Transformer-based Compound Correlation Miner for Smart Home Anomaly Detection

Andrew D'Angelo*, Chenglong Fu†, Xiaojiang Du*, and Paul Ratazzi‡
*Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA
†Department of Software and Information Systems, University of North Carolina at Charlotte, Charlotte, NC 28223, USA
‡Information Warfare Division, Air Force Research Laboratory, Rome, NY 13441, USA
Email: {adangelo, xdu16}@stevens.edu, chenglong.fu@uncc.edu, edward.ratazzi@afrl.af.mil

2023 IEEE 12th International Conference on Cloud Networking (CloudNet) | 979-8-3503-1306-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/CloudNet59005.2023.10490063

***Abstract*—IoT-enabled smart homes can be double-edged swords. On one side, the convenience and efficiency brought about by smart device integrations can improve the standard of living dramatically. However, on the other side, the prevalent interconnectivity among home devices can drastically increase the potential risk of attack. Therefore, in order to reduce the possibility of a successful attack we propose a complex correlation-based anomaly detection system powered by intricate two-to-one correlations. Through the use of a state-of-the-art transformer model, we present a novel correlation mining mechanism that leverages the power of attention weights to develop an understanding of the underlying correlations that exist between IoT events in a smart home environment. Using this knowledge, we use a special validation algorithm to verify 52 two-to-one correlations in our system. Furthermore, we simulate four distinct attack scenarios and attain an average detection accuracy and recall of 96.59% and 97.38% respectively. Our results indicate that our method is effective at identifying a range of IoT attacks and successfully demonstrates the capabilities of IoT correlations.**

***Index Terms*—Smart home, Security, Anomaly Detection**

I. Introduction and Related Works

The rapid development of smart home IoT technology has facilitated the emergence of home automation systems that encompass IoT devices, hubs, mobile apps, and cloud services. This new system architecture enables adaptable integrations of devices and services, along with advanced automation according to user-customized rules. However, these conveniences also come with significant security risks that can impose harm directly to the physical environment. Recent research has revealed various vulnerabilities in APIs that are opened for third-party vendors and developers [5], [9], [10], which may be exploited by attackers to manipulate IoT messages. These manipulated messages can further affect other devices in the same smart environment through automation rules and cause more severe consequences such as burglary and fire hazards.

To cope with these security issues and bring IoT users peace of mind, various security enhancement solutions have been proposed in a series of research works [7], [12]. The most prominent advancements fall in the category of smart home anomaly detection [6], [11], which aims to raise alarms whenever the behavior of any device in a smart home IoT system deviates from its regular routine. HAWatcher [11] proposes the novel concept of inter-device correlation that serves as a powerful tool to profile invariant device behaviors for accurate anomaly detection. However, both Peeves [6] and HAWatcher [11] use statistics-based solutions for capturing correlations between two devices and fail to generate more complicated correlations that involve multiple devices. Missing these more sophisticated correlations undermines their detection performance and limits their capabilities for detecting stealthy attacks.

In this paper, we focus on extracting compound correlations that have one more device in the anterior position, which is called "two-to-one" correlations. Because of the added device, the permutation space of hypothetical correlations will become too large that make the original method in HAWatcher no longer usable. Targeting this challenge, we present a novel correlation mining method that leverages the power of the self-attention layer of a transformer deep neural network model to correlate events of different devices to one another in an event sequence. Using these correlated events, we develop a comprehensive set of correlation rules that are used to flag anomalous behaviors caused by a range of attacks in the system. We evaluate the performance of our system through a comprehensive experiment on a smart home environment consisting of, a total of 20 IoT smart devices spanning nine different types. In the experiment, we successfully mine 57 unique two-to-one correlations and calculate a detection recall and accuracy as 96.59% and 97.38% respectively.

For reference, the paper is organized as follows. In Section II we present background information and investigate related works. Then, in Section III, we define the system model and examine different IoT-related attacks relevant to the study. Moreover, in Section IV, we break down correlations by type and complexity and include their corresponding notation. In Section V, we analyze the fundamental components of a transformer and highlight the specific parts we leverage in our system. Then in Section VI, we characterize our unique correlation mining process. Thereafter, in Section VII, we present our experiment including environment details, model

ACKNOWLEDGMENT: This work was supported in part by the US National Science Foundation (NSF) under grants CNS-2204785 and CNS-2205868.
DISTRIBUTION STATEMENT: A. Approved for public release; distribution unlimited. Case number AFRL-2023-3377, 17 Jul 2023.

configuration, selected correlation parameters, and so on. After that, we provide a thorough performance evaluation in Section VIII. Finally, we conclude the paper in Section IX.

II. Background and Related Works

Smart Home IoT systems epitomize the revolutionary advancement in sensing, communication, and intelligent control technologies. Typical smart home IoT systems, such as SmartThings [4], Alexa [2], and Google Home [3] all have devices connected to smart home hubs or cloud servers which offer publicly accessible APIs for the connection of third-party devices. The interaction of all connected devices is uniformly defined by a set of schema that contains the format and content of event and command messages. Automation rules are programmed using Trigger-Action-Programming [14] and triggered by event messages emitted by IoT devices. The execution of automation rules leads to commands that dictate actuator devices to make the corresponding actions. Since the runtime of automation rules does not have the built-in capability to detect IoT message anomalies, they can be violated by attacks for manipulating benign actuator devices [9].

HAWatcher [11] represents the state-of-the-art solution to detect anomalous events and device actions. It models the smart home IoT systems' normal behavior into a collection of correlations between two devices that captures their causal or co-occurring relationship. More specifically, it crafts a list of possibly correlated devices and uses them to derive hypothetical correlations. Then, it verifies all hypothetical correlations against a training dataset. When deployed, every event from the smart environment is checked against all accepted correlations and any violation will be reported as potential anomalies.

