IoT Device Anomaly Detection using Provenance Graphs

Ebelechukwu Nwafor
Howard University
Washington, DC 20059
ebelechukwu.nwafor@bison.howard.edu

Gedare Bloom
Howard University
Washington, DC 20059
gedare.bloom@howard.edu

ABSTRACT

The Internet of Things (IoT) has revolutionized the way we interact with devices. Unfortunately, this technological advancement 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 pinpoint the source of an intrusion. In a given IoT device, provenance graphs which denotes causality between system events offers immense benefit for intrusion detection. Provenance provides a comprehensive history of activities performed on a system which indirectly ensures device trust. Given a provenance graph, how can we determine if an anomalous activity has occurred? This paper seeks to address this issue. In this paper, we propose two lightweight approach to intrusion detection on IoT devices. The first is a whitelist approach which involves matching patterns of known normal provenance graph. The provenance graph exhibiting normal execution of IoT applications are stored in a knowledge repository. Incoming provenance graphs of similar applications are compared with provenance graph in the knowledge repository to determine if they exhibit anomalous behaviors. The second approach involves the use of a similarity metric to compare provenance graphs from the learning and detection phase. We evaluate the performance of our approach by comparing provenance graphs generated from sample IoT applications and determine the number of false positive, and false negative rates.

CCS CONCEPTS

•Computer systems organization →Embedded systems; •Security and privacy →*Anomaly detection systems;*

KEYWORDS

Internet of Things, Provenance Graphs, Anomaly Detection

ACM Reference format:
Ebelechukwu Nwafor and Gedare Bloom. 2018. IoT Device Anomaly Detection using Provenance Graphs. In *Proceedings of ACM IoTDI Conference, Orlando, Florida USA, July 2018 (IoTDI'18),* 6 pages.
DOI: 10.475/123_4

1 INTRODUCTION

IoT devices have become an essential part of our daily lives in commercial, industrial, and infrastructure systems. These devices offer immense benefits to consumers by interacting with our 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 [2]. In 2015, security researchers demonstrated a vulnerability on the Jeep vehicle which allowed remote control of the automotive system over the Internet [**?**]. 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 [**?**]. These are a few example of some of the potential malicious vulnerabilities that could have devastating long lasting impact on an IoT system.

Intrusion detection [**?**] 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 measure to compare observed provenance graphs with provenance graph derived from an application's normal execution. The result is an anomaly score which is compared with a previously set threshold to classify observed provenance graph as either anomalous or benign. We evaluate the effectiveness of our approach with a sample IoT application which simulates a climate control system.

2 BACKGROUND

2.1 Anomaly Detection

The notion of what constitutes an anomaly is often domain specific. Hawkins defines an anomaly as an "observation which deviates so much from the other observations as to arouse that it was generated by a different mechanism" [7]. In computing, an anomaly often indicates the presence of a malicious intrusion or a system fault. For example, an anomaly could be a sudden increase in web traffic of a web server which could be indicative of a denial of service attack. Additionally, in a critical care health device such as a pacemaker, an anomalous event could be detrimental to the health of a patient which could result in the loss of life.

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 or system is a complex task. A major

IoTDI'18, Orlando, Florida USA

DOI: 10.475/123_4

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.

2.2 Provenance Graphs

Provenance denotes the origin of an object and all activities that occurred on it. An example of provenance can be seen with a college transcript. A transcript is the provenance of a college degree because it outlines all of the courses satisfied in order to attain a degree. In computing, 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 [3]. 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 various data entities and the edges correspond to the interaction between them.

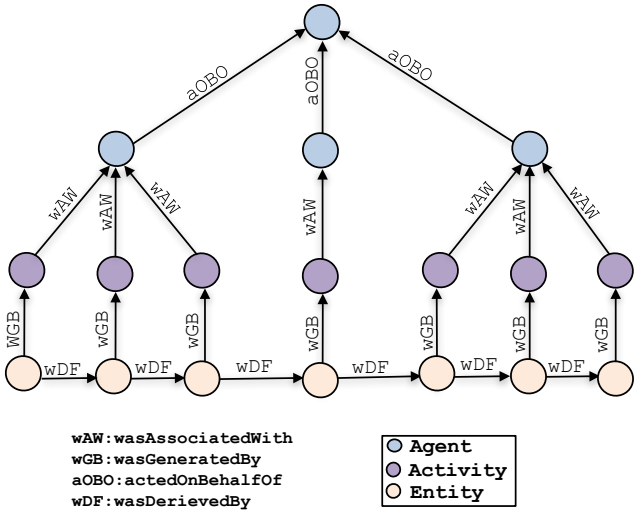

Figure 1: Components of a provenance graph where nodes represents types (agent, activity, and entity) and edges represents relationship between types

