Anomaly Detection Based on Behavior Feature Correlation for IoT Systems

Yifan Lu[1], Qixiao Lin[1], Jian Mao[1,2,3]*, Qiange Liu[1],
Ziwen Liu[1], Yaodong Zhang[1], Yan Huo[4]
{luyifan@buaa.edu.cn, linqx529@buaa.edu.cn, maojian@buaa.edu.cn,}
{liuqiangebuaa@buaa.edu.cn, liuziwen@buaa.edu.cn, cstzyd@buaa.edu.cn, yhuo@bjtu.edu.cn}
[1]School of Cyber Science and Technology, Beihang University, Beijing, China
[2]Tianmushan Laboratory, Hangzhou, China
[3]Hangzhou Innovation Institute, Beihang University, Zhejiang, China
[4]School of Electronics and Information Engineering, Beijing Jiaotong University, Beijing, China

Abstract. In Internet of Things (IoT) systems, automation rules are crucial for the intelligent control of devices. However, their reliance on event-driven logic is vulnerable to various security threats, such as event injection and command interception, which can disrupt system operations and compromise safety. Existing approaches typically extract state correlations from event sequences segmented by fixed time windows. These approaches struggle to identify long-range, multi-hop dependencies and can be bypassed by adversarial event sequences that partially mimic legitimate patterns, leading to false negatives. To address these, in this paper, we propose an anomaly detection method based on behavior feature correlation. Our approach models inter-device state correlations using a heterogeneous graph structure. It partitions behavior patterns through iterative community detection and automated semantic annotation, and represents normal behavior by embedding and clustering of state sequences. During the detection phase, anomalies are identified by measuring the similarity between the representation of a test sequence and the established benign profile. Experimental results demonstrate that our anomaly detection approach enhances precision by 1 time and recall by 3.2 times, compared to the leading baseline method.

Keywords: Internet of Things; IoT security; community detection; behavior feature representation.

1 Introduction

With the rapid development of automation technology, Internet of Things (IoT) systems have been widely applied in various fields, such as smart homes[1], industrial control[2], and smart

*Corresponding author.

IIKI 2025, December 18-21, Chengdu, People's Republic of China

DOI 10.4108/eai.18-12-2025.2365298

cities[3]. However, related security problems have attracted increasing concern[4, 5], including device faults[6, 7], network-based threats[8, 9] and side channel attacks [10, 11]. The introduction of automation rules considerably expands the attack surface. Existing works[12, 13] primarily focus on extracting state correlations from automation rules, physical channel properties, and historical event logs. These works typically segment continuous event streams using fixed time windows to learn normal behavioral patterns and identify deviations. However, as the IoT networks and rule sets grow in scale and complexity, an increasing number of state correlations exhibit long-range and multi-hop characteristics, which existing approaches fail to capture. Furthermore, attackers with domain knowledge can infer physical channel correlations and construct partly expected fake event sequences to bypass correlation-based detection mechanisms.

Existing methods suffer from two fundamental limitations. First, the prevalent reliance on fixed-time segmentation artificially fragments continuous activities. This processing obscures the semantic integrity of behaviors, which often span uncertain time windows and involve complex device interactions. Second, these methods focus on the co-occurrence or immediate succession of events without interpreting their higher-level meaning. As they recognize patterns based on superficial event sequences, any injected sequence that statistically resembles a legitimate pattern can potentially evade detection, even if it is semantically nonsensical in context.

To address such concerns, we propose a novel anomaly detection approach based on behavior feature correlations. We design a heterogeneous graph to model the correlations among device states. Based on the graph, we employ an iterative community detection algorithm coupled with automated semantic annotation to partition the event streams into distinct behaviors. For each behavior, we generate embeddings from state sequences and cluster them to construct a benign representation. During the anomaly detection phase, we assess the similarity between the test representation and the benign representation from the perspectives of edge consistency and node consistency to spot deviations in state.

In summary, we make the following contributions:

- We propose a novel fine-grained correlation graph representation that can effectively model the complex dependencies between device states in IoT, providing a structured foundation for subsequent analysis.
- We propose an iterative behavior partitioning method that automatically abstracts low-level device state sequences into high-level behaviors. By combining iterative community detection with automated annotation, our method achieves semantic representation.
- We introduce an anomaly detection approach based on behavior feature representation, which identifies deviations from two perspectives: edge consistency and node consistency. Compared to the leading baseline method, our approach increases precision and recall by 1 time and 3.2 times, respectively.