III. System and Threat Model

A. System Model

A smart home defines a IoT-enabled living space, outfitted with a range of smart sensors, actuators, and controllers. Most often, smart homes contain numerous devices that are inherently related to one another through some form of predefined rule or shared action. We consider devices that exhibit such inherent relationships to be correlated to one another. By nature, smart home devices communicate with each other using popular communication standards such as Zigbee, Z-wave, Bluetooth, or Wifi. Special platforms such as Samsung Smartthings, Apple Homekit, and Google Home, promote easy communication between devices in the system. In this paper, we consider a highly correlated smart home environment consisting of multiple Smartthings compatible devices that communicate using a range of different communication standards. Additionally, we engineer a system such that correlations are inherently influenced by device location and pre-defined automation rules.

B. Threat Model

In this section, we examine different IoT attacks published by recent studies that our method has the capability to detect. More specifically, we focus on attacks that assert a change that is reflected in the historical log of the smart home system.

Attacks on Events: We distinguish attacks on events as specialized attacks that work to undermine the integrity of IoT sensor devices. We bifurcate such attacks by way of their method of modification made to the system.

a) Event Injection: In event injection, an attacker works to maliciously inject a nefarious event into the system [17]. The goal of this attack can range from indirectly influencing an actuator, to attempting to disguise one's presence in the home or fool the homeowner to gain access to their residence. For example, in the situation that a person leaves their home, but accidentally leaves the door slightly open an attacker may inject a *front door closed* into the system to fool the homeowner to believe the front door is closed when in reality it's actually open. In this manner, the attacker can then enter the home through the door without the owner ever knowing.

b) Event Interception: For an event interception attack, an attacker can intercept events and prevent them from ever being registered by the smart home system. This form of attack can be devastating to the home because an attacker can use it to block critical events to gain access to the home, prevent an alert from indicating his or her presence on the property, or even prevent safety systems such as a fire alarm from registering a disaster.

Attacks on Commands: We designate attacks on commands as specific attacks that are executed on actuation devices to influence a certain state change or action in the environment.

c) Fake Commands: Fake commands describe a subset of injection attacks that are focused on injected actuator-specific events into the system to cause or prevent an action by that device. For instance, given a smart door lock, an attacker may inject an *unlock* command into the system to unlock the front door and gain access to the home. Moreover, an attacker may also choose to turn on power-consuming devices to overload the electric systems of the home.

d) Command Interception: In a command interception attack, a malicious actor attempts to block commands from executing to either gain unauthorized access to the premises or inflict damage to the system. One common scenario involves an attacker intercepting and discarding the *lock door* command as the user leaves their home, thereby gaining easy entry into the residence. Another example is when an attacker intercepts and alters a command regulating a home's thermostat. By manipulating the *set temperature* command, the attacker can cause the system to either overheat or become excessively cold, potentially causing damage to appliances and creating an inhospitable environment within the residence.

IV. Correlations In IoT-Enabled Environments

Correlations describe underlying relationships between device events per a variety of modalities. Such modalities include physical operation channels, user activity channels, and automation rules. We define correlations as a directed relationship between device events using the notation $E^{capability}_{device=value}$

where a *capability* is an event's operation mode. For example, some capabilities include "motion", "presence", "contact", etc. The capability is important for differentiating between separate event types that originate from the same device. For the purpose of this study, we define events as they appear in the SmartThings CLI historical logs. Mainly, the event log is an irregularly spaced time series consisting of categorical and continuous values. Moreover, events are only recorded when a state change in a device is initiated.

A. One-to-one Correlations

At the most basic level, a one-to-one correlation represents a simple directed relationship between two events or an event and a subsequent state in a smart system [11]. Given two events A and B we represent one-to-one correlations using the following form: $A \rightarrow B$. In this manner, we say that event A is correlated to event B such that there exists some inherent relationship between these events that, if violated, indicates an anomaly in the system.

One-to-one correlations exist as the foundation for more complex two-to-one correlations defined in Sec. IV-B. In terms of anomaly detection, one-to-one correlations stand as the first line of defense against nefarious attacks. Unfortunately, they are not perfect and possess certain gaps in their detection capabilities [11]. Our method addresses these limitations by targeting those correlations that don't meet a certain validation threshold. For those correlations that don't meet the threshold, we refine them by incorporating a third event, creating what we term as two-to-one correlations. Two-to-one correlations possess greater detection capabilities by adding complexity to the rules. In this way, an attacker has to consider more components when trying to infiltrate the smart system.

B. Two-to-one Correlations

Considering three events, A, B, and C, we define a two-to-one correlation as a bidirectional relationship where, under certain conditions, all three events occur together within a specific time frame with high statistical significance. We represent such correlations using the notation $A \wedge B \leftrightarrow C$. Here, $\wedge$ signifies that there is a dependence relationship between the first two events, A and B. The $\leftrightarrow$ symbol indicates that the correlation has a focal order involving all three events, A, B, and C. Furthermore, we designate that in the presence of A and B if the subsequent correlated C is missing, the correlation is violated and an anomaly should be flagged.

Two-to-one correlations expand upon the detection capabilities of one-to-one correlations highlighted in [11] by adding a third condition to the relationship. Specifically, we aim to discover two-to-one correlations that achieve statistical significance where their foundational one-to-one correlation roots have previously failed.

Notably, we consider two-to-one correlations as bi-directional formulae to consider cases where the order may not be preserved. For example, during times of severe network congestion, certain communications between devices and the smart hub may be delayed by some indeterminate amount of time. In such cases, two events that often happen at the same time might occasionally flip order in the log.

V. Transformer Framework

This section aims to uncover the underlying transformer architecture and highlight the specific modules we leverage to attain consistent and accurate event correlations.

A. Transformer Architecture

Transformers are powerful AI models that achieve state-of-the-art performance in a variety of complex tasks. The secret behind their success lies in their usage attention, a special mechanism that can learn context and meaning between inputs by tracking relationships in sequential data [15].