Most provenance collection frameworks developed to track provenance use system level event sequences in an operating system [14, 17, 21]. For most 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 [**?**], a provenance collection framework which tracks the information flow of sensor based events in an IoT device over its lifecycle. We formally define a provenance graph as follows:

Definition 2.1. 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 $v_i = (label, type, value)$. Two vertices v_x, v_y are similar (denoted $v_x \sim v_y$) if $v_x.type = v_y.type$ and $v_x.value = v_y.value$. 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). 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. *label* takes on one of the following values: *wasGeneratedBy*, *used*, *wasInformedBy*, *wasDerivedFrom*, *wasAttributedTo*, *wasAssociatedWith*, *actedOnBehalfOf*.

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. This notion can be leveraged to define an expected provenance graph for each application. Each iteration of a control loop sequence generates a path in a provenance graph. The nodes contained along the path denote historical event transformations. For example, in a thermostat application, temperature readings generated might be converted from Celsius to Fahrenheit and utilized as feedback to an actuator. We focus on networked connected control system. Network connection opens doors to potential vulnerabilities

The expected regularity of provenance graphs in IoT applications motivate a graph-based approach to anomaly detection. This approach consists of two phase: observation phase, also known as the training phase and the test or detection phase. In the observation phase, the system collects provenance data considered to be a representation of the normal system behavior. In the detection phase, provenance graph from the observation phase is compared with the provenance graph derived from subsequent observations to determine if an anomaly exists.

2.3 Similarity Metric

Similarity metric 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. 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 of the vectors, we are only concerned with the positive values bounded in [0,1]. To compute the cosine similarity between two vectors, X and Y, cosine similarity is:

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

Our method of comparing the similarity of provenance graphs was inspired by an information retrieval technique for document retrieval. 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. To achieve this, documents contained in

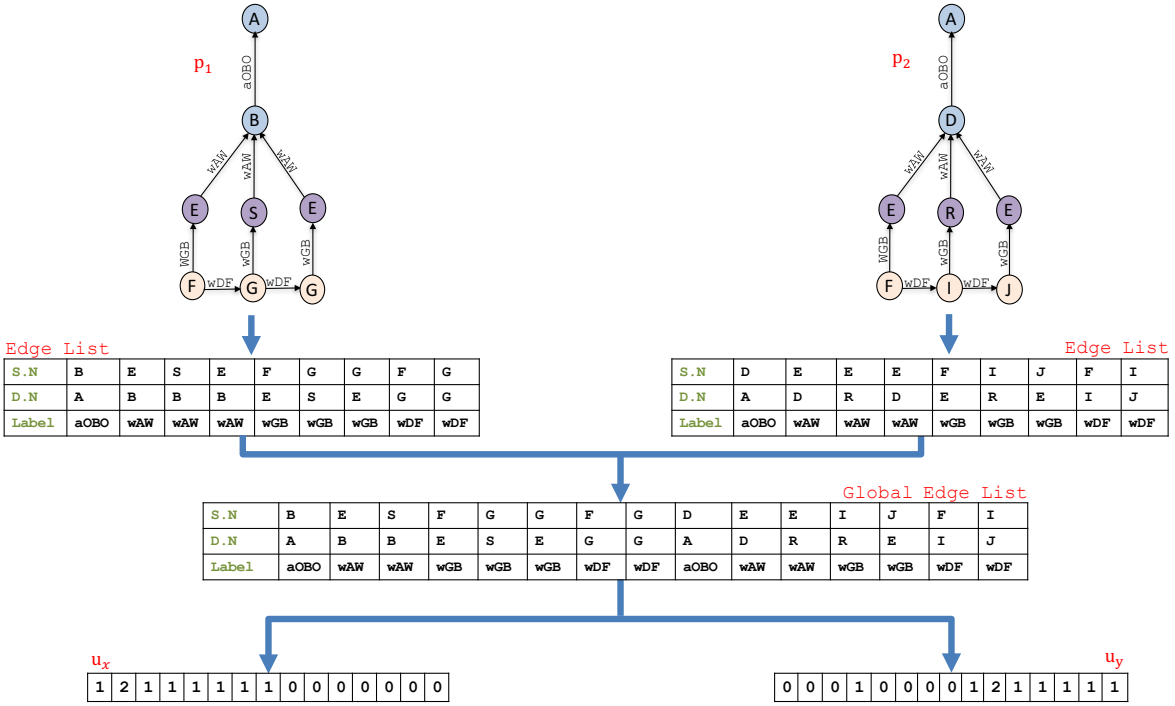

Figure 2: Provenance graph conversion to vector space. u_x, u_y represents the vectors generated from both provenance graphs

the corpus are converted into a vector space representation which allows documents to be ranked based on some similarity metric.

