Anomaly-based Intrusion Detection of IoT Device Sensor Data using Provenance Graphs

Ebelechukwu Nwafor, Andre Campbell and Gedare Bloom
Department of Electrical Engineering and Computer Science
Howard University, Washington, DC 20059
Email: ebelechukwu.nwafor@bison.howard.edu

Abstract—The Internet of Things (IoT) has revolutionized the way humans interact with devices. Unfortunately, this technological revolution has been met with privacy and security challenges. One major concern is the notion of malicious intrusions. A common way of detecting a malicious intrusion is by treating an attack as an anomaly and using anomaly detection techniques to detect the source of intrusion. In a given IoT device, provenance, which denotes causality between system events, offers immense benefit for intrusion detection. Provenance provides a comprehensive history of activity performed on a system’s data, which indirectly ensures device trust. In this paper, we propose an approach to intrusion detection of IoT devices that leverages sensor data flow as seen through provenance graphs. In this approach, an observed provenance graph is compared to an application’s known provenance graph. This comparison involves the dimensionality reduction of a provenance graph to a vector space representation and similarity metric. We evaluated our approach on an IoT application which simulates a climate control system.

Index Terms—Anomaly detection, Intrusion detection, Internet of Things, Data Provenance

I. INTRODUCTION

IoT devices have become an essential part of our daily lives in commercial, industrial, and infrastructure systems. These devices offer benefits to consumers by interacting with the physical environment through sensors and actuators, which allows device automation thereby improving efficiency. Unfortunately, the proliferation of IoT devices has led to an increase in the number of remotely exploitable vulnerabilities leading to new attack vectors that could have disastrous financial and physical implications. In 2015, security researchers demonstrated a vulnerability on the Jeep vehicle which allowed remote control of the automotive system over the Internet [1]. In 2016, researchers discovered a vulnerability that allows Internet connected smart thermostats to be subject to remote ransomware attacks in which an attacker gains sole control of the thermostat until a fee is paid [2]. These are a few examples of some of the potential malicious vulnerabilities that could have devastating, long-lasting impact on an IoT system.

Intrusion detection [3] is the process of discovering malicious exploits in a system. One way of detecting an intrusion is by the use of anomaly detection techniques. An anomaly, also referred to as an outlier, is data that deviates from the normal system behavior. Anomalies could be indicative of a system fault or that a system has been compromised by a malicious event. Due to the sensitive nature of safety critical systems, detecting malicious attacks is of utmost importance.

We propose an approach to identifying anomalous sensor events using provenance graphs. This approach involves the use of a similarity metric to compare observed provenance graphs with provenance graphs derived from an application’s normal execution. The result is an anomaly score which is compared with a previously set threshold to classify an observed provenance graph as either anomalous or benign. We evaluate the effectiveness of our approach with a sample IoT application that simulates a climate control system.

II. GRAPH SIMILARITY ANOMALY DETECTION

A. Provenance Graphs

Provenance denotes the origin of an object and all other activities that occurred on that object. Data provenance, also known as data lineage, can be defined as the history of all activities performed on a data object from its creation to its current state. Provenance ensures data trust [4]. It establishes causality between entities contained in a data object through information flow tracking thereby it allows for the verification of a data source. Provenance is represented by a directed acyclic graph (DAG) in which vertices represent data entities and the edges correspond to the interaction between them.

Most provenance collection frameworks developed to track provenance use system level event sequences in an operating system [5]–[7]. For IoT devices, containing limited or no operating system functionality, it is essential to use a provenance collection framework that places less emphasis on an operating system and more emphasis on application level information flow tracking. For our provenance graph generation, we use PAIoT [8], a provenance collection framework which tracks the information flow of sensor-based events in an IoT device. In PAIoT, sensor data containing observation information is represented as a provenance graph. Sensor events are instrumented in the application source code. Each sensor data generated is defined as a tuple \((t, e, a, s_1, r_1)\) where \(t\) represents the time a sensor reading was generated, \(e\)
represents the sensor data, a represents activity performed on
sensor data, s1 represents a sensor identifier, and v1 represents
the device information. Tuple information is constructed as
a provenance graph and stored in a graph database (Neo4j)
where further processing and query analytics can be performed
on provenance data.