IoT devices are influenced by a wide variety of incidents in the smart system. As a result, it is important to consider the context of the event log from both directions. Therefore, we adopt the RoBERTa transformer model, a special sub-type of the BERT architecture that uses dynamic masking and byte pair encoding for performance optimization [8], [13]. Unlike most models, BERT considers the surrounding context of a token by considering both the tokens that come before it and after it. This is especially useful for IoT-related applications because a slight difference in an event sequence can mean the difference between two entirely separate actions taking place in the environment. Moreover, the surrounding context of the sequences might change the influence one event has on another event, a vital aspect to consider for correlation generation.

B. Token Embedding

By default, transformers do not perform recurrence or convolution; therefore, sequential order must be incorporated into the model separately. In RoBERTa, positional embedding is used to fill this gap by applying a learnable positional embedding layer to the model architecture. Thus, as the model is trained it learns a vectorized positional representation for each token that is appended to the token embedding to form a final representation for each input.

C. Multi-headed Self Attention

Attention is a critical component of Transformer models that allows them to focus on different parts of the input data with varying degrees of emphasis. In general, attention lets the model dynamically assess and combine information from any position in an input sequence, making it effective for handling long-range dependencies.

A single attention head is defined as the scaled dot product of *Query*, *Key*, and *Value* vectors represented as Q, K, and V respectively. An attention matrix defined by [15] is calculated using Equ. 1

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

where d_k is the dimension of the queries and keys in the input sequence. The attention score output is an NxN matrix where N is the number of tokens in the input sequence. A

single score at position (i, j) indicates how much token i, attends to token j with respect to the sequence at hand. For the purpose of understanding how different tokens relate to one another in a sequence, we use a special form of attention known as *Self Attention*. In Self Attention, the query, key, and value vectors are populated by different tokens from the same input sequence. Therefore, when we perform attention we are left with a matrix with scores that attend different tokens within the same sequence to one another.

Moreover, in Multi-Headed Self Attention, self-attention is performed h times for a single input such that multiple learned linear projections are considered for each input. The resultant matrices are concatenated by a typical transformer model, but for the purpose of correlation generation described in Sec. VI-A, we use the raw weights before concatenation. All in all, the true power behind Multi-headed Self Attention is its ability to capture the relationship between two tokens whilst considering a variety of representational subspaces.

VI. Correlation Mining

This section describes the underlying procedure of our novel correlation mining method. First, we discuss how the event log is tailored into a transformer-compatible form. Then, we describe how the attention weights of the output model are normalized before diving into how the weights are interpreted for generating hypothetical correlations.

A. Event Log Transformation

This paper considers a standard historical log from SmartThings CLI [1]. Specifically, we consider an event as a logged state change containing a timestamp, capability, device name, and value. The log, by nature, is an asymmetrical time series which means that the events are irregularly spaced in the temporal domain. Moreover, we apply discretization to continuous values to maintain uniformity among the data log.

With regard to the chosen transformer framework, we draw a subtle parallel between an IoT event log and a Natural Language Corpus. We regard an event sequence akin to a sentence, where each event is analogous to a word, and the order in the log holds positional importance comparable to words in a sentence.

We formulate event sequences through a specialized algorithm that compares the time stamps of an anterior event to that of a posterior event and a time tolerance value. If the temporal difference between the two events is less than the time tolerance, then we add it to the sequence. If not, we end the previous sequence and add the event to a new sequence.

B. Correlation Discovery

Once we sequence the event log we pass it to the transformer for attention score analysis. By nature, a transformer can only compute attention in relation to a particular sequence. Therefore, it is not possible to discover a global attention score for each token pairing. Rather, we take all the event sequences from the training data and pass them through the model to generate attention matrices for each individual sequence.

For each unique token pairing, we take the maximum attention score as an overall indication of correlation for those two events. In other words, the attention scores provide an initial measure of correlation for basic one-to-one pairings. Inherently, BERT attention scores undergo processing through a softmax function. So, to enhance their interpretability, we apply a natural log function to the weights, and use a min-max scaler to standardize the data range between 0 and 1.

Using the definitions established in Sec. IV we say that two events are hypothetically correlated if their scaled attention score surpasses a threshold T. To incorporate dynamism into the method, we ascertain the threshold by computing the mean and subsequently positioning it at a distance of n standard deviation intervals from the calculated mean.

C. Correlation Validation

For all one-to-one correlations that pass the threshold, a special validation algorithm checks them against the data log itself. First, all unique events are counted throughout the data. Then, an occurrence count is found for each one-to-one correlation pairing by iterating through the log and checking for all instances both events are found within a certain time tolerance from each other. The time tolerance value should be the same as the tolerance mentioned in Sec. VI-A. In order to filter between valid and invalid correlations, we establish a conditional occurrence ratio using the formula $ratio = \frac{len(A \cap B)}{len(A)}$, where $len()$ indicates the count of the events. For all one-to-one correlations that pass a ratio threshold, we say they are valid correlation rules and save them for anomaly detection. As for the correlations that fail the threshold, they get passed on for two-to-one correlation formulation.

D. Two-to-one Correlation Formulation and Validation

The primary objective of two-to-one correlations is to establish more intricate relationships between devices possessing enhanced detection capabilities compared to the fundamental one-to-one correlations. As such, we formulate two-to-one correlations from hypothetical one-to-one correlations that fail the validation stage. Specifically, we formulate two-to-one correlations such that they can not be broken down into two valid one-to-one formulae. Therefore, for two one-to-one correlations $A \rightarrow B$ and $A \rightarrow C$, the first form we consider is $B \wedge C \leftrightarrow A$. In this situation, we notice two cases where a specific input event allocates high attention scores to two distinct events; however, individually, their occurrences in the data are too scarce to be considered significant rules. Thus we combine both posterior events together to formulate the first kind of two-to-one correlation.