2 Problem Analysis

Existing anomaly detection works artificially fragment continuous activities, obscuring the semantic integrity of behaviors that span variable time lengths. Consequently, an increasing proportion

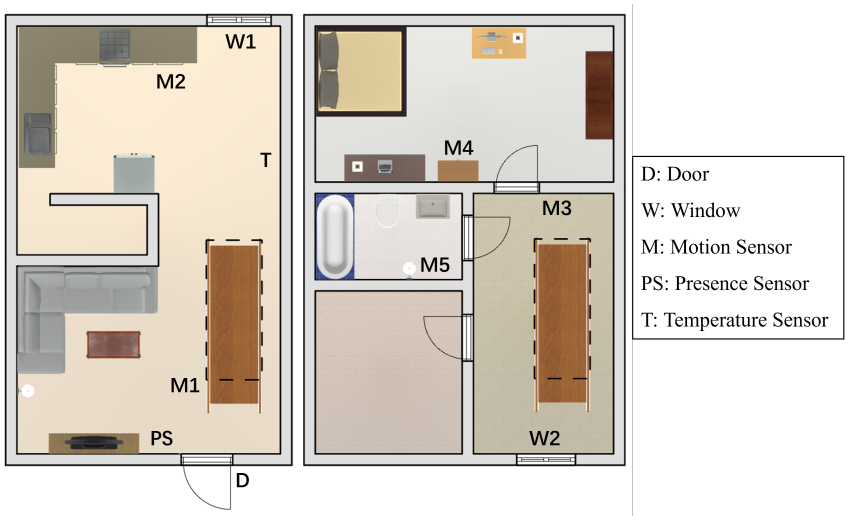


Fig. 1. The room layout of the motivating example.

of long-range and multi-hop state correlations remains difficult to capture effectively. To illustrate this problem, we design a two-story smart home scenario, as shown in Figure 1. The environment is equipped with five categories of devices. Besides, a sample automation rule is defined: "If T exceeds 28°C and PS is present, open W1 and W2."

In this system setting, an attacker can inject false events indicating "T is 29°C" and "PS is present". Such an injected event may trigger the preset rule, causing the windows to open when no one is home. Existing anomaly detection approaches extract a correlation: *the state of PS should only change after D is opened*. The initial injection would violate this correlation and be detected. However, an informed attacker can bypass detection by injecting a fabricated event sequence of "D is open - PS is present".

The limitations become even more pronounced for multi-hop correlations. For example, if an intruder enters through W2 and triggers M3, existing methods may fail. At this point, the user can choose to go to different rooms, causing M1, M4, and M5 to potentially correlate with M3. This makes it impossible for existing approaches to determine the specific device correlation, leading to false negatives. Although incorporating the state of the presence sensor PS seems intuitive, as motion sensors should be triggered only when a user is at home, the time intervals and state interference between PS and M3 prevent existing approaches from extracting such long-range dependencies from logs.

This analysis reveals the central **challenge**: confinement to low-level states. Existing methods operate primarily at the level of individual device states and their immediate correlations. The reliance on predefined windows severs the natural semantic continuity of extended activities. Consequently, systems cannot holistically model complete behaviors, making them blind to anomalies that only become apparent when the complete behavior is considered.