3 THREAT MODEL AND ASSUMPTIONS

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 the compromise to the integrity of data that exploits information flow of sensor events.

4 GRAPH SIMILARITY ANOMALY DETECTION

A similarity function measures how identical two provenance graphs are in a vector space. In order to apply a similarity function, we need to compute a vector representation. In our approach, a provenance graph undergoes dimensionality reduction into an n-dimensional vector space where n represents the number of unique edges (global set of edges) contained in the provenance graph set. This representation is used as input to our anomaly detection algorithm.

Selecting the most appropriate feature from provenance graphs is an important task. We need to utilize features that preserve the order of edges and nodes contained in the provenance graph. Our approach utilizes the occurrence of unique edges contained in both graphs.

We provide a means of classifying such edges as identical. To determine identical edges, we utilize a combination of edge labels and source and destination node. That is, two edges e_x and e_y are similar (denoted $e_x \sim e_y$) if $e_x.v_s \sim e_y.v_s$, $e_x.v_d \sim e_y.v_d$, and $e_x.label = e_y.label$

An edge list E_p consists of all of the edges contained in p. A global edge list E_G consists of all non-similar edges contained in P. E_G preserves the ordering of edges contained in P in the vector space.

Definition 4.1. Let $P = \{p_1, ..., p_n\}$ such that $p_i = (V, E)$. $E_G = \{e_1, ..., e_n\} \mid E_G \in P$ where $e_i \neq e_{i+1}, 1 \leq i \leq n$. $E_p \leftarrow \{e_1, ..., e_n\} \mid E_p \in p$ where $e_{pi} \neq e_{pi+1}, 1 \leq i \leq n$

Finally, we define our provenance graph to vector approach as follows:

Definition 4.2. Let $P = \{p_1, ..., p_n\}$ where $p_i = (V, E)$. The vector space representation of $p_i, \boldsymbol{u_i}$, is the number of times edges contained in E_p occurs in E_G. $\boldsymbol{u_i}$ has a length of k which consists of all of the unique edges contained in the global edge list E_G.

Figure **??** displays the vector space conversion of provenance graphs. $\boldsymbol{p_1}$, and $\boldsymbol{p_2}$ consists of vertices $A, B, E, F, G, I, J, K, M$ and edges $aOBO, wAW, wGB, wDF$. The vector representation of graphs, u_1, and u_2 is the occurrence of edges contained in the edge list that is found in the global edge list.

4.1 Graph Based Similarity Detection Algorithm

Given u_x, u_y which denotes the vector representation of provenance graphs p_x, p_y. The similarity of p_x, p_y is found by calculating the cosine similarity between the two vectors where 1 denotes

similarity between the two vectors and 0 denotes non-similarity between the two vectors. A threshold value is set which is used to classify the behavior of provenance graphs in the detection phase as normal or an anomaly.

$$sim(u_x, u_y) = \frac{u_x \cdot u_y}{\|\mathbf{u_x}\| \cdot \|\mathbf{u_y}\|} = \frac{\sum_{i=1}^{n} u_{xi} u_{yi}}{\sqrt{\sum_{i=1}^{n} u_{xi}^2}\sqrt{\sum_{i=1}^{n} u_{yi}^2}} \in [0, 1]$$

Algorithm 1 depicts a formal definition of the graph-based similarity algorithm.

```
1:  procedure GraphSetToEdgeList(P)
2:      INPUT: P = {p_0, ..., p_n} | p_i ← (V_i, E_i), 0 ≤ i < n.
3:      E_G ← {}
4:      for p_i = (V_i, E_i) ∈ P, 0 ≤ i < n do
5:          for e_j ∈ E_i do
6:              Found ← False
7:              for e_g ∈ E_G do
8:                  if e_j ~ e_g then
9:                      Found ← True
10:             if Found = False then
11:                 E_G ← E_G ∪ e_j
12:     return E_G
1:  procedure GraphtoVector(E, E_G)
2:      k = |E_G|
3:      Q[k] | Q[i] ← 0, 0 ≤ i < k
4:      for e_j ∈ E do
5:          for e_g ∈ E_G | 0 ≤ g < k do
6:              if e_j ~ e_g then
7:                  Q[g] ← Q[g] + 1
8:      return Q
1:  procedure GraphAnomaly(P, p)
2:      E_G ← GraphSetToEdgeList(P ∪ p)
3:      Q ← GraphtoVector(p, E_G)
4:      for p_i ∈ P do
5:          N_i ← GraphtoVector(p_i, E_G)
6:          z ← sim(Q, N_i)
7:          if z ≥ threshold then
8:              return normal
9:      return anomaly
```