Moreover, given two one-to-one correlations $A \rightarrow C$ and $B \rightarrow C$, we deem $A \wedge B \leftrightarrow C$ as a legitimate two-to-one correlation as well. In this scenario, we have two different events that both attend to the same event, yet as standalone one-to-one correlations, they fail the validation threshold. Thus we combine both events together and consider the combined two-to-one correlation rule instead.

To validate the hypothesized two-to-one correlations, we employ a specialized algorithm that searches throughout the training data for all instances of each correlation. Using the standard notation for a two-to-one correlation, $A \wedge B \leftrightarrow C$, the algorithm works by first finding all values for A, B, and C as well as their opposite state events we denote as A', B', and C'. We denote an event of the opposite state using the ' indicator. For example, given an event A, where A indicates a sensor is active, A' indicates the sensor is inactive. After all the events and their primes are gathered, we search through the training data for all instances of the event in the A position. At each instance of A, we first search backward for events B or B'. If B' is found, then we search 30 seconds after A for an instance of B. This accounts for a case where B doesn't happen before A but rather happens right after it instead. After the search for event B is complete, if B is found we log it and perform the same steps, but for event C. If B is not found, we simply move on to the next instance of A. Overall, the algorithm returns the ratio of how many times A, B, and C all happen together over how many times solely A, and B are found together. Using this ratio, we say a two-to-one correlation is validated if it achieves a ratio of greater than or equal to 95%. This means that at least 95% of the time that events A and B were found together, C is also found. By using a validation threshold of 95% we can severely reduce the likelihood of false positives as defined in Sec. VIII-B3.

VII. Experiment

A. Environment Setup

For data collection and experimentation we put together a smart home testing environment made of four distinct testbeds. Throughout the testbeds, a total of 20 IoT smart devices spanning nine different types were deployed throughout the testbeds as shown in Tab. I. Data was collected over a 24-day span and accessed using the SmartThings CLI. Moreover, the smart home possessed six full-time residents with varying degrees of technical expertise. All residents consented to the devices used throughout the testbeds. For privacy purposes, we excluded recording devices that collect sensitive data such as cameras and microphones.

B. Model Configuration and Training

As mentioned in Sec. V-A we use a RoBERTa transformer architecture to learn the attention weights of all the events in the smart home system. The model configuration we use has a maximum positional embedding size of 512, a hidden size of 768 units, 6 attention heads, and 6 hidden layers. Furthermore, we use an Adam optimizer with a learning rate of $1e-4$. Note, due to the smaller scale of our testing environment, a more complex configuration was not necessary to yield good results. For consistency purposes, we mask 15% of the tokens at random as is performed in [13]. The model is trained to predict the masked tokens and in doing so adjusts the attention weights accordingly as it learns the hidden relationships between the event tokens in the system.

TABLE I: IoT Devices, Capabilities, and Values

Code	Device Name	Capabilities	Values
B	SmartThings Button	Button	pressed, double pressed, held
CA	SmartThings Multipurpose Sensor	Contact, Acceleration	active, inactive, open, closed
SW	Third Reality Smart Switch	Switch	on, off
MS	SmartThings Motion Sensor	Motion	active, inactive
PS	SmartThings Arrival Sensor	Presence	present, not present
PO	SmartThings Power Outlet	Switch, PowerMeter	on, off
Z	Zooz 4-in-1 Sensor	Motion, Illuminance, Tamper, etc.	active, inactive, tampered, not tampered, etc.
SL	August Smart Lock	Lock, Contact	locked, closed, unlocked, open,
TV	Samsung 6 Series	Switch	on, off

For training, we set aside 18 days' worth of data, while the last 6 days collected are reserved for testing. Through experimentation, we discovered that a batch size of 8 yielded the best results for our system. Consequently, we train the model over 75 epochs.

C. Correlation Identification

The output of the trained model is a high-dimensional matrix of attention weights between all tokens in the model vocabulary. To develop hypothetical one-to-one correlations using the attention weights, we run all of the event sequences from the training data through a specialized algorithm that performs the following actions. First, it finds all permutations of token pairings in the sequence. Then, for each token pair, it averages the attention weights for that pair by layer per attention head. The output for each pair is a list of 6 attention weights: one for each attention head. Once all of the attention weights have been found for each pair of each sequence, we scale the scores to fit between 0 and 1 and determine the maximum score out of all of the attention heads for each pairing of each sequence. Using the maximum scores, we apply a threshold as described in Sec. VI-B to determine a list of hypothetical one-to-one correlations. Out of maximum scores for 20,304 token pairings, we determine the mean to be 0.379 and the standard deviation to be 0.200. Therefore, using two deviation steps we determine the threshold to be $0.779 \approx 0.78$.

As a result, after applying the threshold, we find 622 pairs that pass. Finally, we take the maximum score for each unique pairing out of all of the sequences tested and find 88 unique one-to-one hypothetical correlations that need to be validated, described in Sec. VI-C. Out of the 88 correlations validated,

37 passed the threshold qualifying them as legitimate one-to-one correlations. Moreover, the remaining 51 one-to-one correlations that failed the threshold passed onto two-to-one correlation generation.

Building on the foundation laid by the 51 one-to-one correlations that failed, we employ the techniques outlined in Section VI-D to formulate 171 unique hypothetical two-to-one correlations. Subsequently, we input the 171 correlations into the validation algorithm, as defined in Sec. VI-C, to compute the occurrence ratio for each correlation. In an effort to mitigate the likelihood of false positives, we confine our considerations to two-to-one correlations that have an occurrence ratio of 95% or higher. Taking everything into account, 57 of the two-to-one correlations surpassed this threshold. A selection of the correlations unearthed through this process can be found in Table II.