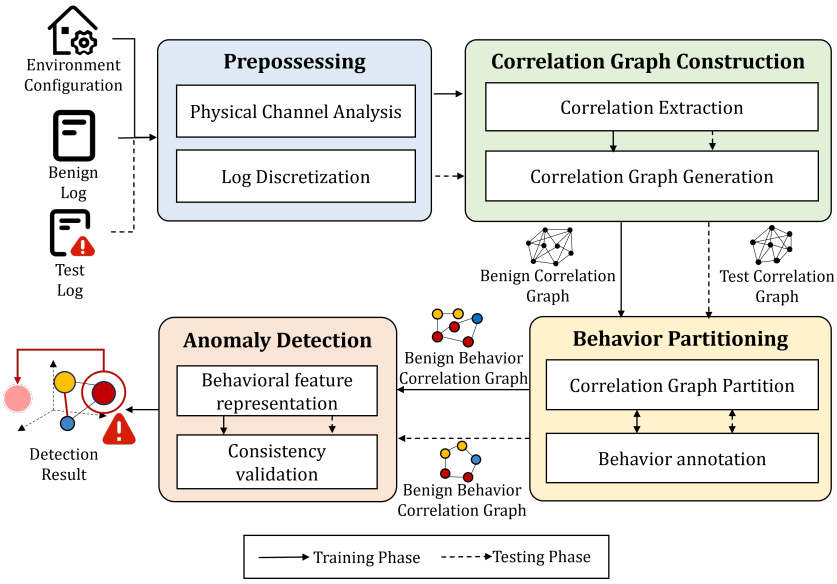


Fig. 2. The architecture of our work.

To address the challenge, we propose an anomaly detection system centered on behavioral semantic modeling. We construct a heterogeneous graph to comprehensively model the correlations between device states and employ iterative community detection for adaptive behavior partitioning. Using embedding techniques, we transform state sequences within each behavior into high-dimensional behavior feature representations.

3 System Design

In this section, we introduce the architecture and workflow of our approach, as shown in Figure 2. Our approach consists of four main modules: 1) Preprocessing, 2) Correlation Graph Construction, 3) Behavior Partitioning, and 4) Anomaly Detection.

The workflow of our work can be divided into two phases: the training phase and the testing phase. During the training phase, benign logs undergo the Preprocessing, Correlation Graph Construction, and Behavior Partitioning modules to build the behavior correlation graph. The Anomaly Detection module subsequently models benign behavioral representations. In the testing phase, the Anomaly Detection module processes the incoming graph solely through state sequence extraction and embedding representation. It then performs consistency checks between test and benign behavior representations to identify anomalous edges and embeddings.

3.1 Preprocessing

This module performs data refinement to facilitate downstream anomaly detection, comprising physical channel correlation analysis and log discretization.

Physical Channel Correlation Analysis. To infer physical correlations, we design a novel approach leveraging Natural Language Processing (NLP) techniques. We encode textual descriptions and channel information into high-dimensional vectors and compute cosine similarity between these embeddings. If the cosine similarity of a device and a physical channel exceeds the predetermined threshold, we consider the correlation to be reliable and mark the device with the physical channel.

Log Discretization. Continuous device values are discretized to prevent state explosion. Unlike prior work that partitions based only on numerical distribution, our approach intelligently incorporates domain knowledge from automation rules to define semantically significant intervals. For each continuous device, if its value triggers a rule, the rule's state values are used as discretization breakpoints; otherwise, we apply the Jenks natural breaks classification algorithm[14] to determine binary classification breakpoints. Finally, we update the original logs by replacing continuous values with their corresponding discrete state labels while eliminating duplicates.

3.2 Correlation Graph Construction

To comprehensively capture the complex interactions within a smart environment, we propose a weighted heterogeneous graph $G = (V, E)$, where the set of vertices V represents device states and the set of edges E denotes correlation between two states. Each edge $e \in E$ is assigned a numerical weight $W(e)$ that quantifies the strength of the underlying correlation derived from statistical analysis of the event logs. Each vertex is characterized by five attributes: $Name$, $State$, $Time$, $Location$ and $Channel$. We define six fundamental categories of correlation from the sequence of log state transitions. Formal representations of the correlations are defined as follows:

(1) **Time Correlation** (R_{Time}): This correlation exists between two log records that occur adjacently or in close temporal proximity, formally expressed as $v_j = v_i.Next \cup (v_j.Time - v_i.Time) < T_t$.

(2) **Device Correlation** (R_{Device}): This correlation links log records that involve the same device, formally expressed as $v_i.Name = v_j.Name$.

(3) **Rule Correlation** (R_{Rule}): This is a direct causal link where one log record triggers another through an automation rule, formally expressed as $\exists R \in Rules, v_i.State \in R.Trigger \cap\ v_j.State \in R.Action \cap\ (v_j.Time - v_i.Time) < T_r$.

(4) **Location Correlation** ($R_{Location}$): This correlation connects log records from devices located within the same geographical region, formally expressed as $v_i.Location = v_j.Location \cap\ v_i.Name != v_j.Name \cap\ (v_j.Time - v_i.Time) < T_l$.

