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 [2]. 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 [3]. 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 [4] 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 [5]. 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 [6]–[8]. 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 [9], 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, s_1 represents a sensor identifier, and r_1 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.

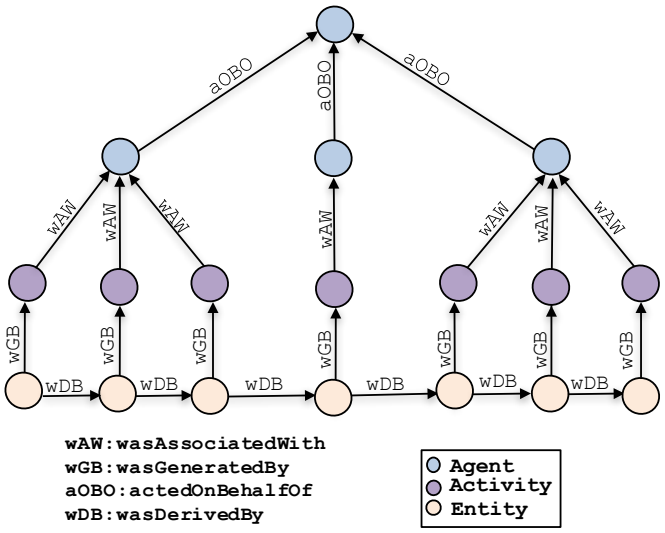

Fig. 1: Components of a provenance graph where nodes represent types (agent, activity, and entity) and edges represent relationships between types

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 $v_i = (type, value)$ and E is a set of edges $E = \{e_1, ..., e_n\}$ where $e_i = (v_s, v_d, label)$ and v_s, v_d are source and destination vertices. Two vertices v_x, v_y are equal (denoted $v_x = v_y$) if $v_x.type = v_y.type$ and $v_x.value = v_y.value$. Two edges e_x and e_y are equal (denoted $e_x = e_y$) if $e_x.v_s = e_y.v_s$, $e_x.v_d = e_y.v_d$, and $e_x.label = e_y.label$. 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: *wasGeneratedBy*, *used*, *wasInformedBy*, *wasDerivedFrom*, *wasAttributedTo*, *wasAssociatedWith*, *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:

$$\cos(\theta) = \frac{X \cdot Y}{\|\mathbf{X}\| \cdot \|\mathbf{Y}\|} = \frac{\sum_i^n X_i Y_i}{\sqrt{\sum_i^n X_i^2} \times \sqrt{\sum_i^n Y_i^2}}$$

Algorithm 1 describes how to calculate the cosine similarity of two graphs p_x, p_y given their vector representations u_x, u_y

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. $\boldsymbol{p_1}$, and $\boldsymbol{p_2}$ 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 4 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.

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

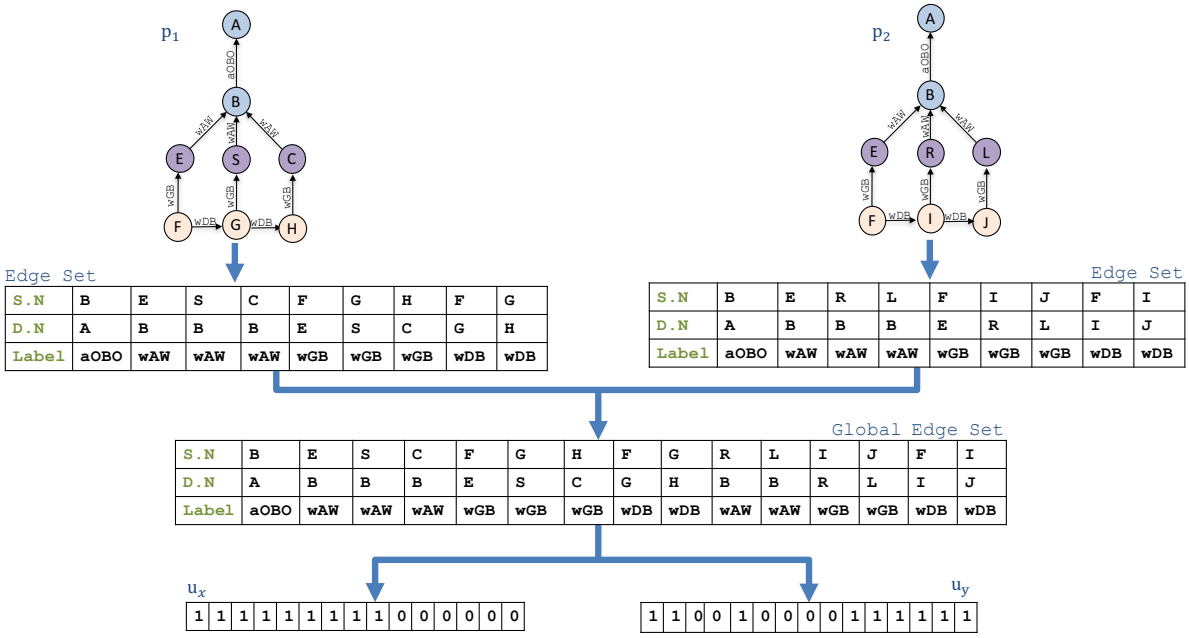

Fig. 2: Provenance graph conversion to vector space. u_x, u_y represents vectors generated from both provenance graphs and labels are an abstraction of $(type, value)$ tuple.

