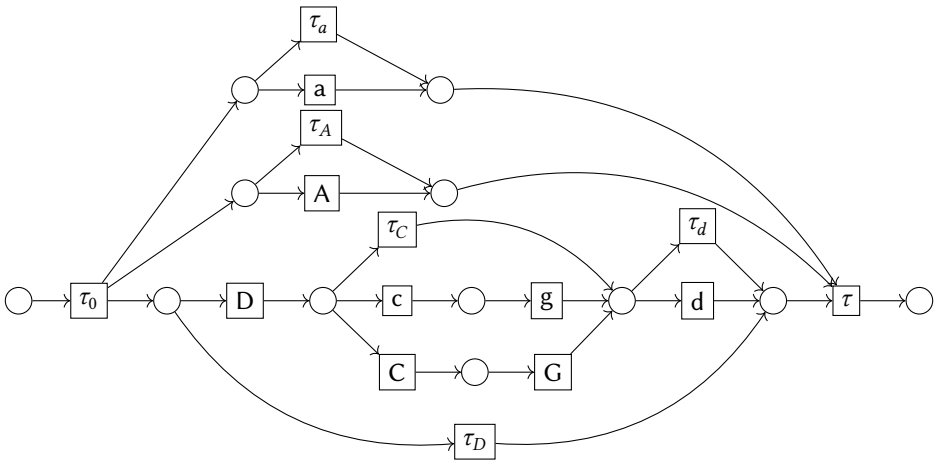

Figure 2: Section of Petri net for validating TCP protocol traces, discovered by Inductive Miner. Transition labels are protocol packet types as per Table 1; τ is a process mining convention for a silent transition.

To give a sense of how a Petri net model can be used to evaluate if a trace is conformant with the reference model, we compare two traces from the Zeek dataset: one fitting, one non-fitting. Following a convention established by Zeek, each type of packet is represented by a letter, as summarised in Table 1. Sequences of letters then describe the session history as a trace, with packets from the session initiator shown in upper case, and packets from the responder in lower case. The trace `Sh` represents a SYN (`S`) from the originator followed by a SYN-ACK (`h` - handshake) from the responder. The trace is fitting when each letter in sequence matches an enabled transition, which then fires, moving tokens that collectively represent the state.

Table 1: Zeek log protocol abbreviation dictionary. Upper case indicates packets from the connection initiator, and lower case the responder.

Letter	Meaning
a	Pure ACK
c	packet with a bad checksum
d	packet with payload (“data”)
f	packet with the FIN bit set
g	content gap
h	SYN-ACK (“handshake”)
s	SYN without ACK bit set

Figure 2 displays a section of our reference model in the form of a Petri net. The trace `ShADacgdFf` is a real trace from the IoT-23 dataset which matches our reference model, and we consider the corresponding subtrace for our figure, `ADacgd`. A single token starts in the leftmost place. The τ_0 silent transition creates three tokens in its successor places, now corresponding to three threads of execution. The middle token can fire the A transition, matching the first `A` event. To consume `D`, the bottom token fires the D transition. The top token can then fire transition a to consume the `a` event. The sequence `cgd` then corresponds to the bottom thread of execution progressing through the c, g and d transitions. All tokens synchronise on the rightmost silent τ transition to complete execution with a single token in the rightmost place, making `ADacgd` a fitting trace.

Similarly, we can take another trace `ShADacdtfF` from the IoT-23 dataset and compare the relevant portion, `ADacd`, to the Petri net reference model. In this example, the trace is only partially conformant. It follows the model for the initial sequence `ADac`, however, after c we see some deviation. According to the model, a transition c must be followed by a transition g and then an optional transition d (as the silent τ_d transition may fire instead). In our trace, the compulsory g transition is missing.

Conformance measures will quantify the degree to which traces match in different ways. Trace probability for non-matching traces will be zero, where measures such as EMSC will be sensitive to

partial trace matches, and entropic relevance will use a background model for non-matching traces.

4. Experimental Evaluation

The approach was evaluated using the IoT-23 dataset [8], which contains labelled network traffic. Each capture file represents a distinct scenario, such as a specific malware infection or normal device operation. Network connection level attributes include packet counts, byte volumes, and connection durations. 15 of the 23 available data captures were used, excluding the larger datasets for this initial investigation. Table 2 shows the main characteristics of the subset of the IoT-23 dataset used.