Figure 3: Graph based anomaly detection algorithm

5 EXPERIMENT

This section outlines experimental procedures involved in evaluating our proposed algorithm.

5.1 Evaluation

The experiment evaluation serves as 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. Constant irregularities in temperature could have devastating effects on industrial machinery. Climate control systems ensures a proper functional environment for people and machinery. This system consist primarily of a Heating Ventilating and Air Conditioning (HVAC) System which uses temperature, humidity data to regulate environmental conditions within a building. The temperature is set within a bounded limit such that when the temperature exceeds a certain threshold, a cooling phase is activated and the air conditioning is switched on until the temperature decreases below a threshold in which the heating phase is activated in which the heater is turned on. We utilize a publicly available dataset [**?**] 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. This dataset consist of temperature, humidity, air velocity generated at a duration of fifteen minutes. We utilize the temperature and humidity data as input to our aforementioned simulation program. We generate provenance graphs for each week of the year. We compare the provenance graph generated in various weeks to see how they differ using our graph similarity approach (e.g week 1 compared to week 2, week 2 compared to week 3).

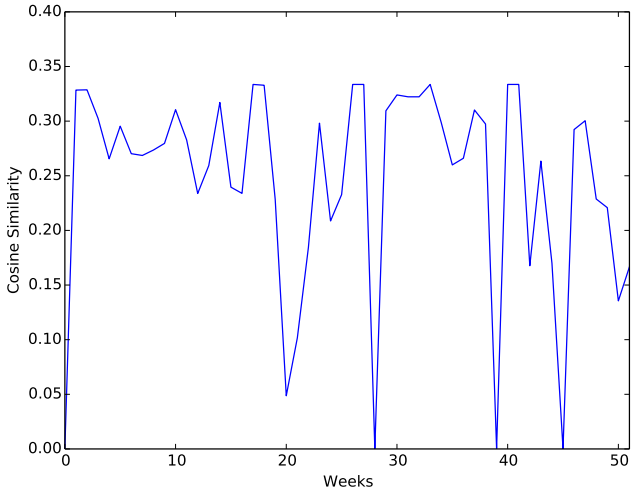

Figure 4: Provenance graph comparison of climate control system by week

Figure **??** depicts the cosine similarity between provenance graphs generated from the first occupant by preceding weeks. It is worth nothing that the rapid decline in the graph is as a result of anomalies between the provenance graphs. This might be as a result of rapid variation in temperature change for each week in the given dataset.

6 RELATED WORK

There has been a considerable amount of research done on anomaly detection most of which involves the analysis of system call sequence. Liao et al [12] 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 [9] defines a link between the human immune

system and intrusion detection systems. They developed a normal system behavior repository by analyzing system call sequences of an application. The application's system call sequence is stored in a normal database which is queried during observation. If an application executes a sequence of system calls that is not found in the normal database, a mismatch is recorded. If the mismatch for that application exceeds a threshold, an anomaly is detected. Yoon et al. [20] developed a technique for intrusion detection on embedded systems by analyzing system call frequencies. This is achieved by learning normal system profile of observed patterns in the system call frequency distribution.Their hypothesis is that applications follow a known frequency pattern which is centered around the centroid. Data from the training set is clustered using k-means which groups legitimate system behavior. Observation at run-time are compared with the clusters in the detection phase, if the incoming observation does not fit into a cluster, it is considered an anomaly. Additionally, anomaly detection on graphs has also been explored. Manzoor et al [13] proposed a centroid based clustering anomaly detection for instances of streaming heterogeneous graphs in real time. Papadimitriou et al [16] proposed five similarity algorithms for comparing the similarity of web graphs namely signature similarity, vertex/edge vector similarity, vertex ranking,and vertex edge overlap.

7 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 implemented our approach on an IoT application which simulates a climate control system. We plan on conducting detailed experiments to identify false positives, true positives, and false negative rates on other IoT applications.

