Detecting DoS Attack in Smart Home IoT Devices Using a Graph-Based Approach

Ramesh Paudel
Department of Computer Science
Tennessee Technological University
Cookeville, TN
rpaudel42@students.tntech.edu

Timothy Muncy
Department of Computing
East Tennessee State University
Johnson City, TN
muncyt@etsu.edu

William Eberle
Department of Computer Science
Tennessee Technological University
Cookeville, TN
weberle@tntech.edu

***Abstract*—The use of the Internet of Things (IoT) devices has surged in recent years. However, due to the lack of substantial security, IoT devices are vulnerable to cyber-attacks like Denial-of-Service (DoS) attacks. Most of the current security solutions are either computationally expensive or unscalable as they require known attack signatures or full packet inspection. In this paper, we introduce a novel Graph-based Outlier Detection in Internet of Things (GODIT) approach that (i) represents smart home IoT traffic as a real-time graph stream, (ii) efficiently processes graph data, and (iii) detects DoS attack in real-time. The experimental results on real-world data collected from IoT-equipped smart home show that GODIT is more effective than the traditional machine learning approaches, and is able to outperform current graph-stream anomaly detection approaches.**

***Index Terms*—IoT Security, Smart Home, DoS Attack Detection, Graph Mining, Anomaly Detection**

I. Introduction

Specific-use computing devices are becoming further integrated into the daily life of many individuals around the world. These Internet of Things (IoT) devices are designed to optimize, automate, and improve quality of life in ways that general-use computing devices such as laptops, desktops, and modern smartphones typically do not. As devices such as Amazon Echo, Alexa, Philips Hue lightbulbs, and many appliances that are actively connected to the internet continue to gain in popularity, we must begin to address the security risks associated with them.

Most modern users of IoT devices are not aware of the security vulnerabilities of these devices, and how that may affect the network on which they reside. This is largely influenced by the fact that current software on a large number of IoT devices has little to no significant security beyond a password that many users often never change, leaving them vulnerable to attacks [1]–[7]. Most existing security solutions for IoT devices generate heavy computation and communication loads. Whereas, IoT devices like cheap sensors, with light-weight security protections, are vulnerable to attacks [8]. This lack of substantial security is made evident by a number of high profile hacks in recent years including the late 2016 Mirai malware DDoS attacks [9] which utilized compromised IoT devices as part of a botnet to orchestrate large-scale attacks, and the 2017 casino fish tank IoT thermometer hacking [10] which allowed hackers to pull sensitive customer data through the device to the cloud. Many devices are not designed to receive updates and yet many others will send the information for their updates over plain-text making it impossible for the majority of devices to remain secure through their entire life-cycle as new vulnerabilities and attack vectors are discovered [3]. With the plethora of new specific-use computing devices reaching the market every year, it is becoming increasingly urgent to have pragmatic solutions to these security concerns. A common solution being discussed in the literature is implementing a network-level security system. However, it is often not enough to simply have a passive network-level monitor or intrusion detection system as many times it will not be until after an attack has occurred and had a measurable impact that one will notice a new device has connected or launched an attack. This is why it is important to have an active network-level monitoring system that will react to anomalous behavior in real-time, thereby protecting the whole network of devices.

One possible approach for detecting anomalous behavior in *connected*, IoT devices, is through *graphs*. A graph provides a natural abstraction for representing such highly relational and inter-dependent data where each device in the network can be treated as a node and the communication between devices as edges. In addition, the anomalies in such a connected domain could exhibit themselves as relational. For example, a distributed denial of service (DDoS)attack in a smart home IoT network is perpetrated by a set of attack machines sending a high amount of traffic to a victim machine. Each entity (a device in a smart home IoT network) might appear to exhibit a normal behavior, yet while analyzing their interaction with other individual entities we can discover anomalies like a DDoS attack. Thus, a graph provides a robust tool for anomaly detection in such relational data domains. Another advantage of utilizing a graph-based approach is that we do not need any extra information like traffic packet size, protocol, etc., because the graph-based approach can only use the source IP and destination IP to build a graph of traffic flow. This

This work is supported by the US National Science Foundation under grant number 1852126. The statements made herein are solely the responsibility of the authors.

is particularly important in an encrypted network where full packet inspection is not possible and approaches that rely on features such as protocol and packet size can not be used.

In this paper, we propose to represent a real-time smart home IoT traffic as a *graph stream*. A normative pattern or behavior at any time in the IoT network would represent the expected traffic flow in an IoT network, while deviations from the expected traffic flow (like a DoS attack) would constitute an anomaly. To address the problem of DoS attack detection in smart home IoT devices, we introduce a novel graph-based anomaly detection approach called Graph-based Outlier Detection in Internet of Things (GODIT) that (i) represents smart home IoT traffic as a real-time graph stream, (ii) efficiently processes graph data, and (iii) detects DoS attacks in real-time. GODIT first performs a fixed-length random walk in a graph to construct *n-shingles* from a walking path. It then sketches a graph G into a $d-$dimensional sketch vector S_d using a $d-$discriminative *n-shingle*. The use of a shingle-based approach to representing the graph helps to capture both their local graph structure and an edge order [11]. Finally, the sketch vector S_d can be processed by any standard machine learning algorithm for detecting the DoS attack. The contributions of this paper can be summarized as follows:

- We propose a framework to represent smart home IoT traffic into a real-time graph stream.
- We propose a shingling-based graph sketching technique to represent higher dimensional graphs data into a memory-efficient fixed-sized vector.
- Real-time DoS attack detection using a sketch vector as an input.
- Experimentation on real-world smart home IoT traffic achieves better precision and recall than the current approaches, which demonstrates the effectiveness of GODIT on detecting anomalies in graph streams.

In the following sections, we will present the related work, a detailed framework for representing a smart home network as a graph, and a proposed methodology for graph sketching. We then discuss the dataset, present our experimental results and analysis, and conclude with some future plans.