Table 2
Main characteristics of the subset of the IoT-23 dataset [8] used in our experiments.

File name	Total Connections	Benign	Malicious	Most Frequent Attack Type
1-1-conn.log.labeled	1,008,748	469,275	539,473	PartOfAHorizontalPortScan
3-1-conn.log.labeled	156,103	4,536	151,567	PartOfAHorizontalPortScan
8-1-conn.log.labeled	10,403	2,181	8,222	C&C
9-1-conn.log.labeled	6,378,293	22,548	6,355,745	PartOfAHorizontalPortScan
20-1-conn.log.labeled	3,209	3,193	16	C&C-Torii
21-1-conn.log.labeled	3,286	3,272	14	C&C-Torii
34-1-conn.log.labeled	23,145	1,923	21,222	DDoS
42-1-conn.log.labeled	4,426	4,420	6	FileDownload
44-1-conn.log.labeled	237	211	26	C&C
48-1-conn.log.labeled	3,394,338	3,734	3,390,604	PartOfAHorizontalPortScan
49-1-conn.log.labeled	5,410,561	3,665	5,406,896	PartOfAHorizontalPortScan
60-1-conn.log.labeled	3,581,028	2,476	3,578,552	DDoS
4B-1-B-conn.log.labeled	452	452	0	-
5B-1-B-conn.log.labeled	1374	1374	0	-
7B-1-B-conn.log.labeled	130	130	0	-
Overall	19,975,733	523,390	19,452,343	PartOfAHorizontalPortScan

4.1. Evaluation

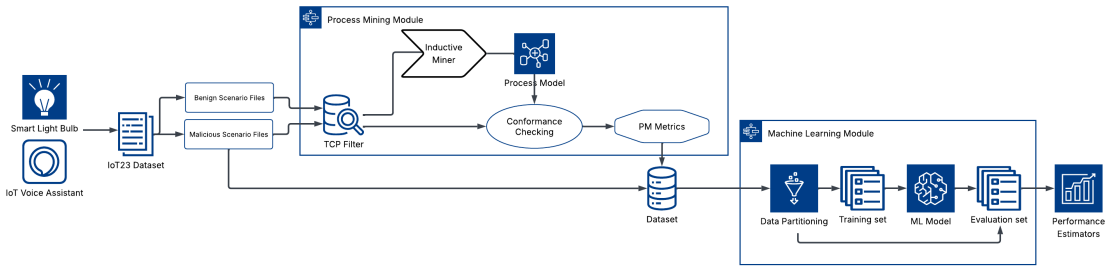

Figure 3: Evaluation pipeline for the IoT cyber-threat detection framework.

Figure 3 illustrates our experimental design. Network traffic from the IoT-23 dataset is processed through two main modules: Process Mining and Machine Learning. First, we process Zeek files obtained from the IoT device packet capture data. The process mining module processes benign and malicious scenario files separately, as they serve different purposes. Benign files only (i.e., the files with prefixes 4B-1-B, 5B-1-B, 7B-1-B) were used to create the reference model, which represents the normal communication behaviour of TCP. After obtaining the model of expected network behaviour, traces corresponding to connections from the remaining scenario files are used to calculate the conformance metrics. Once they are calculated, they are merged with the feature vectors of the corresponding

connections to form a dataset that is passed directly to the ML module. There, an ML model can be trained and evaluated.

The intrusion detection pipeline from Section 3.1 was implemented in Python [1]. Inductive miner [18] from `pm4py` was used to discover Petri net models, followed by stochastic weight discovery using the alignment estimator [24] in `ebi-pm`. The Random Forest classifier from `scikit-learn` was chosen due to its robustness and widespread use in intrusion detection research.

We used a Leave-One-File-Out (LOFO) evaluation strategy, where each capture file is used as a test set while training on the remaining files. As TCP sessions are contained within a single capture file, this ensured no single session appeared in both training and test data.

4.2. Reproduction of ML Experiments From Literature

Two existing papers also use IoT-23 data to train machine learning models for threat classification [22, 23]. In selecting appropriate benchmarks for our experiments, we have analysed and reproduced these experimental results. Both papers' experiments suffer from information leakage arising from the sequential nature of network sessions and connections. In [22], rows were arbitrarily selected from across all 23 captures and reorganised into three new datasets, each subsequently divided using a 7:3 ratio random train/test split. In [23], the dataset was loaded in full, then partitioned using an 8:2 random test/train split. In both cases, rows originating from the same network capture session, carrying similar endpoint addresses, were present in both training and testing sets. This constitutes a session-level data leakage, in which the test set is not truly unseen. It contains traffic from the same devices, and the corresponding labels (malicious or benign) were obtained from the same attack scenario already observed during training. As a result, models were trained on scenario-specific patterns rather than generalised IoT threat detection rules.