(5) **Channel Correlation** ($R_{Channel}$): This correlation links devices that interact through a shared physical channel, formally expressed as $v_i.Channel \cap v_j.Channel != \emptyset \cap\ v_i.Name != v_j.Name \cap\ v_i.Location = v_j.Location \cap\ (v_j.Time - v_i.Time) < T_c$.

(6) **Sequential Correlation** ($R_{Sequence}$): This higher-order correlation captures recurring temporal patterns, formally expressed as $v_i.State = v_j.State \cap\ v_i.Next.State = v_j.Next.State$.

During graph generation, each device state derived from the discretized logs is represented as a vertex. Edges are created between vertex pairs that satisfy any correlation rule, with weights cumulatively updated if multiple correlations apply. The final representation of the correlation graph $G = (V, E)$ is generated through exhaustive log traversal.

3.3 Behavior Partitioning

This module presents a hierarchical behavioral partitioning framework to transform low-level device states into high-level semantic behaviors.

Correlation Graph Partition. Our approach employs community detection based on modularity[15]. This approach segments the graph into sub-communities by optimizing the modularity metric, which quantitatively measures the density of connections within communities. The modularity Q is formally defined as:

$$Q = \frac{1}{2m}\sum_{i,j}[A_{ij} - \frac{k_i k_j}{2m}]\delta(c_i, c_j) \tag{1}$$

$$\delta(c_i, c_j) = \begin{cases} 1 & c_i = c_j \\ 0 & c_i \neq c_j \end{cases} \tag{2}$$

where A_{ij} represents the weight of the edge between vertex i and j, k_i,k_j indicate the vertex degrees, m is the total sum of all edge weights and c_i,c_j denote the communities of the vertices.

The Louvain algorithm[16] is a classical modularity optimization approach for community detection. Each iteration assigns vertices to communities based on modularity gain and aggregates communities into super-vertices. We repeat the process until a global maximum of modularity is achieved and no further changes occur.

Behavior Annotation. To bridge the semantic gap between low-level device states and high-level user behaviors, we establish a state-behavior mapping model to automatically annotate the partitioned communities. However, identical device states may occur across distinct behavioral patterns, signifying divergent semantic interpretations. We quantify the contextual significance of a state within a specific community using an adaptation of the Term Frequency-Inverse Document Frequency (TF-IDF) algorithm. We formally define the significance of a state s_i in a community c_j as:

$$\omega(s_i, c_j) = \frac{f_{s_i,c_j}}{\sum_{s_k} f_{s_k,c_j}} \times log\frac{N}{n_{s_i}} \tag{3}$$

where $f_{s,c}$ denotes the frequency of state s in community c, N represents the total number of communities and n_s indicates the number of communities containing state s.

We calculate the importance of each device state within its assigned community. These states are then mapped to candidate behavioral labels via the pre-trained state-behavior model. The prob-

ability that community c_i exhibits behavior b_j is given by:

$$P(c_i, b_j) = \frac{\sum_{s_k \in S_{ij}} \omega(s_k, c_j)}{\sum_{b_m \in B} \sum_{s_k \in S_{im}} \omega(s_k, c_i)} \tag{4}$$

where S_{ij} denotes the set of states within community c_i that are mapped to behavior b_j, and B represents the set of all possible behaviors. If the probability $P(c_i, b_j)$ exceeds a predefined threshold, the community c_i is annotated with the behavioral label b_j.

Iterative Optimization. Due to the inherent greedy nature of the Louvain algorithm, a single execution of community detection may converge to a local optimum. To address this limitation, we propose an iterative optimization-based behavior partitioning algorithm designed to achieve consistent and reliable behavior semantic extraction.

Specifically, for the initial correlation graph G^0, we perform community detection and behavior annotation. All vertices belonging to communities that are confidently labeled are removed. The remaining graph is considered the residual graph G^1 for the next iteration. This process repeats until the residual graph is empty or its vertex size stabilizes. Any remaining unlabeled communities in the final residual graph G^t are assigned the behavioral label with the highest probability $P(c_i, b_j)$. Finally, we integrate all labeled sub-communities obtained throughout all iterations to form the final behavior correlation graph.

3.4 Anomaly Detection

This module detects anomalies by performing dual-perspective consistency validation against established benign behavioral patterns, using representations derived from the Behavior Correlation Graph.