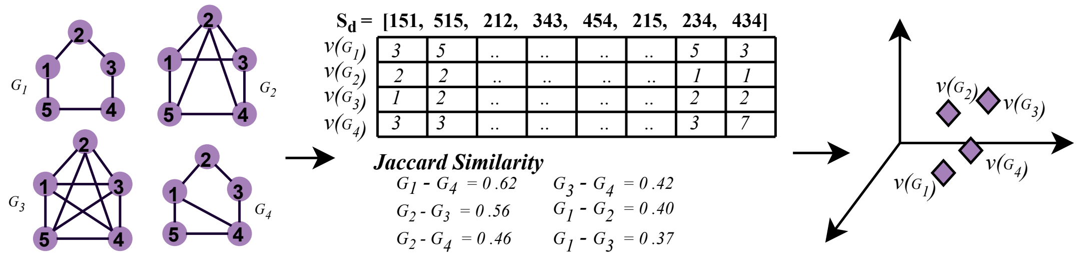
II. Related Work

In recent years, security for IoT devices has garnered a lot of attention given the discovery of more advanced botnets which rely on the weak security of IoT devices. In 2017 Zarpelao et al. [12] outlined many current approaches to intrusion detection for networked IoT devices. Their study categorized intrusion detection systems for IoT based on placement, detection, attack type, security threat, and validation strategy, using 18 papers in the relevant literature as reference. As stated in their paper, at this time it is not evident which methods for detection or detection system placement strategies are best suited for IoT systems. This is why it is important to continue to test the strengths and weaknesses of different approaches.

In the literature, some have taken the approach of more preventive measures. One example includes IoT Sentinel, which attempts to mitigate traffic to vulnerable devices as they are added to the network, but does not have much in the way of stopping a missed threat already in the process of attacking [13]. Other preventive measures include the use of blockchain algorithms for smart contracts [14], [15]. However, very little research has addressed reactive security solutions for the IoT [1]. Unfortunately, older solutions such as Zeek (formerly Bro), and Snort, which may implement network-level security solutions, do not work as well due to relying on a Signature-Based Intrusion Detection system. As such, it should be noted that signature-based approaches are not going to be substantial enough to address all of the relatively new and ever-increasing security issues associated with IoT devices as attack signatures simply can not be developed for the increasing number of these devices [1].

Some studies, while not directly exploring a reactive solution, are testing whether machine learning could be implemented to detect these attacks, which is the first real step towards a truly reactive solution. Doshi et al. [5] used a machine learning approach to categorize attack traffic with a test accuracy over 0.99 using a supervised approach like k-nearest neighbors and random forest. They created a dataset of a simulated attack and normal data in an experimental network environment that comprised of many consumer IoT devices. Canedo et al. [16] used Artifical Neural Network in a network's gateway to identify anomalous traffic and network behavior. In their study, they created a simulated IoT test-bed connecting 10 different Arduino Unos to a network, each with their own wifi chip and a temperature sensor that would send temperature data to the gateway approximately every 2 seconds. Hamza et al. [1] proposed a clustering approach for anomaly detection in a test-bed of 28 different devices. Sivanathan et al. [17] used random forest on the data collected from the testbed of 28 IoT devices used by Hamza et al. [1] to find anomalous traffic patterns for IoT devices. They were able to identify anomalous behaviors with over 99 percent accuracy using these traffic activity pattern profiles. Most machine learning approaches need full packet access to generate features such as protocol, packet size, activity cycles, port numbers, signaling patterns, and cipher suites, as well other detailed information about the traffic which may not always be available.

Graph-based approaches have been used for intrusion and anomaly detection in network traffic, however, they are less common. The typical structure of a graph-based approach is to define the computer network as a graph and feed the network traffic data into an anomaly detection algorithm to distinguish normal traffic from anomalous or attack traffic. Pahl et al. [18] introduced a distributed graph-based approach to monitor the communication of IoT devices that automatically create a communication model of IoT devices and classify communication traffic as normal or anomalous based on the communication model. The anomaly detection runs locally on each node. Ding et al. [19] proposed an intrusion detection approach in a network graph to identify malicious nodes. They define intrusion in a network graph as an antisocial behavior where a node is entering a community to which one does



S_d =	[151,	515,	212,	343,	454,	215,	234,	434]
$v(G_1)$	3	5	..	..	..	..	5	3
$v(G_2)$	2	2	..	..	..	..	1	1
$v(G_3)$	1	2	..	..	..	..	2	2
$v(G_4)$	3	3	..	..	..	..	3	7

Fig. 1: An overview of our proposed approach. Our input is a collection of graphs. We obtain a fixed-length random walk (l) and generate a shingle ($n = 3$). Discriminative shingles S_d are obtained using entropy-based measures. Sketch vector hold cost of S_d in each graph G_t. The similar graph share higher proximity (shown by Jaccard similarity)

not belong. Such antisocial behavior is identified using graph-cut. Parveen et al. [20] used an ensemble of the Graph-based Anomaly Detection (GBAD) tool [21] on real-time network data to detect insider threats. However, while this approach has some promise, it can become computationally intensive, and does not directly address DoS attacks. A single instance of the GBAD tool has been used for DoS attack detection in a network graph by [22]. However, their approach is limited to a static setting. StreamSpot [11] used a graph-based approach for anomaly detection in a streaming graph by representing graphs into a sketch-vector using shingles generated from an Ordered k-hop Breadth First Traversal. The anomaly detection is done by clustering the graph using a $K-$medoid algorithm on the sketch vector. Another good example of a graph-based anomaly detection method in a streaming graph is Spotlight [23]. SpotLight extracts a $K-$dimensional sketch $v(G)$ for every G, such that graphs containing the sudden (dis)appearance of large dense subgraphs are 'far' from 'normal' graphs in the sketch space. It then exploits the distance gap in the sketch space to detect graphs yielding anomalous sketches as anomalous graphs. As both SpotLight and StreamSpot use a sketching technique which is similar to our approach, both are used as comparison approaches to test the effectiveness GODIT in detecting DoS attacks in Smart Home.

III. Problem Definition

In this section, we introduce our streaming graph model, notations, framework, and formally define the problem.

A. Problem Setting

Each network traffic generated inside a smart home network is parsed and converted into a graph. The source IP and destination IP are considered nodes. Traffic flow between source IP and destination IP is represented as an edge between the nodes. Traffic within a fixed duration (usually 1 minute) are represented by a single graph. Let $G_s = \{G_i, G_{i+1}, ..., G_t, G_{t+1}, ...\}$ be a graph stream where each G_i denotes a graph at each one minute interval. We consider a graph $G_i = (v, e, f)$ as a generic undirected heterogeneous graph, i.e., $\exists v \in A \ni (x, y) \in A \land x \neq y$ and $\forall e \in (x, y)$ there is an edge going from vertex x to vertex y and (x, y) is an unordered pair. Whenever a new graph G_i arrives in the stream, a biased-random walk of a fixed length l is performed from each node in G_i extracting a walk path $p_{G_i} = \{v_1, v_2, ..., v_l\}$. The *n-shingles* are then constructed from a walk path p_{G_i}.

