Detection of Demand Manipulation Attacks on a Power Grid

Srinidhi Madabhushi
Department of Computer Science
University of Denver
Denver, USA
nidhi.madabhushi@du.edu

Rinku Dewri
Department of Computer Science
University of Denver
Denver, USA
rdewri@cs.du.edu

***Abstract*—An increased usage in IoT devices across the globe has posed a threat to the power grid. When an attacker has access to multiple IoT devices within the same geographical location, they can possibly disrupt the power grid by regulating a botnet of high-wattage IoT devices. Anomaly detection comes handy to inform the power operator of an anomalous behavior during such an attack. However, it is difficult to detect anomalies when attacks take place obscurely and for prolonged time periods. To effectively detect such attacks, we propose a novel dynamic thresholding mechanism that is used with prediction-based anomaly score techniques. We compare our detection rates to predefined thresholding mechanisms and commercial detection methods and observe that our method improves the detection rate up to 97% across different attacks that we generate.**

***Index Terms*—attack detection, anomaly detection, intrusion detection system**

I. Introduction

With many security risks that Internet of Things (IoT) imposes [1], there is a threat to the power grid as well, when these devices are breached and manipulated by an adversary. This category of attacks where IoT devices are controlled by an attacker with the goal of affecting the power grid was discussed by Soltan et al. and termed as Manipulation of Demand via IoT devices (MadIoT) attacks [2]. The primary research question we explore in this work is whether it leaves avenues for an attacker to inject higher power usages without getting noticed. In fact, since statistical learning methods aim to not overfit a model to the training data, there remains the possibility for an attacker to exploit a model to work within its learned parameters and still inflict damage.

In this paper, we generate demand manipulation attacks using available real-world power consumption data as discussed in Section IV, and demonstrate gaps in anomaly detection techniques that rely on predefined thresholds calculated for different time windows. We propose a novel thresholding mechanism based on a spring system in Section V that aims to detect anomalies which other predefined thresholding mechanisms fail to detect. This approach has provided encouraging results by detecting 100% of attack situations in some cases. We also evaluate and compare the performance of our method against other detection methods, including five commercially used applications in Section VI.

II. Related Work

Comprehensive surveys of the research on anomaly detection has been performed in various works [3]. Sisworahardjo and Saad performed a spatio-temporal context anomaly detection to detect irregular power consumption using a normalized anomaly score [4]. A novel unsupervised anomaly detection algorithm for commercial buildings was provided by Janetzko et al. [5]. Chahla et al. have explored the deep learning approach to anomaly detection and prediction of power consumption [6]. Cui and Wang proposed a hybrid model that combines polynomial regression and Gaussian distribution to detect anomalies in school electricity consumption data [7]. Zeng et al. used a stream anomaly score technique for finding contextual and collective anomalies in streaming data [8]. Araya et al. aimed to detect contextual and collective anomalies based on a sliding window approach using mean square errors [9]. There are dynamic thresholding mechanisms proposed which are based on the mean, variance and standard deviation of the historic data [10]–[12], but using such statistical approaches allows the thresholds to adapt to the attack cases as well.

III. Background

A. MadIoT Attacks

The MadIoT attack, which is short for Manipulation of Demand via IoT, is a term introduced by Soltan et al. [2] in their paper published in 2018. They briefly describe how different attacks can be possible in the power grid due to manipulation and control of various IoT devices by synchronously switching them on and off, leading to the disruption of the power grid. They demonstrate attacks that can be broadly classified into one of the three types: (i) attacks that result in frequency instability that can lead to a sudden generation tripping or disrupting a grid re-start, (ii) attacks that cause line failures and cascading failures, and (iii) attacks that increase operating costs without causing any line overloads leading to the Independent System Operators (ISOs) purchase additional power in the form of reserve generators.

B. Score-based Anomaly Detection

Anomaly detection mechanisms can be designed using a prediction method followed by a scoring technique which provides a score specifying the extent to which a data point should

be considered as an anomaly. A thresholding mechanism is then used to provide a threshold for the scores beyond which the instance will be flagged as an anomaly.

We use six time series prediction methods namely, seasonal naïve [13], simple exponential smoothing [13], weighted average [14], Holt Winters' [13], long short-term memory (LSTM) [6], [15] and convolutional neural network (CNN) [16]. After the prediction method is applied, we compute an anomaly score as implemented by Janetzko et al. [5], where the score for a given time t is as follows.

$$a_t = \frac{|x_t - y_t|}{avg_{t' \epsilon T}|x_{t'} - y_{t'}|} \tag{1}$$

where y_t is the prediction obtained for the time t, x_t is the observed value for the time t, T is the set of all time units starting from 1 until $(t-1)$.

C. Commercial Anomaly Detection Methods

In this study, we also compare five commercial methods and packages that are used in the area of anomaly detection. The methods we use are HTM Studio which is a biologically inspired machine intelligence technology developed by Numenta [17], Arundo's ADTK which is a Python package for unsupervised/rule-based time series anomaly detection [18], Twitter's AnomalyDetection which is an R package that automatically detects anomalies in big data [19], Anomalize package in R which enables a tidy workflow for detecting anomalies [20] and Facebook's Prophet which is a Python package for forecasting time series data [21].