VIII. Performance Evaluation

A. Anomalous Activity Generation

To evaluate the detection proficiency of our method, we emulate four attack scenarios within our system: *event injection*, *event interception*, *fake commands*, and *command interception*. A detailed explanation of these attacks is present in our threat model in Sec. III-B. In order to evaluate the performance of our method we simulate the following attack cases:

1) Event Injection: In this scenario, we consider an attacker who attempts to fool the system by injecting a fake sensor event. To simulate an injection attack, we insert n number of sensor events into the data log such that the injected event follows an event reflecting the opposite state of that capability and device pairing. For example, for a case where we inject n instances of $E^{motion}_{MS2=active}$ we position the injection in the log such that each instance of $E^{motion}_{MS2=active}$ follows an instance of $E^{motion}_{MS2=inactive}$. This method of injection reflects a more likely case of an attacker attempting to fool the system by quickly reversing the state of a device in the system. Moreover, for consistency purposes, we select the injection event as the A event given a two-to-one correlation $A \wedge B \leftrightarrow C$.

2) Event Interception: For event interception, we simulate an attacker who wants to fool the system by blocking certain sensor events from happening. Given the two-to-one correlation format, $A \wedge B \leftrightarrow C$, we conduct an interception attack by removing n number of sensor events from the data log in the C position.

3) Fake command: Similar to the event injection attack in Sec. VIII-A1, we insert n number of commands into the data log. We define commands as events that solely pertain to actuator devices such as a smart lock or smart switch. In the scenario of this attack, an attacker will try and fool the smart home system by injecting fake commands into the log to enact some form of physical transformation in the environment (I.E. a smart door lock unlocking).

4) Command Interception: Given the same fundamental concept as an Event Interception attack, Command Interception attacks define an actuator-specific attack by which an attacker blocks a command from being sent to the system. For the purpose of simulating this method of attack, we eliminate the n number of actuator-specific events from the log. The ultimate goal of the attacker is to block a certain actuator device from changing its states (I.E. preventing a smart door lock from locking).

B. Performance Metrics

We measure the performance using standard recall and accuracy metrics defined in [16]. Further, we establish the following definitions to distinguish *True Positive*, *True Negative*, *False Positive*, and *False Negative* below. It is important to note that for any particular attack, a single event is cross-compared with all correlations that it is related to. For metrics

TABLE II: Subset of Validated Two-to-One Correlations

Index	Correlation	Ratio
$\mathcal{C}1$	$E^{contact}_{SL=closed} \wedge E^{switch}_{SW1=on} \leftrightarrow E^{motion}_{MS2=active}$	100.00%
$\mathcal{C}2$	$E^{contact}_{CA3=open} \wedge E^{switch}_{SW1=on} \leftrightarrow E^{motion}_{MS2=active}$	100.00%
$\mathcal{C}3$	$E^{contact}_{CA3=open} \wedge E^{switch}_{SW1=on} \leftrightarrow E^{lock}_{SL=locked}$	100.00%
$\mathcal{C}4$	$E^{contact}_{CA3=open} \wedge E^{contact}_{SL=closed} \leftrightarrow E^{motion}_{MS2=active}$	98.91%
$\mathcal{C}5$	$E^{contact}_{CA3=open} \wedge E^{lock}_{SL=locked} \leftrightarrow E^{motion}_{MS2=active}$	98.51%
$\mathcal{C}6$	$E^{contact}_{CA3=open} \wedge E^{switch}_{SW3=off} \leftrightarrow E^{motion}_{MS2=active}$	98.49%
$\mathcal{C}7$	$E^{motion}_{MS2=active} \wedge E^{acceleration}_{CA3=active} \leftrightarrow E^{contact}_{CA1=closed}$	97.73%
$\mathcal{C}8$	$E^{motion}_{MS2=active} \wedge E^{contact}_{CA3=active} \leftrightarrow E^{contact}_{CA1=closed}$	97.90%
$\mathcal{C}9$	$E^{motion}_{MS2=active} \wedge E^{contact}_{CA3=active} \leftrightarrow E^{lock}_{SL=locked}$	100.00%
$\mathcal{C}10$	$E^{contact}_{CA1=closed} \wedge E^{contact}_{CA1=open} \leftrightarrow E^{motion}_{MS1=active}$	96.28%
$\mathcal{C}11$	$E^{contact}_{CA1=closed} \wedge E^{contact}_{CA3=open} \leftrightarrow E^{motion}_{MS2=active}$	97.98%
$\mathcal{C}12$	$E^{acceleration}_{CA3=active} \wedge E^{contact}_{CA3=open} \leftrightarrow E^{contact}_{CA1=closed}$	97.05%
$\mathcal{C}13$	$E^{acceleration}_{CA3=active} \wedge E^{switch}_{SW1=off} \leftrightarrow E^{contact}_{CA1=closed}$	97.41%
$\mathcal{C}14$	$E^{acceleration}_{CA3=active} \wedge E^{motion}_{MS4=inactive} \leftrightarrow E^{contact}_{CA1=closed}$	97.35%
$\mathcal{C}15$	$E^{contact}_{CA3=closed} \wedge E^{motion}_{MS4=inactive} \leftrightarrow E^{contact}_{CA1=closed}$	96.22%
$\mathcal{C}16$	$E^{contact}_{CA3=closed} \wedge E^{contact}_{CA3=open} \leftrightarrow E^{lock}_{SL=locked}$	97.05%
$\mathcal{C}17$	$E^{motion}_{MS2=inactive} \wedge E^{acceleration}_{CA3=active} \leftrightarrow E^{contact}_{CA1=closed}$	95.86%
$\mathcal{C}18$	$E^{motion}_{MS2=inactive} \wedge E^{motion}_{MS2=active} \leftrightarrow E^{contact}_{CA1=closed}$	96.11%
$\mathcal{C}19$	$E^{contact}_{SL=open} \wedge E^{contact}_{CA3=open} \leftrightarrow E^{lock}_{SL=locked}$	100.00%
$\mathcal{C}20$	$E^{contact}_{SL=open} \wedge E^{motion}_{MS2=active} \leftrightarrow E^{lock}_{SL=locked}$	100.00%
$\mathcal{C}21$	$E^{contact}_{SL=open} \wedge E^{contact}_{CA1=closed} \leftrightarrow E^{lock}_{SL=locked}$	100.00%
$\mathcal{C}22$	$E^{contact}_{SL=open} \wedge E^{contact}_{CA3=open} \leftrightarrow E^{lock}_{SL=locked}$	100.00%
$\mathcal{C}23$	$E^{motion}_{MS1=inactive} \wedge E^{contact}_{CA3=open} \leftrightarrow E^{contact}_{CA1=closed}$	95.14%
$\mathcal{C}24$	$E^{switch}_{SW1=off} \wedge E^{switch}_{TV=on} \leftrightarrow E^{contact}_{CA1=closed}$	100.00%
$\mathcal{C}25$	$E^{lock}_{SL=locked} \wedge E^{switch}_{SW1=off} \leftrightarrow E^{contact}_{CA1=closed}$	95.40%
$\mathcal{C}26$	$E^{lock}_{SL=locked} \wedge E^{motion}_{MS4=inactive} \leftrightarrow E^{contact}_{CA1=closed}$	95.60%
$\mathcal{C}27$	$E^{motion}_{MS2=active} \wedge E^{contact}_{CA1=closed} \leftrightarrow E^{contact}_{CA3=open}$	95.68%
$\mathcal{C}28$	$E^{motion}_{MS2=active} \wedge E^{lock}_{SL=locked} \leftrightarrow E^{contact}_{CA3=open}$	95.58%
$\mathcal{C}29$	$E^{contact}_{CA3=open} \wedge E^{contact}_{CA1=closed} \leftrightarrow E^{acceleration}_{CA3=active}$	95.90%
$\mathcal{C}30$	$E^{contact}_{CA3=open} \wedge E^{contact}_{CA1=open} \leftrightarrow E^{acceleration}_{CA3=active}$	97.58%