Definition 1: A *shingle* is a contiguous sub-sequence contained in a walk path p_{G_i}.

Definition 2: The *n-shingle* $s(v, n)$ is a sub-sequence of size n constructed from a biased-random walk path p_{G_i}.

From the given set of *n-shingles* in a graph stream, the top d *n-shingles* (*n-shingles* with the highest entropy values) called discriminative shingles are determined. The detailed entropy calculation process is explained in Section IV. Finally, a $d-$dimensional sketch vector (S_d) is generated for each graph G_i by enumerating the walk-cost defined as the sum of edge weight times the frequency for each discriminative shingle in graph G_i.

Definition 3: Sketching is a data representation technique where the higher dimensional object M can be projected into a lower dimensional objects, i.e., $M : R^N \rightarrow R^d$ that preserves properties of the original object with high probability.

Thus, our goal is to embed a graph into a $d-$dimensional sketch S_d. The use of shingle vectors to represent the graph will help to captures both their local graph structure and edge order [11]. Also, we want to ensure the graph proximity is well-preserved in such a $d-$dimensional space. The idea is that if two graphs share common structure, they share a higher number of *n-shingles*, therefore their *sketch* S_d is close.

Definition 4: Given two graphs $G_1 = (v_1, e_1)$ and $G_2 = (v_2, e_2)$, the **graph proximity** between G_1 and G_2 is larger if the number of shared discriminative shingles is larger.

Intuitively, graph proximity measures how many nodes, edges, and paths are shared by two graphs [24]. The graph proximity can be calculated using Jaccard similarity between the set of *sketch* (S_d). Note that using sketches of n-shingles (if $n > 2$) for measuring graph proximity is not just the Jaccard similarity of nodes or edges in the two graphs, as it also takes the connections among the common nodes into account. For illustration of the graph proximity, let us consider graphs G_1,

G_2, G_3, and G_4 each having nodes $\{1, 2, 3, 4, 5\}$ as shown in Figure 1. Since, G_1 and G_4 share more common structure (G_4 only have one extra edge) than with G_2 and $G3$, we can assume G_1 and G_4 share close proximity then with G_2 and G_3. As shown in Figure 1, the Jaccard similarity of G_1 is higher with G_4 than with G_2 and G_3. Also, G_2 and G_3 are closer to each other than with G_1 or G_4.

IV. Methodology

Our proposed methodology for DoS attack detection in smart home IoT traffic consists of a sketch vector representation for graphs constructed from IoT traffic using a top d discriminative *n-shingles*. The sketch vector obtained is then used as an input for anomaly detection. The methodology consists of three key steps: (i) shingling using a biased random walk, (ii) generating a fixed-sized $d-$dimensional graph sketch using discriminative shingles, and (iii) anomaly detection. We will now discuss the proposed methodology in detail.

A. Shingling using Biased Random Walk:

Similar to a $k-$gram in a text document to construct vector representation [25], a graph is decomposed into a set of $n-$shingles to construct the sketch-vector. The similarity between the two graphs can then be defined on the similarity between their $n-$shingle-based sketch vectors. $n-$shingles are created from the walk path in the graph. Random walks have been used as a similarity measure for a variety of problems in graphs [24], [26]–[29]. The random walk can also measure the similarity between graphs by measuring the distance between two nodes in the graphs, the number of shared paths, the number of shared neighbors between nodes in the graphs, etc. Hence, the random walks can effectively capture the contexts in which graphs have high similarity.

For each node v_i in graph G_i, the random walk of length l starting at node v_i is simulated in such a way that each i^{th} node in the walk path (c_i) are generated by the following distribution:

$$P(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \varepsilon e \\ 0 & \text{otherwise} \end{cases}$$

where π_{vx} is the unnormalized transition probability between node v and node x, and Z is the normalizing constant. The unnormalized transition probability can be calculated using edge weights. The normalizing factor Z in this case can be the total weight of all the edges between node v and x. We simulate a short random walk from each node instead of a single longer random walk to ensure that the walk traverses all the nodes in case of a disconnected graph.

Neighborhood sampling using a breadth first search (BFS) provide the structural information (like if the node is a bridge or a hub) about the local neighborhood. On the other hand, a depth first search (DFS) can explore the global view of the graph and give information beyond the 1-hop neighborhood. Using node2vec [26], with a random walk we can smoothly interpolate between BFS and DFS and gather information about both local and global information of the node. The

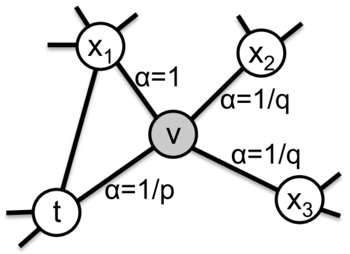


Fig. 2: Pictorial representation of node2vec random walk from t to v and is now evaluating its next step out of node v where edge labels indicate search biases α [26].

Node2vec random walk can interpolate between BFS and DFS using parameters p and q. As shown in Fig. 2, if a random walk just traversed the edge (t, v) and is currently at node v, the next step to the node x leading from the current node v on the walk can be decided using transitional probability $\pi_{vx} = \alpha_{pq}(t, x).w_{vx}$, where w_{vx} is the weight of the edge (v, x) and

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ 0 & \text{if } d_{tx} = 2 \end{cases}$$

where d_{tx} is the shortest distance between nodes t and x. The parameter p controls the likelihood of revisiting an older node in the walk while parameter q allows differentiating between "inward" and "outward" nodes. The high value of q provides a walk with a local view (equivalent to BFS behavior) while a low value of q provides a walk away from the current node (equivalent to DFS behavior). Thus, we can get structural information efficiently (in both time and space) from a graph using a biased random walk [26]. To capture the local view which can be important for DoS attack detection (to identify the attack-victim device group), we interpolate our random walk as a BFS by setting q higher than 1. Finally, we generate sets of *n-shingles* for each graph from the random walk path.

B. Graph Sketching using Discriminative Shingle:

Sketching techniques, analogous to *k-grams* in text mining, have been used as key features for representing graphs [11], [23], [30], [31]. A graph G_i can be sketched using a set of *n-shingles* present in the graph. However, when a newer graph G_{i+1} arrives in a stream, if the new *n-shingle* appears in G_{i+1}, then the number of dimensions for a sketch will increase. In a streaming scenario, the sketch vector will increase to the exponential number of dimensions and is infeasible to compute or store in the long run. Hence, we propose to compose a fixed-size sketch containing the top d discriminative *n-shingle*. The discriminative nature of the shingle is determined by the entropy value of the shingle in the training window. The higher the entropy value the higher the discriminative nature of the shingle.