Behavior Feature Representation. The approach begins by extracting inter-community relationships from the behavior correlation graph. For any pair of communities c_i and c_j, we define a binary correlation function $\rho(c_i, c_j)$. Specifically, $\rho(c_i, c_j) = 1$ if a time or rule correlation exists from c_i to c_j, otherwise $\rho(c_i, c_j) = 0$. For each community c_i, we identify an outgoing boundary vertex set $O(c_i)$ and an incoming boundary vertex set $I(c_i)$. These sets serve as anchors for verifying the structural consistency of cross-behavior interactions during anomaly detection.

Within each community, we extract state sequences and transform them into embedding vectors via a sentence embedding model. All benign sequence embeddings of a community are clustered. We adaptively determine the optimal number of clusters k^* to circumvent the limitations associated with preset cluster counts. For each cluster $Cluster_i$, we compute its centroid μ_i and the associated benign radius d_i:

$$\mu_i = \frac{1}{N_i} \sum_{e \in Cluster_i} e \tag{5}$$

$$d_i = \max\{\| e - u_i \|_2 \, | e \in Cluster_i\} \tag{6}$$

where N_i is the number of points in $Cluster_i$. This process yields two critical sets for the behavior community: the benign centroid set μ_i and the benign distance set d_i.

Consistency Validation. We perform anomaly detection through edge-level consistency verification and vertex-level consistency verification.

In edge-level consistency verification procedure, for edge (v_i, v_j) connecting different communities c_i and c_j, if $\rho(c_i, c_j) = 0$, the edge is flagged as anomalous. If the correlation is valid, the condition $v_i \in O(c_i) \wedge v_j \in I(c_j)$ must hold; otherwise, the edge is treated as an anomaly.

In vertex consistency verification procedure, for a state sequence embedding e extracted from a community c in the test graph, let $M_c = \{\mu_i\}_{i=1}^{N_c}$ denote the set of benign cluster centroids for that community, and $D_c = \{d_i\}_{i=1}^{N_c}$ represent the corresponding set of benign radii, where N_c is the number of benign clusters in c. The embedding e is flagged as anomalous if it lies outside all benign clusters:

$$\forall \mu_i \in M_c, \| e - \mu_i \|_2 > d_i \tag{7}$$

4 Implementation and Evaluation

This section describes the implementation and evaluation of our approach. For semantic embedding of state sequences, we use the all-mpnet-base-v2 model[17] for state-sequence embeddings. Correlation weights are tuned via grid search. During detection, we apply Principal Component Analysis (PCA) to reduce the dimensionality and cluster them with K-Means to build normal behavioral profiles.

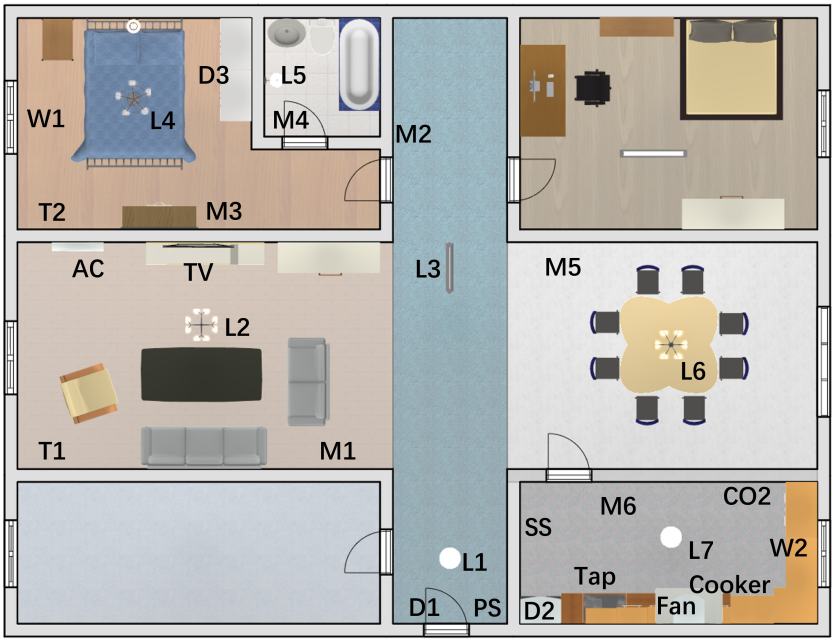
4.1 Experiment Setting

We construct a smart home testbed with 28 devices across 7 areas to emulate a realistic residential environment, as shown in Figure 3. Besides, 11 corresponding automation rules are defined in Table 1. To accurately simulate real-life user scenarios, we design multiple routine behavioral patterns with natural randomness and sensor-value variations. All activities are simulated to represent a single resident over a continuous three-day period, comprising two weekdays and one weekend day. Detailed logs are collected using the Home Assistant platform[18], forming a benign training dataset.