purposes, we consider the set of related correlations to the target event and check for an occurrence of a correlation violation for any item in that set.

1) True Positive (TP): For a standard two-to-one correlation, $A \wedge B \leftrightarrow C$, we say that a True Positive reflects a positive correlation violation such that A and B are found but the related C is missing in any given event sequence for a modified data log. Here, a modified data log refers to data that has undergone alterations such as injections or interception.

2) True Negative (TN): Given a correlation of the form $A \wedge B \leftrightarrow C$, a True Negative defines a situation where the correlation is satisfied in the unmodified data. In this case, a C is found to exist in any particular sequence where an A and B event are also found.

3) False Positive (FP): Continuing with the correlation format $A \wedge B \leftrightarrow C$, a False Positive is a scenario where events A and B are found in the unmodified data log, but there is no corresponding C found in that sequence. We limit false positives by only considering correlations that pass a 95% occurrence ratio threshold, meaning a C must be found with a corresponding A and B in at least 95% of all sequences where A and B occur together.

4) False Negative (FN): A false negative defines a case where for a particular sequence of events A and B in a modified data log, a C is also present resulting in a situation where the correlation is satisfied when it should be violated.

C. Performance Results

Across 21 test scenarios, our method achieved a mean recall of 97.38% and mean accuracy of 96.59%, as detailed in Table III. Notably, of the 21 scenarios, 17 had a 100% recall rate and 15 exceeded or met an accuracy rate of 95%. In the following cases below, we elaborate on some situations that highlight the detection capabilities of our method. For detection, we consider the set of all correlations that are related to the modification event. Therefore, the more correlations an event is related to, the greater the detection capabilities are for an attack directed at that instance.

Scenario $\mathcal{S}1$. In this scenario, an attacker attempts to fool the system into believing the front door is closed when in reality it is still open. In the testbed, the hallway's entrance features a smart lock and contact sensor labeled as SL. To enter or exit the home, residents must unlock and open the door before closing it. Notably, when the door closes, two correlations, $\mathcal{C}1$ and $\mathcal{C}4$, capture this activity. When the event $E^{contact}_{SL=closed}$ occurs alongside either $E^{switch}_{SW1=on}$ or $E^{contact}_{CA3=open}$, the event $E^{motion}_{MS2=active}$ is expected to show up as well in this sequence. For an attacker to successfully fool the system he or she must insert the $E^{contact}_{SL=closed}$ such that it follows an $E^{contact}_{SL=open}$ event whilst also occurring in a sequence with either the $E^{switch}_{SW1=on}$ or $E^{contact}_{CA3=open}$ events with a corresponding $E^{motion}_{MS2=active}$. Given the recall of 100%, this indicates the likelihood of successfully injecting the event such that all the necessary conditions are satisfied to produce a false negative, are next to none.

Scenario $\mathcal{S}7$. In the following scenario, the attacker injects a fake $E^{contact}_{CA3=open}$ event into the system to divert the attention of the homeowner away from a particular area. The $E^{contact}_{CA3=open}$ is found in correlations $\mathcal{C}2$ through $\mathcal{C}6$. The detection of this attack attained a recall of 100% indicating there were no false negatives found during the simulation. For an attacker to successfully pull off an attack of this kind, they would be required to find a gap in the system such that none of the correlations are violated. For instance, given the schematic of the testbed, it is easy to see that motion sensor $MS2$ is deployed right at the entryway of contact sensor $CA3$. Moreover, the smart lock and corresponding contact sensor SL are located just opposite the $CA3$ doorway. For these given circumstances this location is highly correlated making it quite likely that with the injection of $E^{contact}_{CA3=open}$ there will be at least one violation in the set of related correlations which will notify the homeowner to the nefarious activity.