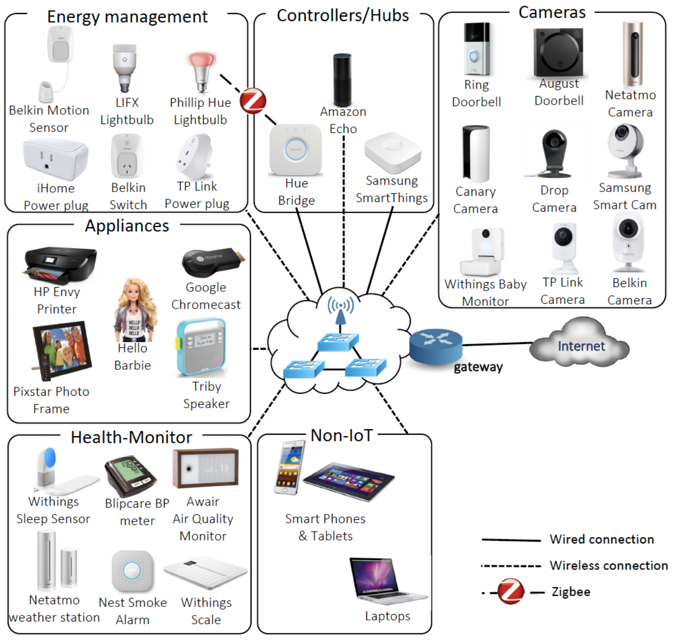


Fig. 3: Data collection testbed architecture showing 28 different [1], [17]

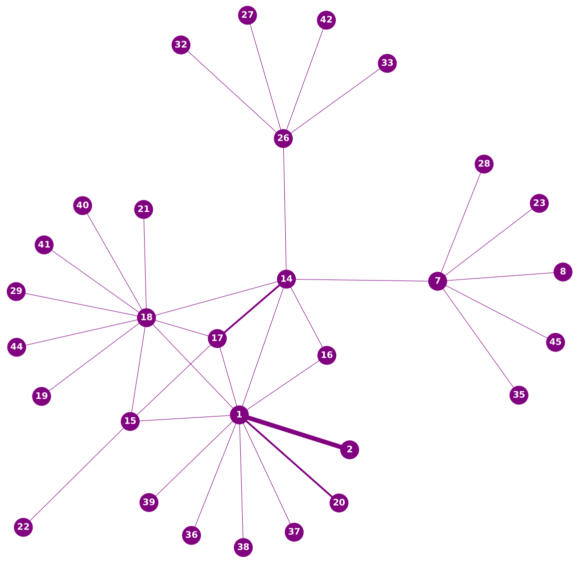


Fig. 4: Graph layout of smart home network. Node represents IoT devices and edge represents the traffic flow.

The entropy of a shingle is the measure of its structural information content. We will be using Shannon's entropy formula [32] to calculate the entropy of the shingle in the training window. Our definition of entropy is based on the distribution of the shingle over different graphs in the training window. Let W denote a training window containing graphs, $\{G_i, G_{i+1}, ..., G_{i+|w|-1}\}$, and S_w $=\{S_j, S_{j+1}, ..., S_N\}$ denote a set of shingles discovered from all of the graphs in the training window W. The probability of each shingle S_j in the graph G_i can be estimated as follows:

$$P(S_j|G_i) = \frac{r_{S_j}^{G_i}}{\sum_{i=1}^{|G|} r_{S_j}^{G_i}} \tag{1}$$

where $r_{S_j}^{G_i}$ is the number of occurrences of shingle S_j in graph G_i and $|G|$ is the total number of graphs where shingle S_j appears. Entropy of the shingle S_j in the training window W can now be defined as:

$$E(S_j) = -P(G_i|S_j)\sum_{i=1}^{N} P(S_j|G_i)log_2 P(S_j|G_i) \tag{2}$$

where N is the total number of graphs S_j present in and $P(G_i|S_j)$ is the fraction of graphs S_j is present in. Using Eq. 2, we can get the measure of the distribution of shingles in the training window. The entropy depends on both the frequency of shingles in individual graphs as well as the number of graphs it is present in. The discriminative shingles are the top d shingles with the highest entropy value meaning they are the most common shingles and are present in more graphs. Finally, the sketch vector S_d for a graph G_t will hold the cost of each discriminative shingle in graph G_i. The cost of a shingle S_j in graph G_i is defined as:

$$S_j^{cost} = wr_{S_j}^{G_i} \tag{3}$$

where w is the sum of edge weight in the shingle S_j and $r_{S_j}^{G_i}$ is the frequency of S_j in G_i. As stated earlier, the duplicate communication between the source and destination IP during a unit time interval (in a single graph) are merged into a single edge with the edge weight holding the count. Using the edge weight to calculate the cost of the shingle will help to reflect the amount of traffic flowing through the network. Fig. 1 shows a visual illustration of graph sketching using a discriminative shingle on a graph stream.

C. Anomaly Detection

The sketch vector S_d obtained using the above methodology is used as an input to an anomaly detection algorithm. The anomaly detection is performed by using the state-of-the-art Robust Random Cut Forests (RRCF) [33]. RRCF algorithm is an unsupervised ensemble method for detecting outliers in streaming data. We initialize RRCF using 100 trees and 256 samples. RRCF initializes the forest of 100 trees by using the sketch vector of the first 256 graphs. For the detail description of RRCF, the reader can refer to [33].

V. Data-sets

The smart home IoT traffic dataset used in this study is collected from The University of New South Wales Sydney (UNSW Sydney) smart home test-bed [1], [17]. A real-life smart home environment was created with 28 different IoT devices that include cameras, switches and triggers, hubs, air quality sensors, electronics, healthcare devices, and light bulbs.

Fig. 3 shows the testbed architecture of a smart home that shows all 28 IoT devices as well as non-IoT devices like laptops, mobile phones, and an Android tablet. The traffic traces were collected from this infrastructure for a period of 6 months, a subset of which will be used in our experimentation. During this period, several IoT devices inside the network would be at some points involved in attacks within the network. Some of these attacks include ARP Spoofing, Ping of Death, and Smurf Attacks. Except for the ARP Spoofing, every attack launched was some type of DoS attack. The data primarily comes in the form of large packet capture files. For more detail on the process of data collection and data, the reader can refer to [1], [17].