This is directly visible in the reported metrics. In [22] accuracy reported was ≈99%, precision ≈100%, recall ≈100%, and F1-score ≈100%, while [23] reported accuracy ≈99.97%, precision ≈99.98%, recall ≈99.97%, and F1-score ≈99.97%. These figures neither represent the model's genuine performance nor provide reliable evidence of its ability to detect intrusions in an unseen scenario. Evaluation protocols should reflect realistic deployment conditions. In operational settings, models must generalise to entirely new network environments and unseen attacks.

To validate the experimental setup of our and existing techniques, we compare two data splitting strategies:

- Random train/test split, where data is randomly partitioned across all captures.
- Leave-One-File-Out (LOFO), where each capture file is used as a test set while training on the remaining files.

We include this comparison to show that the LOFO strategy better reflects the real-world performance of cyberthreat detection models by ensuring that test data originates from entirely unseen network scenarios.

The results in Table 3 confirm the presence of data leakage when using random splits. While the random split produces near-perfect performance with a small standard deviation, the LOFO baseline results show a significant drop, for example to an F1-score of 0.6401, indicating a more realistic estimate of model performance.

4.3. Results

Table 3 also demonstrates that process mining features substantially improve classification performance compared to default network features alone. LOFO PM performance improves on the LOFO network baseline features across all estimation statistics, and includes a reduction of variance. This supports our hypothesis that conformance metrics capture meaningful behavioural information.

[1] Code and experimental scaffold available at https://github.com/adamburkegh/iot-cyber-pm.

Table 3
Comparison of quality metrics computed for two performance estimation approaches: (LOFO vs. random train/test split), and different feature sets (network features vs. network features + process conformance metrics). Reported as metric ±standard deviation.

Evaluation Strategy/Feature Set	Accuracy	Precision	Recall	F1 Score
Random split network baseline	0.9948 ±0.0001	0.9947 ±0.0001	1.000 ±0.0000	0.9973 ±0.0000
LOFO network baseline	0.7748 ±0.3004	0.6459 ±0.4222	0.8481 ±0.2896	0.6401 ±0.3876
LOFO PM	0.9559 ±0.1232	0.8973 ±0.2712	0.9993 ±0.0021	0.9067 ±0.2720

Feature importance analysis in Table 4 further shows that process mining metrics dominate the model's decision-making process. In particular, Earth Movers Stochastic Conformance and Chi-squared metrics contribute the most, at 0.3651 and 0.3567 respectively. Entropic Relevance also contributes to classification at importance 0.2050. Source and destination ports are the most important direct network features, at 0.0212 and 0.0132, much lower than most process mining features. Relying on values like specific ports or source IP addresses can also be brittle in the context of cybersecurity, as these may be easily changed by malicious actors with small code or configuration changes, or malicious code may itself be written to try multiple ports. Interestingly, the trace probability conformance metric was not of much importance in the machine learning model. Trace probability is an important input to other stochastic conformance metrics, so may not have provided much discrimination after those other metrics were prioritised in the constructed random forest model.

Table 4
Features used in the experiment, and their importance scores. Reported as score±standard deviation.

Features	Description	Importance Score
earth_movers_sc	**Earth Movers Stochastic Conformance** [25]	**0.3651 ±0.0522**
chi_squared	**Chi Squared** [27]	**0.3567 ±0.0472**
entropic_relevance	**Entropic Relevance** [26]	**0.2050 ±0.0136**
id.orig_p	Source port	0.0212 ±0.0172
id.resp_p	Destination port	0.0132 ±0.0178
orig_bytes	Source-to-Destination bytes	0.0109 ±0.0164
resp_ip_bytes	Flow of destination bytes	0.0092 ±0.0160
resp_pkts	Destination packets	0.0062 ±0.0138
orig_ip_bytes	Flow of source bytes	0.0059 ±0.0058
duration	Total duration	0.0027 ±0.0040
orig_pkts	Source packets	0.0020 ±0.0020
resp_bytes	Destination-to-Source bytes	0.0017 ±0.0018
trace_probability	**Trace Probability** [28]	**0.0003 ±0.0004**
missed_bytes	Missing bytes during transaction	0.0000 ±0.0000

The experimental results highlight several important findings. First, evaluation methodology plays a critical role in reported performance. Improper data splitting can lead to misleading conclusions about model effectiveness. Second, process mining features can provide a strong signal for anomaly detection, outperforming strictly network features in isolation. However, the variability observed in LOFO results suggests that model performance is sensitive to the specific characteristics of each network scenario. This indicates that further work is needed to improve robustness across diverse IoT environments.