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 6 is the graph anomaly detection function given an observation phase graph set, P, and a detection phase graph, p. Z represents a list of the cosine scores from comparing each of the provenance graphs in the observation phase graph set with a detection phase graph. A function, $calculateAnomalyScore$ is used to determine the anomaly score, which is based on the minimum cosine similarity score of elements contained in Z.

Algorithm 1: Cosine similarity given two vectors.

procedure SIM(u_x, u_y)
  INPUT: $u_x = \{u_{x_0}, ..., u_{x_n}\}, u_y = \{u_{y_0}, ..., u_{y_n}\}$
  $S_p, S_{u_x}, S_{u_y} \leftarrow 0$
  for $0 \leq i \leq n$ **do**
    $S_p \leftarrow S_p + (u_x[i] \times u_y[i])$
    $S_{u_x} \leftarrow S_{u_x} + u_x[i]^2$
    $S_{u_y} \leftarrow S_{u_y} + u_y[i]^2$
  end for
  $G \leftarrow \dfrac{S_p}{\sqrt{S_{u_x}} \times \sqrt{S_{u_y}}}$
  return G
end procedure

Algorithm 3: Construction of global edge set from a set of provenance graphs.

procedure GRAPHSETTOGLOBALEDGESET(P)
  INPUT: $P = \{p_0, ..., p_n\} \mid p_i \leftarrow (V_i, E_i), 0 \leq i \leq n.$
  $E_G \leftarrow \cup_{i=0}^{n} E_i$
  return E_G
end procedure

Algorithm 4: Graph to vector conversion.

procedure GRAPHTOVECTOR(E, E_G)
  $n \leftarrow |E_G|$
  $Q[k], Q[i] \leftarrow 0, 0 \leq i < n$
  for $e_j \in E$ **do**
    for $e_g \in E_G \mid 0 \leq g < n$ **do**
      if $e_j = e_g$ **then**
        $Q[g] \leftarrow Q[g] + 1$
      end if
    end for
  end for
  return Q
end procedure

III. EXPERIMENT

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

Algorithm 5: Calculates an anomaly score given a set of cosine similarity values.

```
1:  procedure CALCULATEANOMALYSCORE(Z)
2:      INPUT: Z = {z_0, ..., z_n}, 0 ≤ i ≤ n.
3:      score ← 0.0
4:      for z_i ∈ Z do
5:          if score > z_i then
6:              score ← z_i
7:          end if
8:      end for
9:      return score
10: end procedure
```

Algorithm 6: Detection algorithm given an observation phase graph set, P, a detection phase graph, p, and a $threshold$.

```
1:  procedure GRAPHANOMALY(P, p, threshold)
2:      E_G ← GraphSetToGlobalEdgeSet(P ∪ p)
3:      Q ← GraphtoVector(p, E_G)
4:      Z ← {}
5:      for p_i ∈ P do
6:          N_i ← GraphtoVector(p_i, E_G)
7:          z ← SIM(Q, N_i)
8:          Z ← Z ∪ z_i
9:      end for
10:     s_val ← calculateAnomalyScore(Z)
11:     if s_val ≥ threshold then
12:         return normal
13:     end if
14:     return anomaly
15: end procedure
```

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 [10] 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

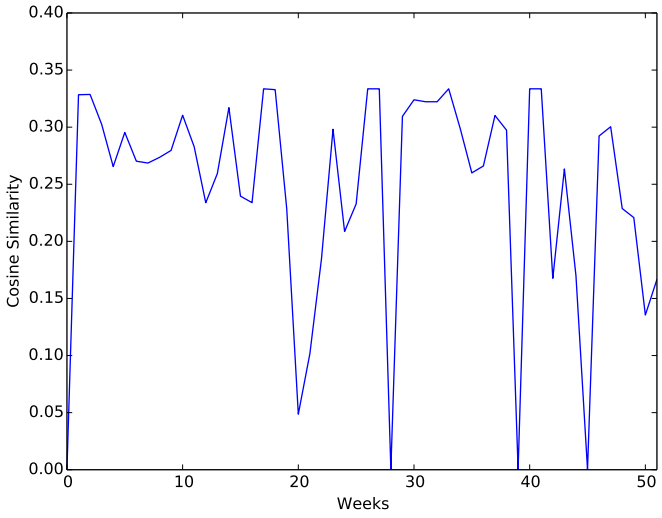

Fig. 3: Provenance graph comparison of climate control system by week.