8 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] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery* 29, 3 (01 May 2015), 626–688. DOI: http://dx.doi.org/10.1007/s10618-014-0365-y

[2] Mario Ballano Barcena and Candid Wueest. Insecurity in the Internet of Things. (????).

[3] Elisa Bertino. 2015. *Data Trustworthiness—Approaches and Research Challenges.* Springer International Publishing, Cham, 17–25. DOI: http://dx.doi.org/10.1007/978-3-319-17016-9_2

[4] Deepayan Chakrabarti. 2004. Autopart: Parameter-free graph partitioning and outlier detection. In *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 112–124.

[5] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly Detection: A Survey. *ACM Comput. Surv.* 41, 3 (July 2009), 15:1–15:58. DOI: http://dx.doi.org/10.1145/1541880.1541882

[6] Sushmito Ghosh and Douglas L Reilly. 1994. Credit card fraud detection with a neural-network. In *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*, Vol. 3. IEEE, 621–630.

[7] D. M. Hawkins. 1980. *Identification of outliers.* Chapman and Hall, London [u.a.]. http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+02435757X&sourceid=fbw_bibsonomy

[8] Victoria Hodge and Jim Austin. 2004. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review* 22, 2 (Oct. 2004), 85–126. DOI: http://dx.doi.org/10.1023/B:AIRE.0000045502.10941.a9

[9] Steven A. Hofmeyr, Stephanie Forrest, and Anil Somayaji. 1998. Intrusion Detection Using Sequences of System Calls. *J. Comput. Secur.* 6, 3 (Aug. 1998), 151–180. http://dl.acm.org/citation.cfm?id=1298081.1298084

[10] HP. Internet of things research study. (????). http://h20195.www2.hpe.com/V4/getpdf.aspx/4aa5-4759enn

[11] Terran Lane, Carla E Brodley, and others. 1997. Sequence matching and learning in anomaly detection for computer security. In *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*. 43–49.

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

[13] Emaad Manzoor, Sadegh M. Milajerdi, and Leman Akoglu. 2016. Fast Memory-efficient Anomaly Detection in Streaming Heterogeneous Graphs. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 1035–1044. DOI: http://dx.doi.org/10.1145/2939672.2939783

[14] Kiran-Kumar Muniswamy-Reddy, David A. Holland, Uri Braun, and Margo Seltzer. 2006. Provenance-aware Storage Systems. In *Proceedings of the Annual Conference on USENIX '06 Annual Technical Conference (ATEC '06)*. USENIX Association, Berkeley, CA, USA, 4–4. http://dl.acm.org/citation.cfm?id=1267359.1267363

[15] Caleb C. Noble and Diane J. Cook. 2003. Graph-based Anomaly Detection. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*. ACM, New York, NY, USA, 631–636. DOI: http://dx.doi.org/10.1145/956750.956831

[16] Panagiotis Papadimitriou, Ali Dasdan, and Hector Garcia-Molina. 2010. Web graph similarity for anomaly detection. *Journal of Internet Services and Applications* 1, 1 (01 May 2010), 19–30. DOI: http://dx.doi.org/10.1007/s13174-010-0003-x

[17] Thomas Pasquier, Xueyuan Han, Mark Goldstein, Thomas Moyer, David Eyers, Margo Seltzer, and Jean Bacon. 2017. Practical Whole-System Provenance Capture. In *Symposium on Cloud Computing (SoCC'17)*. ACM, ACM.

[18] Michal Valko, Gregory Cooper, Amy Seybert, Shyam Visweswaran, Melissa Saul, and Milos Hauskrecht. 2008. Conditional anomaly detection methods for patient–management alert systems. In *Proceedings of the... International Conference on Machine Learning. International Conference on Machine Learning*, Vol. 2008. NIH Public Access.

[19] Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. 1999. Detecting intrusions using system calls: Alternative data models. In *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*. IEEE, 133–145.

[20] Man-Ki Yoon, Sibin Mohan, Jaesik Choi, Mihai Christodorescu, and Lui Sha. 2017. 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 (IoTDI '17)*. ACM, New York, NY, USA, 191–196. DOI: http://dx.doi.org/10.1145/3054977.3054999

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

[22] Y. Zhang, N. Meratnia, and P. Havinga. 2010. Outlier Detection Techniques for Wireless Sensor Networks: A Survey. *IEEE Communications Surveys Tutorials* 12, 2 (2010), 159–170. DOI: http://dx.doi.org/10.1109/SURV.2010.021510.00088