Scenario $\mathcal{S}12$. Given this scenario, the attacker tries to cover his or her tracks by eliminating the event $E^{motion}_{MS2=active}$. In doing so, the attacker is trying to keep their presence hidden from the homeowner. To the benefit of the homeowner, $E^{motion}_{MS2=active}$ is related to correlations $\mathcal{C}2$, $\mathcal{C}4$, $\mathcal{C}5$, and $\mathcal{C}6$. For this highly correlated event, the attacker would have to figure out a means to get into the home and through the hall doorway without setting off the events $E^{contact}_{CA3=open}$, $E^{switch}_{SW1=on}$, $E^{contact}_{SL=closed}$, $E^{lock}_{SL=locked}$, $E^{switch}_{SW3=off}$, which are all events that indicate activity in that region.

Scenario $\mathcal{S}16$. For the following scenario, an attacker attempts to fool the system into believing the front door is locked when in reality it is still unlocked. Given an injected command, $E^{lock}_{SL=locked}$, correlations $\mathcal{C}5$, $\mathcal{C}25$, $\mathcal{C}26$ are evaluated for possible violations. For this attack to be successful the attacker would have to inject the command right after a $E^{lock}_{SL=unlocked}$ command and align it in a sequence such that if $E^{contact}_{CA3=open}$ occurs the event $E^{motion}_{MS2=active}$ must also occur, or if either $E^{switch}_{SW1=off}$ or $E^{motion}_{MS4=inactive}$ occurs the event $E^{contact}_{CA1=closed}$ also occurs. It is important to note that events of this form are logged most often at the moment of occurrence. Thus the attack window to cause a false negative reading would again be very tiny.

Scenario $\mathcal{S}19$. In this scenario, the attacker aims to prevent the system from registering the lock command for the front door. If successful, the attacker could gain access to the home. To the attacker's detriment, $E^{lock}_{SL=locked}$ is related to correlations $\mathcal{C}3$, $\mathcal{C}9$, $\mathcal{C}16$, $\mathcal{C}19$, $\mathcal{C}20$, $\mathcal{C}21$, $\mathcal{C}22$ which makes it very complicated to intercept the $E^{lock}_{SL=locked}$ without violating one of the seven correlations. Due to the number of related correlations, no matter how many different positions $E^{lock}_{SL=locked}$ is removed from, one of the correlations is violated, thus resulting in 0 reported false negatives and a 100% recall.

D. Observations

In this section, we highlight intriguing behaviors observed during our experiments. Notably, the subsequent insights emphasize distinct discrepancies evident in Table III.

TABLE III: Attack Scenarios and Performance Results

Scenario	Attack	Modification	# Trials	Related Correlations	Recall	Accuracy
$\mathcal{S}1$	Event Injection Attack	injected fake $E^{contact}_{SL=closed}$	30	$\mathcal{C}1$, $\mathcal{C}4$	100%	100%
$\mathcal{S}2$		injected fake $E^{contact}_{CA3=open}$	30	$\mathcal{C}2$, $\mathcal{C}3$, $\mathcal{C}4$, $\mathcal{C}5$, $\mathcal{C}6$	100%	93.61%
$\mathcal{S}3$		injected fake $E^{contact}_{CA1=closed}$	30	$\mathcal{C}10$, $\mathcal{C}11$	100%	94.87%
$\mathcal{S}4$		injected fake $E^{motion}_{MS2=active}$	30	$\mathcal{C}7$, $\mathcal{C}8$, $\mathcal{C}9$	100%	94.44%
$\mathcal{S}5$		injected fake $E^{acceleration}_{CA3=active}$	30	$\mathcal{C}12$, $\mathcal{C}13$, $\mathcal{C}14$	93.33%	91.40%
$\mathcal{S}6$		injected fake $E^{contact}_{CA3=closed}$	30	$\mathcal{C}15$, $\mathcal{C}16$	100%	96.48%
$\mathcal{S}7$		injected fake $E^{motion}_{MS2=inactive}$	30	$\mathcal{C}17$, $\mathcal{C}18$	96.55%	94.25%
$\mathcal{S}8$		injected fake $E^{contact}_{SL=open}$	30	$\mathcal{C}19$, $\mathcal{C}20$, $\mathcal{C}21$	100%	100%
$\mathcal{S}9$		injected fake $E^{motion}_{MS1=inactive}$	30	$\mathcal{C}23$	61.90%	89.15%
$\mathcal{S}10$	Event Interception Attack	removed $E^{contact}_{CA3=open}$	30	$\mathcal{C}27$, $\mathcal{C}28$	100%	96.92%
$\mathcal{S}11$		removed $E^{contact}_{CA1=closed}$	30	$\mathcal{C}7$, $\mathcal{C}8$, $\mathcal{C}12$, $\mathcal{C}13$, $\mathcal{C}14$	100%	95.40%
$\mathcal{S}12$		removed $E^{motion}_{MS2=active}$	30	$\mathcal{C}2$, $\mathcal{C}4$, $\mathcal{C}5$, $\mathcal{C}6$	100%	97.49%
$\mathcal{S}13$		removed $E^{acceleration}_{CA3=active}$	30	$\mathcal{C}29$, $\mathcal{C}30$	100%	96.28%
$\mathcal{S}14$		removed $E^{motion}_{MS1=active}$	30	$\mathcal{C}10$	100%	97.32%
$\mathcal{S}15$	Fake Command Attack	injected fake $E^{switch}_{SW1=off}$	30	$\mathcal{C}24$	100%	100%
$\mathcal{S}16$		injected fake $E^{lock}_{SL=locked}$	30	$\mathcal{C}5$, $\mathcal{C}25$, $\mathcal{C}26$	93.10%	95.05%
$\mathcal{S}17$		injected fake $E^{lock}_{SL=locked}$	10	$\mathcal{C}5$, $\mathcal{C}25$, $\mathcal{C}26$	100%	95.68%
$\mathcal{S}18$		injected fake $E^{switch}_{SW1=off}$	10	$\mathcal{C}24$	100%	100%
$\mathcal{S}19$	Command Interception Attack	removed $E^{lock}_{SL=locked}$	30	$\mathcal{C}3$, $\mathcal{C}9$, $\mathcal{C}16$, $\mathcal{C}19$, $\mathcal{C}20$, $\mathcal{C}21$, $\mathcal{C}22$	100%	100%
$\mathcal{S}20$		removed $E^{lock}_{SL=locked}$	10	$\mathcal{C}3$, $\mathcal{C}9$, $\mathcal{C}16$, $\mathcal{C}19$, $\mathcal{C}20$, $\mathcal{C}21$, $\mathcal{C}22$	100%	100%
$\mathcal{S}21$		removed $E^{lock}_{SL=locked}$	2	$\mathcal{C}3$, $\mathcal{C}9$, $\mathcal{C}16$, $\mathcal{C}19$, $\mathcal{C}20$, $\mathcal{C}21$, $\mathcal{C}22$	100%	100%
Average		-	-	-	97.38%	96.59%