Based on the benign dataset, we construct an anomaly dataset covering three IoT-specific threat categories: false events, event losses, and command failures. We give special attention to complex anomalies involving long-range, multi-hop correlations and multi-state violations, which are the main problems we want to solve.

4.2 Anomaly Detection Evaluation

To comprehensively evaluate the performance of our anomaly detection approach, we employ three standard evaluation metrics: Precision (P), Recall (R), and F1-score ($F1$). We perform a comparative analysis against two established anomaly detection schemes in smart home environments: HAWatcher [11] and Veritas [12]. The experimental results, summarized in Table 2, demonstrate that our approach achieves superior performance in all metrics. Specifically, compared to the strongest



L: Light	D: Door	W: Window	M: Motion Sensor
PS: Presence Sensor		SS: Sound Sensor	CO2: Carbon Dioxide Sensor
Fan: Exhaust Fan		Cooker: Cooker	T: Temperature Sensor
AC: Air Conditioner		Tap: Water Tap	TV: Television

Fig. 3. The room layout of our testbed.

baseline HAWatcher, our method improves precision by 1 time and recall by 3.2 times, yielding a substantially higher F1-score.

To further illustrate the practical effectiveness of our approach, we present two detailed case studies of different attack scenarios.

Case 1: False Event Injection After User Leaves. We assume that an attacker injects a false event "M3 is active" after the user has left home. In the benign behavior graph, no edge exists between the "Leave Home" and "Sleep" behaviors, as these activities are semantically and temporally distinct. However, the injected false event creates an anomalous edge that connects these two behaviors, violating edge-level consistency. This inconsistency enables our method to detect the anomaly and precisely localize the abnormal state node.

Intuitively, a correlation should exist: when the motion sensor M3 is active, the presence sensor PS must be present. While this multi-hop dependency is captured by our behavior-aware approach, HAWatcher fails to identify this anomaly, as it can only detect correlations between temporally close

Table 1: Automation rules used in our testbed

Rule	Trigger	Action
R1	PS is present	Turn on L1, L2
R2	PS is away	Turn off L1, L2, W1, W2
R3	M1 is active	Turn on TV
R4	T1 $>$ 30 AND M1 is active	Turn on AC
R5	M2 is active	Turn on L3
R6	M2 is inactive for 15s	Turn off L3
R7	M4 is active	Turn on L5
R8	M4 is inactive	Turn off L5
R9	T2 $>$ 28 AND PS is present	Turn on W1
R10	CO2 $>$ 1000	Turn on Fan, W2
R11	CO2 $<$ 700	Turn off Fan

Table 2: Comparisons for different anomaly detection approaches

Approach	Precision	Recall	F1-score
HAWatcher	47.83%	22.45%	30.56%
Veritas	50.00%	18.37%	26.87%
Our approach	**95.83**%	**93.88**%	**94.84**%

events and is unable to capture such multi-hop logical dependencies.

Case 2: Command Interception During Cooking Activity. In this case, an attacker intercepts the command of automation rule R10 while the user is cooking. We visualize the clustering results of all state sequences within the "Cooking" behavior community in Figure 4. As shown, our approach identifies five distinct benign clusters, representing the multiple behavioral modes included in the "Cooking" behavior. Among all behavioral representations, an outlier deviating from all benign clusters is detected, indicating an anomalous state sequence caused by the interception of the R10 command.

HAWatcher fails to identify this anomaly due to its preprocessing methodology, which discards continuous values from automation rules. After discretization, it becomes incapable of determining the specific state transitions that should trigger particular rules.