A provenance graph is a directed acyclic graph, \( p = (V, E) \)
where \( V \) is a set of vertices \( V = \{v_1, ..., v_n\} \) such that
\( v_i = (\text{type}, \text{value}) \) and \( E \) is a set of edges \( E = \{e_1, ..., e_n\} \)
where \( e_i = (v_x, v_y, \text{label}) \) and \( v_x, v_y \) are source and destination
vertices. Two vertices \( v_x, v_y \) are equal (denoted \( v_x = v_y \))
if \( v_x.\text{type} = v_y.\text{type} \) and \( v_x.\text{value} = v_y.\text{value} \). Two edges
\( e_x \) and \( e_y \) are equal (denoted \( e_x = e_y \)) if \( e_x.\text{value} = e_y.\text{value} \),
\( e_x.\text{label} = e_y.\text{label} \). We use the union operator \( \cup \) over edge sets in the usual way of the union
of sets.

Types may take on one of the following: Agent, Entity, and Activity. An agent is an actor that is represented by
an identifier (e.g., sensor or device name). An entity is an
event, which represents data that is produced as a result of
an activity. An activity is an action performed by an agent or
entity (e.g., read, update, create, delete). label takes on one of
the following values: \( \text{wasGeneratedBy}, \text{used}, \text{wasInformedBy}, \text{wasDerivedFrom}, \text{wasAttributedTo}, \text{wasAssociatedWith}, \text{actedOnBehalfOf} \).

**B. Graph Similarity**

Similarity is a measure of how identical two objects are,
for example, by measuring the angle between objects (using cosine similarity) or a linear distance (using euclidean
distance) between the objects. In this work, we use cosine similarity as our similarity metric. This was inspired by the use of information retrieval techniques for query ranking. e.g.,
given a corpus \( D = \{d_1, ..., d_n\} \), and query, \( q \), how do we
find document(s) \( \{d_x, ..., d_y\} \) which are similar to \( q \) and rank
them by order of importance. Cosine similarity is a measure of orientation between two non-zero vectors. It measures the
cosine of the angle between the vectors. Two vectors which are at an angle of 90° have a similarity of 0, two vectors
which are identical (with an angle of 0°) have a cosine of 1,
and two vectors which are completely opposite (with an angle of 180°) have a similarity of -1. Since we are concerned with the similarity between vectors, we are only concerned with the positive values bounded in [0,1]. The cosine similarity between two vectors, \( X \) and \( Y \), is computed by:

\[
\cos(\theta) = \frac{X \cdot Y}{\|X\| \cdot \|Y\|} = \frac{\sum_{i=1}^{n} X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2} \times \sqrt{\sum_{i=1}^{n} Y_i^2}}
\]

In order to apply cosine similarity between provenance graphs, we compute a vector representation which reduces
the graph into an \( n \)-dimensional vector space where \( n \) represents the total number of edges contained in the union
of all edge sets. Figure 2 illustrates the vector space conversion
of provenance graphs. \( p_1 \) and \( p_2 \) which consists of vertices \( A, B, E, F, G, I, J, S, R \) and edge labels
\( aOBO, wAW, wGB, wDB \). The vector space representation of \( u_1 \) is the occurrence of edges contained in the edge set
of graph \( p_1 \), which are also found in the collective union of
edge sets. Algorithm 1 further outlines the concept of graph
to vector conversion.

**C. Anomaly Detection on Provenance Graphs**

Anomaly detection involves the use of rule-based, statistical,
clustering or classification techniques to determine normal
or anomalous data instances. The process of determining all anomalous instances in a given dataset is a complex task.
A major challenge in anomaly detection is providing the
right feature set from the data to use for detection. Another
challenge exists in defining what constitutes as normal system behavior. Most anomaly detection using point-based data often fail to include the dependencies that exist between data points.

Due to the ubiquitous nature of IoT devices, there are a wide
array of vulnerabilities associated with them. In designing
our anomaly detection framework, we expect an attacker’s
footprint is reflected through the data flow as depicted in the
provenance graph. Our algorithm detects attacks such as false
data injection, and state change as depicted in information flow
of sensor events in provenance graphs.

**Algorithm 1: Graph to vector conversion.**

1: procedure \( \text{GRAPHTOVECTOR}(E, E_G) \)
2: \( n \leftarrow |E_G| \)
3: for \( i = 0 \) to \( n \) do
4: \( Q[k], Q[i] \leftarrow 0 \)
5: for \( e_j \in E \) do
6: \( \text{if} \ e_j = e_g \) then
7: \( Q[g] \leftarrow Q[g] + 1 \)
8: end if
9: end if
10: end for
11: return \( Q \)
12: end procedure
Algorithm 2: Detection algorithm given an observation phase graph set, \( P \), a detection phase graph, \( p \), and a threshold.