A. Data Preprocessing

The packet capture files are parsed using a python script to create a graph. The data consists of traffic flow between devices in the smart home network. Each unique communication is represented as an edge between source IP and destination IP. If there are multiple communications between the same set of source and destination IP, we sum them and assign them as an edge weight. The edge weight will be used to calculate the transition probability of the next node in the random walk path. The higher the edge weight the higher the probability to go to the next node in a random walk. Similarly, the edge weight will be used to calculate the cost of the shingle in the graph. This is particularly important in identifying the DoS attack where the attack device will be sending a high amount of traffic to the victim machine and it can be reflected as an edge weight. We obtained individual graphs by aggregating communications occurring every minute. Each DoS attack lasts for 10 minutes and is launched with the maximum limit of 1, 10 or 100 packets per second. Based on this ground truth, if an individual graph contains at least 50 edges belonging to a DoS attack, the graph was labeled anomalous. The graph layout in Fig. 4 shows the communication during 1 minute intervals. The thick edge reflects edges with high edge weights.

VI. Experimentation

To demonstrate a proof of concept that the proposed sketching approach can be used effectively to classify normal and DoS attack graphs, we first experimented in a static setting using a smaller dataset. The goal of the static setting is to quantify the effectiveness of the proposed approach before using it in a real-time streaming scenario. We then test the scalability and robustness of our approach in streaming scenarios. Table I shows a summary of the datasets used for our experimentation. All experiments are conducted using a MacOS with 2.6 GHz Intel Core i5 with 8 GB 1600 MHz DDR3 memory. The code is implemented in Python and will be released for research purposes after publication [...TBD...].

A. Static Setting

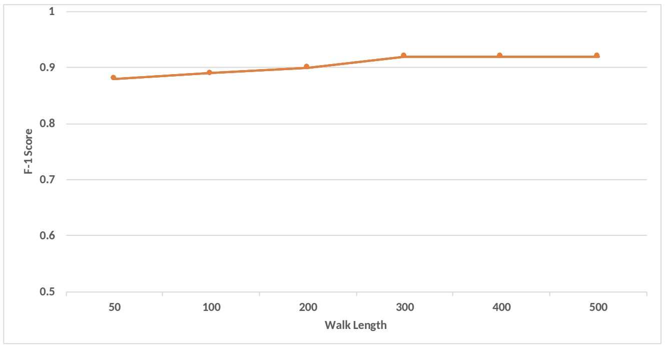
We chose to work with one full day of data (June 01, 2018) where 5 IoT devices are involved in DoS attacks. These devices include a TP-Link Plug, a Netatmo Camera, a Samsung Camera, a WEMO Power Switch, and a WEMO Motion Sensor. Within the date chosen, there are 4,726,992 network packets in total with 231,853 being associated with attacks launched from 1 of the 5 IoT devices mentioned above. There are a total of 44 instances of DoS attacks during this date, spread over 306 minutes, resulting in 306 anomalous graphs and 930 normal graphs. For a static setting, we consider all 1236 graphs in one single window to generate a sketch vector. We then use the sketch vector as input to the following supervised machine learning algorithms: decision tree, support vector machine, gradient boosting, and random forest, for classifying normal and DoS attack graphs.

TABLE I: Smart Home IoT Dataset

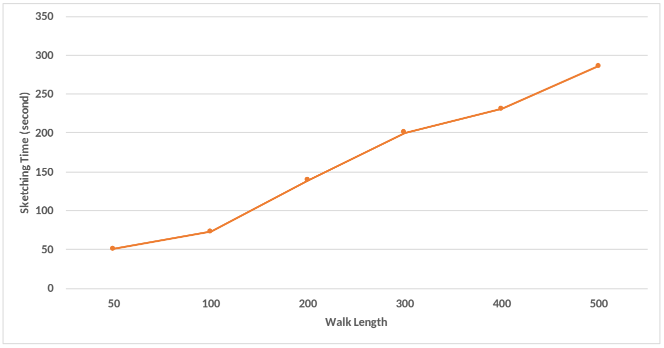
Dataset	# of Graph	# of Anomalies	# of Edges	Total Duration
Static Setting	1,236	306	4,726,992	1 day
Streaming Setting	9,678	1,291	29,959,737	1 week

1) Parameter Sensitivity: In this section, we discuss the parameter sensitivity of the proposed approach in static settings. Using the result of a decision tree, we evaluate how the F-1 score for DoS attack detection changes when we change the three parameters: i) length of the random walk (l), ii) length of the shingles (n), and iii) dimension of the embedding (d). The choice of a decision tree is driven by the fact that the decision tree has the best performance classifying DoS attack and normal graphs (as shown in Fig. 6). Any standard machine learning algorithm can be used for testing parameter sensitivity and should yield similar results. As shown in Fig. 5a, the F-1 score is increasing slightly every time we increase the length of a random walk (l) from each node in the graph. However, it comes with the cost of a higher sketching time as shown in Fig. 5b. Similarly, from Fig. 5c, we can conclude that the F-1 score is increasing rapidly until the sketch dimension (d) is 2048 (i.e. $log_2(2048) = 11$) and stabilizes after that. When d is low, increasing d will likely "illuminate" new parts of the graph and help to detect previously undetected anomalies. However, if d is increased beyond 2048, most of the graph is already "illuminated" and increasing d gives little or no added benefit [23]. The last parameter in consideration is the length of a shingle (n). The highest F-1 score is achieved when $n = 3$ and decreases while we increase the length of the shingle. Based on the above experimentation, we will use $d = 2048$, $l = 300$, and $n = 3$ as our parameters in further experimentation.