5 Related Work

For anomaly detection in IoT environments, many studies analyze potential relationships to predict normal behavior. Such correlations provide reference standards, enabling researchers to perform consistency checks between current states and expected states[8, 12, 19, 20, 21]. Zhang et al.[19] infer a deterministic finite automaton(DFA) from app code and text information. They in-

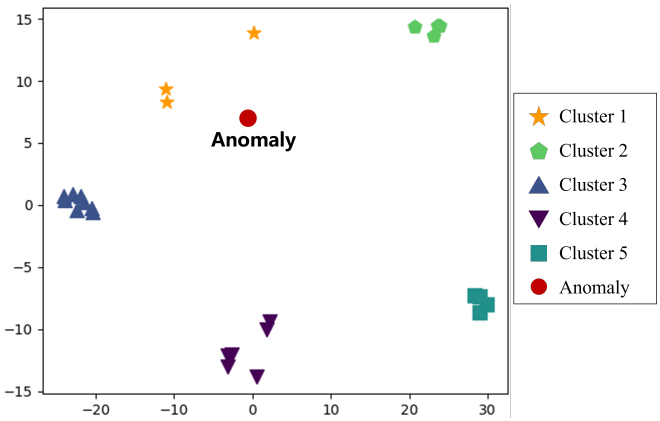


Fig. 4. The case of anomaly detection based on node consistency.

fer communication event types via packet size and intervals and compare inferred event sequences against expected state transitions in the DFA. Fu et al.[12] validate semantic correlations extracted from rules and configurations using log data and simulate behavior to verify consistency between predicted and real events. Lin et al.[8] construct a device-centric behavior graph from automation rules and generate an execution graph by matching normal and runtime traffic to derive causal sequences of rules. They verify consistency by learning and comparing topological, node, and edge features between the two graphs. Li et al.[21] build a knowledge graph with encrypted traffic through fingerprint matching. A feature separation-based heterogeneous graph attention network is designed to detect anomalies in the knowledge graph.

Researchers also establish numerical criteria based on probability, distance, or anomaly scores, where exceeding a threshold indicates a security issue[13, 22, 23, 24]. Ding et al.[22] cluster all baseline intra-app interactions to form a benign boundary. Any testing interaction behavior vector that does not belong to such clusters will be marked as risky. Wang et al.[23] extend causal graphs with temporal factors to define a new device interaction graph. They estimate conditional probability tables via maximum likelihood estimation and therefore compute anomaly scores for incoming events at runtime. Babun et al.[13] build a first-order Markov chain model to compute the probability of observed state sequences for anomaly detection. Rieger et al.[24] model the IoT environment using an unsupervised deep neural network trained on benign data to reconstruct inputs, and detect malicious events by evaluating the reconstruction error of new events.

6 Conclusion

In this paper, we propose an anomaly detection approach based on behavior feature correlation to address the challenges of capturing long-range dependencies and detecting correlation bypass attacks in IoT systems. Our approach models the complex correlations among device states via a fine-grained heterogeneous graph structure. The correlation graph is partitioned using iterative

community detection enhanced with semantic annotation to transform low-level device states into high-level semantic behaviors. We construct representations of normal behavioral characteristics by embedding and clustering state sequences within each pattern. Test behaviors are validated through dual-perspective checks on edge consistency and vertex consistency. We detect states that deviate from normal patterns by evaluating similarity between test and benign representations. To evaluate the effectiveness of our approach, we perform a comparative experiment with HAWatcher and Veritas in a smart home testbed. Experimental results demonstrate that our method significantly outperforms the best baseline HAWatcher, improving precision and recall by 1 time and 3.2 times, respectively. It shows superior capability in identifying complex multi-hop correlations and detecting advanced threats, such as correlation bypass attacks.

Acknowledgments

This work was supported in part by the Zhejiang Provincial Natural Science Foundation of China (No. LZ23F020013) and the National Natural Science Foundation of China (No. 62172027, No. U24B20117).

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] Smart Home Appliances Global Market Report 2025; 2025. The Business Research Company. Available from: https://www.researchandmarkets.com/reports/6033303/smart-home-appliances-global-market-report.

[2] Industrial Automation And Control Systems Market (2025 - 2030); 2024. Grand View Research. Available from: https://www.grandviewresearch.com/industry-analysis/industrial-automation-market.

[3] Smart Cities Market (2025 - 2030); 2025. Grand View Research. Available from: https://www.grandviewresearch.com/industry-analysis/smart-cities-market.

[4] Hammi B, Zeadally S, Khatoun R, Nebhen J. Survey on smart homes: Vulnerabilities, risks, and countermeasures. Computers & Security. 2022;117.