1: procedure GraphAnomaly\((P, p, \text{threshold})\)
2: \hspace{1em} INPUT: \( P = \{p_0, ..., p_n\} \mid p_i \leftarrow (V_i, E_i), 0 \leq i \leq n \).
3: \hspace{1em} \( E_G \leftarrow \bigcup_{i=0}^{n} E_i \).
4: \hspace{1em} \( Q \leftarrow \text{GraphToVector}(p, E_G) \).
5: \hspace{1em} \( Z \leftarrow \{\} \).
6: \hspace{1em} for \( p_i \in P \) do
7: \hspace{2em} \( N_i \leftarrow \text{GraphToVector}(p_i, E_G) \).
8: \hspace{2em} \( z \leftarrow \text{Cosine\_Similarity}(Q, N_i) \).
9: \hspace{2em} \( Z \leftarrow Z \cup z_i \).
10: \hspace{1em} end for
11: \( s_{\text{val}} \leftarrow \min(Z) \).
12: if \( s_{\text{val}} \geq \text{threshold} \) then
13: \hspace{1em} return normal
14: end if
15: return anomaly
16: end procedure

Many IoT devices implement a control systems in which sensor data is used as an input in a feedback loop to an actuator. The operations of most control systems are regular and predictable. For example, in a thermostat application, temperature readings generated might be converted from Celsius to Fahrenheit and utilized as feedback to an actuator. Each iteration of a control loop sequence generates a path in a provenance graph. This notion can be leveraged to define an expected provenance graph for each application.

The expected regularity of provenance graphs in IoT applications motivates a supervised learning approach to anomaly detection. This approach consists of two phases: observation phase, also known as the training phase, and the detection or test phase. In the observation phase, the system collects provenance data considered to be a representation of the normal system behavior. In the detection phase, the provenance graph set is compared with the provenance graph derived from subsequent observations to determine if an anomaly exists by measuring similarity between this graph and the provenance graph set. Note that provenance graphs from the observation and detection phase form a graph set. A global edge set, \( E_G \), represents the union of edge sets contained in a graph set. Algorithm 2 is the graph anomaly detection function given an observation phase graph set, \( P \), and a detection phase graph, \( p \).

The experiment evaluation serves as a preliminary study to confirm the correctness of our theoretical approach in detecting anomalous instances between provenance graphs. We evaluate our intrusion detection algorithm by implementing an IoT application which simulates a climate control system. Climate control systems ensure a proper functional environment for people and machinery. Constant irregularities in temperature could have devastating effects on industrial machinery. This system consists primarily of a heating ventilation and air conditioning (HVAC) system which uses temperature and humidity data to regulate environmental conditions within a building. We utilize a publicly available dataset [9] which...
consists of a year’s worth of longitudinal data on the thermal conditions, related behaviors, and comfort of twenty-four occupants in a medium-sized building generated at a period of fifteen minutes. We utilize the temperature and humidity data as input to our simulation. We generate provenance graphs for each week of the year. We compare the provenance graph generated in consecutive weeks to see how they differ using our graph similarity approach (i.e., week 1 compared to week 2, week 2 compared to week 3 etc.).

Figure 3 depicts the cosine similarity between provenance graphs generated from the first occupant by preceding weeks. Ensuring a proper threshold score is used for detection is an important task that requires extensive knowledge of the application domain. The threshold is manually set to a value which is defined by domain experts. For automatic anomaly threshold detection, one can use prediction methods to define an anomaly score. As an example, a threshold might be set to 0.15 in which all of the scores below the threshold are considered anomalous. Since the dataset does not contain attacks, the declines shown in Figure 3 would likely cause false positives.

IV. RELATED WORK

Liao et al. [10] characterize a system’s normal behavior by the frequency of unique system calls which are converted into a vector space representation. Stephanie et al. [11] define a human immune system inspired intrusion detection analyzing system call sequences. Yoon et al. [12] developed intrusion detection on embedded systems by analyzing system call frequencies clustered using k-means. Manzoor et al. [13] proposed a centroid based clustering anomaly detection for instances of streaming heterogeneous graphs in real time. Papadimitriou et al. [14] proposed five similarity algorithms for comparing the similarity of web graphs. Xie et al. [15] proposed a provenance-aware intrusion detection and analysis (PIDAS) system using provenance graphs generated from system calls which reveals interactions between files and processes. Our approach differs from prior work because we focus on anomaly detection through information flow sequences of sensor data as represented by provenance graphs.

V. SUMMARY AND CONCLUSION

In this paper, we propose an anomaly detection algorithm for detecting anomalous instances of sensor based events in an IoT device using provenance graphs. We evaluate our approach with a preliminary study on an IoT application which simulates a climate control system. Current implementation of our anomaly detection algorithm works with offline data. Future work would include implementation for real-time detection. We also plan on conducting further experimentation to identify the false and true positive rates of our algorithm using select IoT application dataset.

REFERENCES