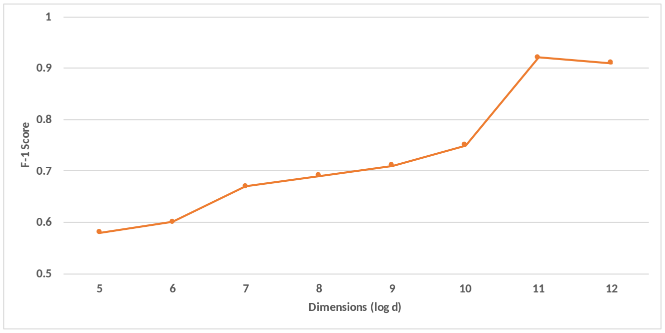
2) Baseline Approach: We compare the performance in a static setting with the standard machine learning approach developed by **Doshi et al. [5]**. This method primarily relies on the use of stateless features like packet size, inter-packet time interval, and protocol. Similarly, their approach uses stateful features to capture how the network evolves over time by aggregating statistics over multiple packets in a 10 second time window. Since there is an inherent overhead in generating stateful features, we tested the performance of the proposed approach against the stateless feature approach. They use common machine learning algorithms including K-nearest



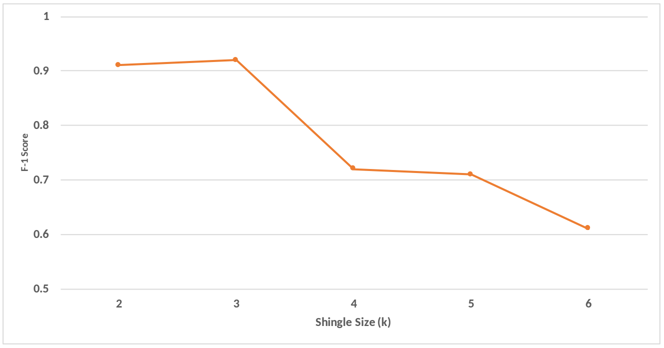
(a)



(b)



(c)



(d)

Fig. 5: Testing parameter sensitivity of the proposed approach using decision tree on sketch vector in static setting. a) Change in F-1 score for different set of walk length (l), (b) Sketching time using different length of a random walk (l), (c) Change in F-1 score for various dimensions of sketch vector (d), and (d) Change in F-1 score for various length of shingle (k)

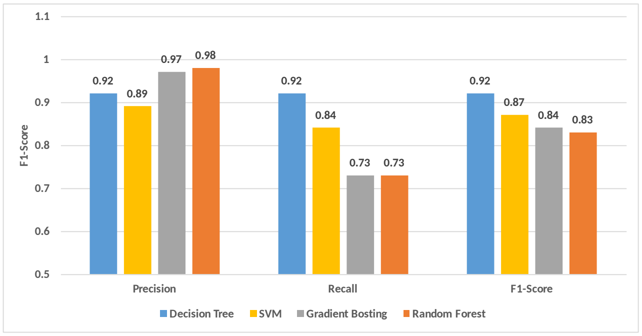


Fig. 6: Result of the proposed approach in static setting using supervised approach (parameter used $l = 300$, $d = 2048$, $k = 3$)

neighbors, linear support vector machine, decision tree, and Random Forest for detecting a DoS attack in a smart home scenario. We use the result of a decision tree, support vector machine, gradient boosting, and random forest to compare with the results of our approach.

3) Results: The sketch vector generated using a static setting with parameters $d = 2048$, $l = 300$, and $n = 3$ is given as an input to the machine learning approach. We

[1]SVM failed to classify DoS attack and normal graphs

TABLE II: Performances Comparison of Static Approach With Doshi et al.

Algorithm	Precision	Recall	F1-Score
Our Approach			
Decision Tree	0.92	**0.92**	**0.92**
Support Vector Machine	0.89	0.84	0.87
Gradient Boosting	0.97	0.73	0.84
Random Forest	**0.98**	0.73	0.83
Doshi et al.			
Decision Tree	0.92	0.88	0.90
Support Vector Machine [1]	-	-	-
Gradient Boosting	0.92	0.88	0.90
Random Forest	0.92	0.82	0.87

compare the performance of each classification algorithm in terms of precision, recall, and F-1 score. The results after running the decision tree, support vector machine (SVM), random forest, and gradient boosting are summarized in Fig. 6. The algorithms are implemented using a python based scikit-learn tool [34].

We see that random forest and gradient boosting have the highest precision, but, due to their low recall, the F-1 score is lower than both decision tree and SVM. Since the decision tree and SVM have superior recall and good precision (92% and 89% respectively), it makes them the best performing algorithm in terms of F-1 score. As shown in Table II, the result of a decision tree on the sketch vector generated by the proposed graph-based approach outperforms other standard machine learning approaches suggested by Doshi et al. [5]. The advantage of utilizing a graph-based approach is that we do not need any extra information like packet size, protocol, etc., because the graph only uses the communications between source and destination devices.

This is important particularly in situations where a network may be encrypted and features such as protocol, packet size, etc., are not available. These results in a static setting demonstrate that the proposed sketching approach can be used effectively to classify normal and DoS attack graphs. We further test the effectiveness of the proposed approach in the real-time streaming setting for unsupervised DoS attack detection.

B. Streaming Setting

In real-world scenarios, the smart home IoT traffic data arrive in a streaming fashion. The real-time anomaly detection approach in smart home should be able to handle the streaming nature of data. Furthermore, it is hard to obtain the class label in real-time streaming data. So the algorithm should be able to detect anomalies without the class label. Therefore to test the robustness of GODIT, we tested its performance in a real-time streaming setting.

For our experimentation in a streaming setting, one week of data spanning across October 1-7, 2018 is used. It has a total of 29,959,737 traffic packets equivalent to the same number of edges. Each graph represents 1-minute of network traffic. Hence, there are a total of 9,879 graphs each coming every minute in the graph stream. The first 25% of the data is used as a training set to generate discriminative shingles. Using the top $d-$ discriminative shingles generated from the training set, the subsequent 75% of the graphs (test set) are sketched into a $d-$dimensional sketch vector S_d. The sketch vector is then given as input to the anomaly detection algorithm. The anomaly detection is performed by using the state-of-the-art Robust Random Cut Forests (RRCF) [33] on the test set. RRCF algorithm is an ensemble method for detecting outliers in streaming data. We initialize RRCF using 100 trees and 256 samples.

1) Baseline Approach: We compare the performance of GODIT in a graph stream with two other sketching-based approaches. **(a) SpotLight** [23] is a randomized sketching-based approach that guarantees that an anomalous graph is mapped 'far' away from 'normal' instances in the sketch space with high probability for an appropriate choice of parameters. SpotLight compose a sketch containing total edge weights of K specific directed query subgraphs chosen independently and uniformly at random, according to node sampling probabilities, p for sources and q for destinations. This leads to $(K, p, q)-$SpotLight graph sketching. **(b) STREAMSPOT** [11] represents each graph G using a shingle-frequency vector z_G. A k-shingle $s(v, k)$ is constructed by traversing edges, in their temporal order, in the $k-$hop neighborhood of node v (also called Ordered $k-$hop Breadth First Traversal). It uses locality-sensitive hashing (LSH) for constant-space graph representation in a streaming graphs. STREAMSPOT sketch contains the counts of unique shingles in a graph. However, our sketch vector contains the cost of a discriminative shingle. The sketches generated by all three approaches are provided as input to RRCF.