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

There has been a considerable amount of research done on contextualized event-based anomaly detection of system call sequence. Liao et al [11] characterizes a system's normal behavior by denoting the frequency of unique system calls which are converted into a vector space representation. A classification algorithm such as k-Nearest Neighbors was used to classify the training data set. Stephanie et al. [12] defines a human immune system inspired intrusion detection systems analyzing system call sequences of an application. Yoon et al. [13] developed a technique for intrusion detection on embedded systems by analyzing system call frequencies clustered using k-means which groups legitimate system behavior. Observations at run-time that do not fit into a cluster are considered an anomaly.

Additionally, anomaly detection on graphs has also been explored. Manzoor et al. [14] proposed a centroid based clustering anomaly detection for instances of streaming heterogeneous graphs in real time. Papadimitriou et al [15] proposed five similarity algorithms for comparing the similarity of web graphs. Xie et al. [16] proposed a provenance-aware intrusion detection and analysis (PIDAS) system using provenance graphs generated from system calls which reveals various interactions between files and processes.

Our approach differs from prior work because we focus on anomaly detection through information flow sequence 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.

VI. Acknowledgment

This research has been supported in part by US National Science Foundation (CNS grant No. 1646317). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF.

References

[1] M. B. Barcena and C. Wueest, "Insecurity in the internet of things." [Online]. Available: https://www.symantec.com/content/dam/symantec/docs/white-papers/insecurity-in-the-internet-of-things-en.pdf

[2] A. Greenberg, "The jeep hackers are back to prove car hacking can get much worse," 2015. [Online]. Available: https://www.wired.com/2016/08/jeep-hackers-return-high-speed-steering-acceleration-hacks/

[3] D. Raywood, "Defcon: Thermostat control hacked to host ransomware," 2016. [Online]. Available: https://www.infosecurity-magazine.com/news/defcon-thermostat-control-hacked/

[4] A. Lazarevic, V. Kumar, and J. Srivastava, *Intrusion Detection: A Survey*. Boston, MA: Springer US, 2005, pp. 19–78. [Online]. Available: https://doi.org/10.1007/0-387-24230-9_2

[5] E. Bertino, *Data Trustworthiness—Approaches and Research Challenges*. Cham: Springer International Publishing, 2015, pp. 17–25. [Online]. Available: https://doi.org/10.1007/978-3-319-17016-9_2

[6] T. Pasquier, X. Han, M. Goldstein, T. Moyer, D. Eyers, M. Seltzer, and J. Bacon, "Practical whole-system provenance capture," in *Symposium on Cloud Computing (SoCC'17)*, ACM. ACM, 2017.

[7] R. H. Zakon, Ed., *28th Annual Computer Security Applications Conference, ACSAC 2012, Orlando, FL, USA, 3-7 December 2012*. ACM, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2420950

[8] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer, "Provenance-aware storage systems," in *Proceedings of the Annual Conference on USENIX '06 Annual Technical Conference*, ser. ATEC '06. Berkeley, CA, USA: USENIX Association, 2006, pp. 4–4. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267359.1267363

[9] E. Nwafor, D. Hill, A. Campbell, and G. Bloom, "Towards a provenance aware framework for internet of things devices," in *Proceedings of the 14th International Conference on Ubiquitous Intelligence and Computing*, ser. UIC '17. San Fransisco, CA, USA: IEEE Computer Society, 2017.

[10] J. Langevin, P. L. Gurian, and J. Wen, "Tracking the human-building interaction: A longitudinal field study of occupant behavior in air-conditioned offices," *Journal of Environmental Psychology*, vol. 42, no. Supplement C, pp. 94 – 115, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0272494415000225

[11] Y. Liao and V. R. Vemuri, "Using Text Categorization Techniques for Intrusion Detection," in *Proceedings of the 11th USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 2002, pp. 51–59. [Online]. Available: http://dl.acm.org/citation.cfm?id=647253.720290

[12] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *J. Comput. Secur.*, vol. 6, no. 3, pp. 151–180, Aug. 1998. [Online]. Available: http://dl.acm.org/citation.cfm?id=1298081.1298084

[13] M.-K. Yoon, S. Mohan, J. Choi, M. Christodorescu, and L. Sha, "Learning execution contexts from system call distribution for anomaly detection in smart embedded system," in *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, ser. IoTDI '17. New York, NY, USA: ACM, 2017, pp. 191–196. [Online]. Available: http://doi.acm.org/10.1145/3054977.3054999

[14] E. Manzoor, S. M. Milajerdi, and L. Akoglu, "Fast Memory-efficient Anomaly Detection in Streaming Heterogeneous Graphs," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 1035–1044. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939783

[15] P. Papadimitriou, A. Dasdan, and H. Garcia-Molina, "Web graph similarity for anomaly detection," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 19–30, May 2010. [Online]. Available: https://doi.org/10.1007/s13174-010-0003-x

[16] Y. Xie, D. Feng, Z. Tan, and J. Zhou, "Unifying intrusion detection and forensic analysis via provenance awareness," *Future Gener. Comput. Syst.*, vol. 61, no. C, pp. 26–36, Aug. 2016. [Online]. Available: http://dx.doi.org/10.1016/j.future.2016.02.005