5. Conclusion

In this paper, we propose a process-informed machine learning pipeline for IoT intrusion detection. Our approach leverages process mining to extract conformance-based features from network traffic and uses these features to train machine learning models. The proposed pipeline demonstrates that

combining process mining with machine learning is a viable approach for IoT threat detection. By transforming raw network traffic into structured behavioural representations, we enable the detection of anomalies that may not be visible through conventional feature engineering.

Through experimental evaluation on the IoT-23 dataset, we showed that process mining features significantly improve detection performance and provide meaningful insights into behavioural deviations. We also highlighted the importance of proper evaluation methodologies, demonstrating that commonly used random splits can lead to inflated results due to data leakage.

One limitation of the approach is the reliance on a single reference model derived from benign data. If the model does not adequately capture normal behaviour, conformance metrics may become less informative. Intrusion detection in real-time on IoT devices can also be expected to have tight computing and time resource constraints. In the current design, inference compute will be relatively cheap compared to model construction, but this deserves experimental and quantitative investigation that this approach would be viable in production use.

Overall, our findings suggest that process mining can complement and enhance existing machine learning approaches in cybersecurity. Future work may explore incremental process discovery, online conformance checking, and integration with streaming architectures to address these limitations. Particularly interesting would be extending our framework by including multiple reference models constructed for different definitions of the notion of cases and activity labels. Reference models could also be created for common malicious activity patterns, which could offer different insightful perspectives on IoT network data.

Acknowledgments

This research was supported by the Faculty of Science Strategic Research Investment Fund, SRIF Kickstarter - Anomaly detection in event streams using AI and process science for cybersecurity threat monitoring.

Declaration on Generative AI

During the preparation of this work, AI assistance was used in writing code for implementations and experiments. The authors have reviewed and edited all content and take full responsibility for it.

References

[1] Zscaler ThreatLabz, 2025 Mobile, IoT, and OT Threat Report, Technical Report, Zscaler, 2025. URL: https://www.zscaler.com/resources/industry-reports/threatlabz-mobile-iot-ot-report.pdf, online.

[2] H. D. Kotha, V. M. Gupta, IoT application: a survey, Int. J. Eng. Technol 7 (2018) 891–896.

[3] G. Kołaczek, Internet of things (IoT) technologies in cybersecurity: Challenges and opportunities, Applied Sciences 15 (2025). doi:10.3390/app15062935.

[4] E. Fieldstadt, Nest camera hacker threatens to kidnap baby, spooks parents, 2018. URL: https://www.nbcnews.com/news/us-news/nest-camera-hacker-threatens-kidnap-baby-spooks-parents-n949251.

[5] A. Titterington, The complete story of the 2024 ransomware attack on unitedhealth, 2025. URL: https://www.kaspersky.com/blog/unitedhealth-ransomware-attack/53065/, accessed 2026-03-29.

[6] W. van der Aalst, Process Mining: Data Science in Action, 2 ed., Springer-Verlag, 2016. doi:10.1007/978-3-662-49851-4.

[7] Y. Bertrand, J. De Weerdt, E. Serral, A bridging model for process mining and IoT, in: Process Mining Workshops, ICPM 2021, volume 433 of *LNBIP*, 2022, pp. 98–110.

[8] S. Garcia, A. Parmisano, M. J. Erquiaga, Iot-23: A labeled dataset with malicious and benign IoT network traffic, 2020. URL: https://doi.org/10.5281/zenodo.4743746. doi:10.5281/zenodo.4743746.