TABLE III: Performance Comparison Between GODIT, Spotlight, and StreamSpot on Top$-k$ Anomalous Graphs in the Stream

Algorithm	**Precision (top$-k$)**				**Recall (top$-k$)**			
	100	**200**	**300**	**400**	**100**	**200**	**300**	**400**
Ground Truth	*1.0*	*1.0*	*1.0*	*1*	*.25*	*.49*	*.74*	*.99*
GODIT	**.91**	**.86**	**.80**	**.67**	**.22**	**.43**	**.59**	**.66**
SpotLight	.75	.53	.38	.30	.18	.26	.28	.30
StreamSpot	.61	.52	.42	.33	.15	.26	.31	.33

2) Evaluation Metrics: The performance of each method is evaluated based on its effectiveness in identifying the DoS attack traffic. As mentioned earlier in Section V, if a graph has at least 50 anomalous edges then it is considered an anomalous graph. Otherwise, it is considered normal. It should be noted that our approach is unsupervised and labeling is done only to provide the ground truth for evaluating the performance of the proposed approach. Therefore, if a graph has a higher number of anomalous edges then they are more anomalous. It is particularly important in DoS attack scenarios where identifying the most anomalous graph means more effective is the approach in detecting severe DoS attacks. For each graph in the stream, the number of anomalous edges is computed. Sorting these in descending order, the top k most anomalous graphs are identified. Each method is then evaluated on how well they can identify the top k anomalous graphs. If there are a total N anomalous graphs (graph with DoS attack traffic), for every k we compute $precision@k = TP(k)/K$ and $recall@k = TP(k)/N$.

3) Results and Discussion: Anomaly detection results on a test set of 1 week IoT traffic are shown in Table III. The anomaly detection is done on the test set that has 7,280 graphs containing 436 ground truth anomalies. RRCF outputs the anomaly score of a graph as a collusive displacement, which measures the change in model complexity incurred by inserting or deleting a given graph. The score above 90th percentile is marked as an anomaly. Precision and recall are reported for the top 100, 200, 300 and 400 anomalous graphs. As RRCF is initialized based on the first 256 graphs, performance is reported on the subsequent $7,280 - 256 = 7,024$ graphs, containing 404 ground truth anomalies (5.7% of total). Ground truth values indicate the ideal score of precision and recall for each k. As shown in Table III, our approach has better precision and recall to baseline approaches for each value of k. Fig. 7 plots the ground truth (blue plot) and the anomaly score (red plot) of all methods in the smart home IoT traffic. The higher the spike the more anomalous is the graph. As shown, our approach was able to identify most of the DoS attack spikes with better precision. Also, it is clear that SpotLight and StreamSpot approach missed several instances of DoS attacks.

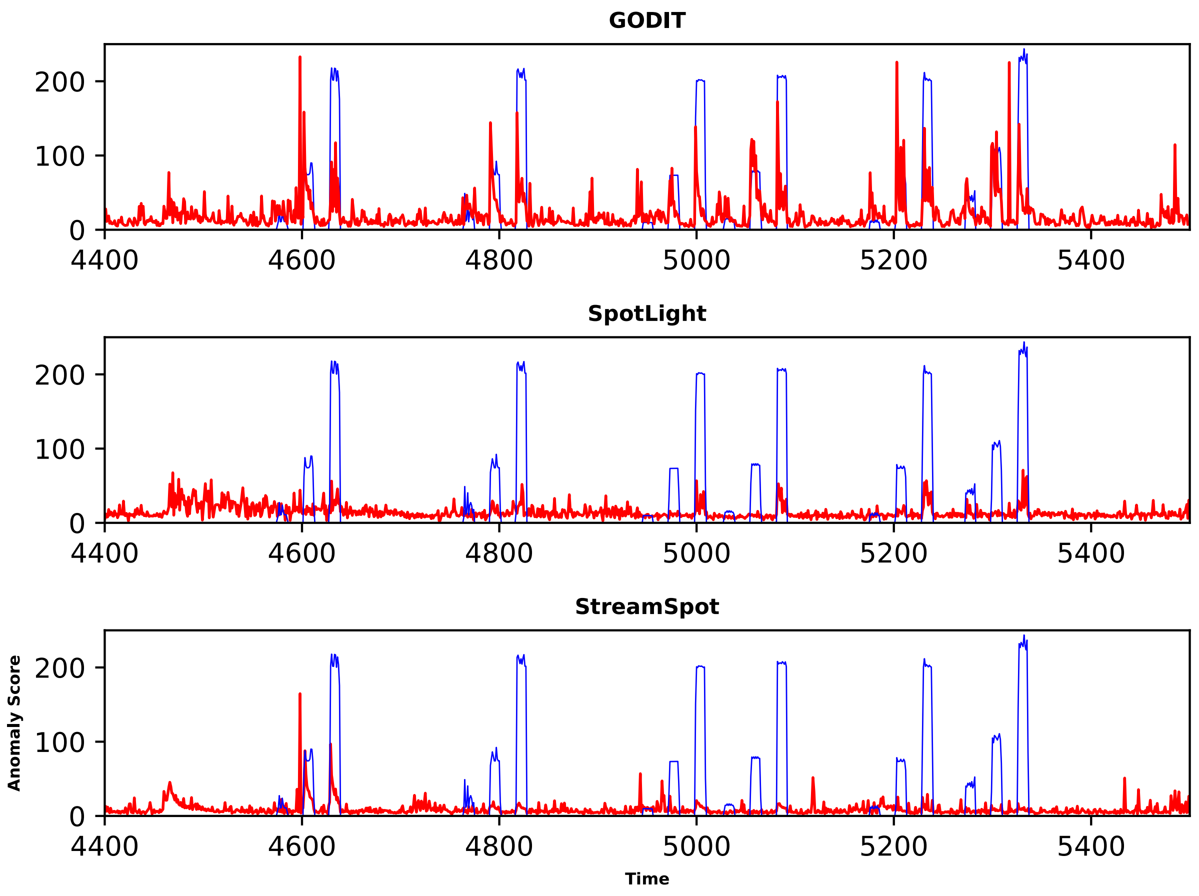


Fig. 7: DoS attack detection on smart home IoT traffic. Blue plot indicates the ground truth anomalies. Spike in red plots indicates the anomalousness reported by the respective approaches.