For scenario $\mathcal{S}9$, we observe a considerably low recall and accuracy when juxtaposed to the performance of the other simulated scenarios. The reason behind the less-than-ideal readings is related to the size of the correlation set considered in detecting the attack. Unfortunately, there was only one two-to-one correlation that passed the threshold that possessed the event $E^{motion}_{MS1=inactive}$ in the A or B position. Therefore, our model's ability to capture the anomalous behavior is severely limited by considering only a single rule which necessitates that $E^{contact}_{CA3=open}$ be present and $E^{contact}_{CA1=closed}$ to be missing in order to be flagged as an anomaly. Nevertheless, when $E^{motion}_{MS1=inactive}$ is injected near a common occurrence of both $E^{contact}_{CA3=open}$ and $E^{contact}_{CA1=closed}$, a false negative is produce and the injected event is wrongfully assumed to be legitimate.

For scenarios $\mathcal{S}16$ and $\mathcal{S}17$, the same fake command attack using $E^{lock}_{SL=locked}$ is performed three separate times for three different quantities of trials. For each scenario, as the number of trials increased, the metric values got slightly worse. The reasoning behind this trend is due to the varying position of insertion in the log. For scenario $\mathcal{S}17$, the command $E^{lock}_{SL=locked}$ was injected 10 times in 10 different positions. For all injections, the attack was successfully detected because one of the correlations in the related correlation set was violated. However, as the number of trials increased from 10 to 30, the locations of injections changed, thereby, increasing the possibility for a false negative. Hence, out of the 30 injection positions tested, 2 resulted in a false negative meaning none of the correlations in the related correlation set were violated. The situation indicates, that despite the reduced likelihood of a successful attack as the related correlation set increases, there is still a minute chance of success. Nonetheless, our method ensures that as the smart devices in the environment increase, so too does the quantity of correlations and the strength of detection of our system.

IX. Conclusion

The integration of Internet of Things (IoT) devices within smart homes exponentially escalates the attack surface for potential security breaches. As these devices operate through constant connectivity and data exchange, they inadvertently create entry points for malevolent entities. In this paper, we presented a novel correlation-based anomaly detection system for identifying nefarious activity in smart home systems. Moreover, we proposed an innovative two-to-one correlation mining schema that leverages the power of transformer attention weights for identifying related events in the smart environment. To prove the capabilities of our approach we simulated 21 attack scenarios ranging over 4 common forms of IoT attacks. Using 57 mined two-to-one correlations, our system attained a detection accuracy and recall of 96.59% and 97.38%, respectively. Our results show that two-to-one correlations are an effective means of detecting a range of attacks in a smart home environment. Furthermore, we demonstrate the potential of two-to-one correlations to fill the gaps left by their elementary one-to-one relatives.

REFERENCES

[1] [Online]. Available: https://developer.smartthings.com/

[2] "Alexa developer documentation welcome," https://developer.amazon.com/en-US/alexa/alexa-developer-documentation-welcome.

[3] "Google home." [Online]. Available: https://home.google.com/welcome/

[4] "Smartthings." [Online]. Available: https://www.smartthings.com/

[5] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "Sok: Security evaluation of home-based iot deployments," in *2019 IEEE symposium on security and privacy (sp)*. IEEE, 2019, pp. 1362–1380.

[6] S. Birnbach and S. Eberz, "Peeves: Physical event verification in smart homes," 2019.

[7] Z. B. Celik, P. McDaniel, and G. Tan, "Soteria: Automated iot safety and security analysis," in *USENIX Annual Technical Conference*, 2018.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[9] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 636–654.

[10] C. Fu, Q. Zeng, H. Chi, X. Du, and S. L. Valluru, "Iot phantom-delay attacks: Demystifying and exploiting iot timeout behaviors," in *52nd IEEE/IFIP International Conference on DSN*, 2022.

[11] C. Fu, Q. Zeng, and X. Du, "HAWatcher: Semantics-Aware anomaly detection for appified smart homes," in *USENIX Security 2021*.

[12] Y. J. Jia, Q. A. Chen, S. Wang, A. Rahmati, E. Fernandes, and A. Prakash, "Contexlot: Towards providing contextual integrity to appified iot platforms." 2017.

[13] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.

[14] B. Ur, E. McManus, M. Pak Yong Ho, and M. L. Littman, "Practical trigger-action programming in the smart home," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2014, pp. 803–812.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[16] Wikipedia, "Precision and recall — Wikipedia, the free encyclopedia," 2022. [Online]. Available: https://en.wikipedia.org/wiki/Precision_and_recall

[17] W. Zhou, Y. Jia, Y. Yao, L. Zhu, L. Guan, Y. Mao, P. Liu, and Y. Zhang, "Discovering and understanding the security hazards in the interactions between iot devices, mobile apps, and clouds on smart home platforms," 2019.