[9] H. El-Sofany, S. A. El-Seoud, O. H. Karam, B. Bouallegue, Using machine learning algorithms to enhance IoT system security, Scientific Reports 14 (2024) 12077.
[10] L. Xiao, X. Wan, X. Lu, Y. Zhang, D. Wu, IoT security techniques based on machine learning: How do iot devices use ai to enhance security?, IEEE Signal Processing Magazine 35 (2018) 41–49. doi:10.1109/MSP.2018.2825478.
[11] A. Janusz, S. Kalukapuge, M. T. Wynn, ThreatTrace: Cyber-attack detection through trace abstraction and soft clustering, in: Advanced Information Systems Engineering - 37th International Conference, CAiSE 2025, Vienna, Austria, June 16-20, 2025, Proceedings, Part II, LNCS, Springer, 2025, pp. 205–222. doi:10.1007/978-3-031-94571-7_12.
[12] Y. Zhong, A. Lisitsa, Can process mining help in anomaly-based intrusion detection?, 2022. arXiv:2206.10379.
[13] S.-E. Samiri, Z. Lamghari, R. Fakhar, Towards the application of process mining for analyzing network security, Statistics, Optimization & Information Computing 15 (2025) 1151–1161. doi:10.19139/soic-2310-5070-2557.
[14] S. Coltellese, F. Maria Maggi, A. Marrella, L. Massarelli, L. Querzoni, Triage of IoT attacks through process mining, in: On the Move to Meaningful Internet Systems: OTM 2019 Conferences, 2019, pp. 326–344.
[15] S. Tadi, Process mining driven by deep learning for anomaly detection in intelligent automation systems, Journal of Scientific and Engineering Research 11 (2024) 317–329.
[16] M. Macak, I. Vanát, M. Merjavý, T. Jevočin, B. Buhnova, Towards process mining utilization in insider threat detection from audit logs, in: International Conference on Social Networks Analysis, Management and Security (SNAMS), 2020, pp. 1–6. doi:10.1109/SNAMS52053.2020.9336573.
[17] P. Empl, F. Böhm, G. Pernul, Process-aware intrusion detection in mqtt networks, in: Proceedings of the fourteenth ACM conference on data and application security and privacy, 2024, pp. 91–102.
[18] S. J. J. Leemans, D. Fahland, W. M. P. van der Aalst, Scalable process discovery and conformance checking, Software & Systems Modeling 17 (2018) 599–631. doi:10.1007/s10270-016-0545-x.
[19] C. W. Günther, W. M. P. van der Aalst, Fuzzy mining – adaptive process simplification based on multi-perspective metrics, in: Business Process Management, 2007, pp. 328–343.
[20] Y. Zhong, J. Y. Goulermas, A. Lisitsa, Process mining algorithm for online intrusion detection system, in: Tools and Methods of Program Analysis, 2024, pp. 15–25.
[21] J. F. Kurose, K. W. Ross, Computer networking : a top-down approach, eighth edition. ed., Pearson Education Limited, 2022.
[22] C. V. Oha, F. S. Farouk, P. P. Patel, P. Meka, S. Nekkanti, B. Nayini, S. X. Carvalho, N. Desai, M. Patel, S. Butakov, Machine learning models for malicious traffic detection in IoT networks /IoT-23 dataset/, in: Machine Learning for Networking MLN2021, volume 13175 of *LNCS*, 2022, pp. 69–84.
[23] J. Jose, J. E. Judith, Unveiling the IoT's dark corners: anomaly detection enhanced by ensemble modelling, Automatika 65 (2024) 584–596.
[24] A. Burke, S. J. J. Leemans, M. T. Wynn, Stochastic Process Discovery by Weight Estimation, in: Process Mining Workshops, ICPM 2020, LNBIP, 2021, pp. 260–272. doi:10.1007/978-3-030-72693-5_20.
[25] S. J. J. Leemans, W. M. P. van der Aalst, T. Brockhoff, A. Polyvyanyy, Stochastic process mining: Earth movers' stochastic conformance, Information Systems 102 (2021) 101724. doi:10.1016/j.is.2021.101724.
[26] H. Alkhammash, A. Polyvyanyy, A. Moffat, L. García-Bañuelos, Entropic relevance: A mechanism for measuring stochastic process models discovered from event data, Information Systems 107 (2022) 101922. doi:10.1016/j.is.2021.101922.
[27] T. Li, S. J. J. Leemans, A. Polyvyanyy, Applying statistical distances for stochastic conformance checking (2026). Under review.
[28] S. J. Leemans, F. M. Maggi, M. Montali, Enjoy the silence: Analysis of stochastic petri nets with silent transitions, Information Systems 124 (2024) 102383.