StreamSpot is primarily designed to detect anomalies in the form of host-level advanced persistent threats. Hence, it performed less effectively in detecting a DoS attack where the anomalies constitute a sudden burst of traffic. SpotLight, is specifically designed to detect anomalies that constitute the sudden appearance or disappearance of dense subgraphs (like found in a DoS attack) performs better than StreamSpot. However, GODIT was able to outperform both baseline approaches. In summary, these results demonstrate the effectiveness of GODIT in identifying the DoS attack in smart home IoT devices.

VII. Conclusion and Future Work

In this paper, we presented GODIT to detect DoS attacks in smart home IoT networks that represent the IoT traffic as a graph stream. We then embedded a graph into a $d-$dimensional sketch vector S_d using a $d-$discriminative $n-$shingle. The sketch vector is then used as input for anomaly detection. Our experimentation shows that GODIT can outperform current graph stream anomaly detection approaches in terms of both precision and recall.

The sketching technique of GODIT needs to be trained at the beginning to generate discriminative shingles. The issue with this is whenever the graph stream undergoes concept drift, then the discriminative shingles need to be updated to match the latest trend. Otherwise, the set of discriminative shingles would not reflect the latest distribution and the performance of the anomaly detection algorithm will deteriorate. Therefore, the proposed approach requires periodical training/updates. In the future, we would like to investigate a way to update discriminative shingles based on the latest distribution of the data. Furthermore, the biased random walk we used for traversing the graph provides the neighborhood information to calculate local and global neighborhoods, but the structure of the graph (number of neighbors, degree, etc.) is omitted. We would like to investigate an approach to integrate this structural information in our sketch vector, which would result in a rich set of information.

References

[1] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman, “Detecting volumetric attacks on lot devices via sdn-based monitoring of mud activity,” in *Proceedings of the 2019 ACM Symposium on SDN Research*. ACM, 2019, pp. 36–48.

[2] D. Minoli, K. Sohraby, and J. Kouns, “Iot security (iotsec) considerations, requirements, and architectures,” in *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2017, pp. 1006–1007.

[3] J. Wurm, K. Hoang, O. Arias, A.-R. Sadeghi, and Y. Jin, "Security analysis on consumer and industrial iot devices," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2016, pp. 519–524.

[4] M. Asplund and S. Nadjm-Tehrani, "Attitudes and perceptions of iot security in critical societal services," *IEEE Access*, vol. 4, pp. 2130–2138, 2016.

[5] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 29–35.

[6] M. Lyu, D. Sherratt, A. Sivanathan, H. H. Gharakheili, A. Radford, and V. Sivaraman, "Quantifying the reflective ddos attack capability of household iot devices," in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, 2017, pp. 46–51.

[7] MarketWatch. (2014) Proofpoint uncovers internet of things (iot) cyberattack. [Online]. Available: https://www.marketwatch.com/press-release/proofpoint-uncovers-internet-of-things-iot-cyberattack-2014-01-16

[8] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "Iot security techniques based on machine learning," *arXiv preprint arXiv:1801.06275*, 2018.

[9] S. Khandelwal. (2016) Friday's Massive DDoS Attack Came from Just 100,000 Hacked IoT Devices. [Online]. Available: https://thehackernews.com/2016/10/ddos-attack-mirai-iot.html

[10] W. Wei. (2018) Casino Gets Hacked Through Its Internet-Connected Fish Tank Thermometer. [Online]. Available: https://thehackernews.com/2018/04/iot-hacking-thermometer.html

[11] 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*. ACM, 2016, pp. 1035–1044.

[12] B. B. Zarpelao, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[13] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "Iot sentinel: Automated device-type identification for security enforcement in iot," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 2177–2184.

[14] M. A. Khan and K. Salah, "Iot security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.

[15] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for iot security and privacy: The case study of a smart home," in *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*. IEEE, 2017, pp. 618–623.

[16] J. Canedo and A. Skjellum, "Using machine learning to secure iot systems," in *2016 14th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2016, pp. 219–222.

[17] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying iot devices in smart environments using network traffic characteristics," *IEEE Transactions on Mobile Computing*, 2018.

[18] M.-O. Pahl, F.-X. Aubet, and S. Liebald, "Graph-based iot microservice security," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–3.

[19] Q. Ding, N. Katenka, P. Barford, E. Kolaczyk, and M. Crovella, "Intrusion as (anti) social communication: characterization and detection," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 886–894.

[20] P. Parveen, N. Mcdaniel, Z. Weger, J. Evans, B. Thuraisingham, K. Hamlen, and L. Khan, "Evolving insider threat detection stream mining perspective," *International Journal on Artificial Intelligence Tools*, vol. 22, no. 05, p. 1360013, 2013.

[21] W. Eberle and L. Holder, "Anomaly detection in data represented as graphs," *Intelligent Data Analysis*, vol. 11, no. 6, pp. 663–689, 2007.

[22] R. Paudel, P. Harlan, and W. Eberle, "Detecting the onset of a network layer dos attack with a graph-based approach," in *FLAIRS Conference*, 2019, pp. 38–43.

[23] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra, "Spotlight: Detecting anomalies in streaming graphs," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1378–1386.

[24] B. Adhikari, Y. Zhang, N. Ramakrishnan, and B. A. Prakash, "Distributed representations of subgraphs," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2017, pp. 111–117.

[25] A. Z. Broder, "On the resemblance and containment of documents," in *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*. IEEE, 1997, pp. 21–29.

[26] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.

[27] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[28] A. Pirotte, J.-M. Renders, M. Saerens *et al.*, "Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation," *IEEE Transactions on Knowledge & Data Engineering*, no. 3, pp. 355–369, 2007.

[29] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE, 2006, pp. 475–486.

[30] K. J. Ahn, S. Guha, and A. McGregor, "Graph sketches: sparsification, spanners, and subgraphs," in *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*. ACM, 2012, pp. 5–14.

[31] C. C. Aggarwal, Y. Zhao, and P. S. Yu, "On clustering graph streams," in *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 2010, pp. 478–489.

[32] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[33] S. Guha, N. Mishra, G. Roy, and O. Schrijvers, "Robust random cut forest based anomaly detection on streams," in *International conference on machine learning*, 2016, pp. 2712–2721.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.