[5] Wang Z, Liu D, Sun Y, Pang X, Sun P, Lin F, et al. A Survey on IoT-Enabled Home Automation Systems: Attacks and Defenses. IEEE Communications Surveys & Tutorials. 2022;24(4):2292-328.

[6] Xue L, Yan Y, Tang Q, Yu L, Luo X, Cai Z, et al. Update If You Dare: Demystifying Bare-Metal Device Firmware Update Security of Appified IoT Systems. IEEE Transactions on Dependable and Secure Computing. 2025;22(3):2367-84.

[7] Xiao H, Zhang Y, Shen M, Lin C, Zhang C, Liu S, et al. Accurate and Efficient Recurring Vulnerability Detection for IoT Firmware. In: Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security. Association for Computing Machinery; 2024. p. 3317–3331.

[8] Lin H, Li C, Yang J, Wang Z, Fan L, Duan C. CP-IoT: A cross-platform monitoring system for smart home. In: Proceedings of the Network and Distributed System Security Symposium. The Internet Society; 2024. .

[9] Antonakakis M, April T, Bailey M, Bernhard M, Bursztein E, Cochran J, et al. Understanding the Mirai Botnet. In: Proceedings of the 26th USENIX Conference on Security Symposium. USENIX Association; 2017. p. 1093-110.

[10] Pang ZH, Fan LZ, Dong Z, Han QL, Liu GP. False Data Injection Attacks Against Partial Sensor Measurements of Networked Control Systems. IEEE Transactions on Circuits and Systems II: Express Briefs. 2022;69(1):149-53.

[11] Chen Y, Yu J, Kong L, Zhu Y. A Comprehensive Survey of Side-Channel Sound-Sensing Methods. IEEE Internet of Things Journal. 2025;12(2):1554-78.

[12] Fu C, Zeng Q, Du X. HAWatcher: Semantics-Aware Anomaly Detection for Appified Smart Homes. In: Proceedings of the 30th USENIX Conference on Security Symposium. USENIX Association; 2021. p. 4223-40.

[13] Babun L, Sikder AK, Acar A, Uluagac AS. The Truth Shall Set Thee Free: Enabling Practical Forensic Capabilities in Smart Environments. In: 29th Annual Network and Distributed System Security Symposium. The Internet Society; 2022. .

[14] Jenks GF. The Data Model Concept in Statistical Mapping. vol. 7; 1967. p. 186190.

[15] Li T, Liu X, Qiao W, Zhu X, Shen Y, Ma J. T-Trace: Constructing the APTs Provenance Graphs Through Multiple Syslogs Correlation. IEEE Transactions on Dependable and Secure Computing. 2024;21(3):1179-95.

[16] Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment. 2008 Oct;2008(10):10008.

[17] all-mpnet-base-v2; 2022. Available from: `https://github.com/homer6/all-mpnet-base-v2`.

[18] Home assistant: Awaken your home; 2025. Available from: `https://www.home-assistant.io`.

[19] Zhang W, Meng Y, Liu Y, Zhang X, Zhang Y, Zhu H. HoMonit: Monitoring Smart Home Apps from Encrypted Traffic. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. Association for Computing Machinery; 2018. p. 1074–1088.

[20] Gu T, Fang Z, Abhishek A, Fu H, Hu P, Mohapatra P. IoTGaze: IoT Security Enforcement via Wireless Context Analysis. In: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications; 2020. p. 884-93.

[21] Li R, Li Q, Huang Y, Zou Q, Zhao D, Zhang Z, et al. SeIoT: Detecting Anomalous Semantics in Smart Homes via Knowledge Graph. IEEE Transactions on Information Forensics and Security. 2024;19:7005-18.

[22] Ding W, Hu H. On the Safety of IoT Device Physical Interaction Control. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. Association for Computing Machinery; 2018. p. 832–846.

[23] Wang J, Li Z, Sun M, Yuan B, Lui JCS. IoT Anomaly Detection Via Device Interaction Graph. In: 2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN); 2023. p. 494-507.

[24] Rieger P, Chilese M, Mohamed R, Miettinen M, Fereidooni H, Sadeghi AR. ARGUS: context-based detection of stealthy IoT infiltration attacks. In: Proceedings of the 32nd USENIX Conference on Security Symposium. USENIX Association; 2023. p. 4301-18.