IV. Attack Generation

A. Power Consumption Data

The power consumption data used in this study is obtained from the UMass Trace Repository [22] and it consists of minute-level power consumption for 114 single family apartments in the year 2016 from January to mid December in West Massachusetts. We use the sum of the consumption of the 114 apartments to represent a grid level consumption in kilowatts.

B. Common Terms

1) Threshold: A threshold is a value above which a point is flagged as an anomaly. This is applied to an anomaly score.

2) Adder: The adder represents the power usage (number of kilowatts) that an attacker is able to increase for a time unit.

3) Attack Profile: A transformation performed on the original dataset by adding adder values to the attack duration gives an attack profile.

4) Gain: Gain is the total power usage (kilowatt) that the attacker is able to inject by increasing the demand values throughout the attack period.

5) Anomaly Detection System: An anomaly detection system consists of a prediction method to predict the consumption values followed by a scoring mechanism that uses an anomaly score, and a thresholding mechanism, to flag anomalies. It can also be a commercial detection application.

C. Thresholding Mechanism

We calculate the anomaly scores for the original dataset without modifying the consumption to explore different window sizes for predefined threshold calculation. With the assumption that the dataset does not have any anomalous values, we expect that a chosen threshold should generate as few false positives as possible. We use ten types of threshold mechanisms which differ by the window size. The window size represents how often the threshold is calculated which can be one of ten intervals- yearly, half yearly, quarterly, monthly, weekly, daily, every twelve hours, every six hours, every one hour and every fifteen minutes. In each window, we set the threshold as the 99^{th} percentile of the anomaly scores in that window.

D. Assumptions

The following are the assumptions based on which the attacks are carried out.

1) The original dataset has no anomalies.
2) The attacker has access to various IoT devices in all the 114 apartments.
3) The amount of demand per apartment can be of any value.
4) The attacker is aware of the anomaly scoring technique and threshold values used by the power grid operator to monitor the consumption.

E. Process Overview

First, we generate multiple attack profiles for each threshold type. An attack profile is generated in such a way that it is not detected while using an anomaly detection system that consists of the weighted average prediction method and anomaly scores to compute the severity of an anomaly. This brings us to the second step where we select 10 attack profiles based on false positive rate and the gain achieved during the attack period. We use R version 3.6.1 for implementing the algorithms to generate the attacks and to calculate the evaluation metrics.

F. Attack Profiles

The attack profiles are generated by performing an obscure progressive attack on the power grid by controlling the overall demand of the grid. All attacks are performed for two weeks by adding some amount of adder, with a minimum value of 0. The attacker assumes weighted average prediction method, but is aware of how the score is calculated. Hence, the adder value is determined by reverse engineering the anomaly score computation in such a way that the assumed anomaly detection system cannot detect the increase in demand. The threshold values can be computed by the attacker by observing the consumption values in the grid and compute the 99^{th} percentile in a chosen window length. If the adder is zero at a time instance, the demand is not manipulated by the attacker. The adder is computed using the following formula.

$$x'_t = (a_t \; avg_{t' \epsilon T}|x_{t'} - y_{t'}|) + y_t \tag{2}$$

$$d_t = x'_t - x_t \tag{3}$$

where x_t is the existing consumption value, x'_t is the resultant demand to be created for time t which also consists of the adder, a_t is the anomaly score which is set equal to the threshold at t, y_t is the prediction of the consumption for time t, and d_t is the adder for time t which is calculated by finding the difference between the resultant demand and the already existing consumption value.

1) Attack Profile Selection: For yearly, half yearly and quarterly thresholds, we create an attack profile for each quarter and for the others, an attack profile for each month is created. A total of 96 attack profiles are generated from which we select 10 profiles based on the gain in kW achieved during the attack period. First, we sort all the 96 profiles in decreasing order of the gain values. If the attacks are carried out during the first quarter, we ignore them to provide a buffer time to stabilize the anomaly scores. Starting from the first valid attack with the highest gain, we uniformly choose different attacks representing different gain values. In order to include all threshold types in this selection, we ensure to choose one attack per threshold type. If we hit a profile whose threshold type is already considered, we look at its neighbors on either side to search for a profile with a different threshold type. This technique is used to get attack profiles that are distributed between the minimum and maximum gain values. Table I shows the list of selected attacks.

TABLE I. List of selected attack profiles with net gain

Threshold Type	Attack Month	Wattage Gain
Yearly	July	9,426 MW
Half yearly	July	5,225 MW
Quarterly	July	3,207 MW
Monthly	June	4,569 MW
Weekly	September	2,128 MW
Daily	December	1,235 MW
Twelve hours	May	4,063 MW
Six hours	April	2,742 MW
One hour	September	1,903 MW
Fifteen minutes	May	3,501 MW

2) Time for MadIoT Attack: As we perform the attacks obscurely, we have the limitation of not increasing the wattage all at once to reflect the effects of the attack within seconds. However, we analyze the time it took to reach the gains required to carry out each of the variations of the MadIoT attacks. Table II shows the gain values for each attack effect which are calculated using the botnet sizes mentioned by Soltan et al. [2] from 114 apartments, each contributing one IoT device. Except for daily threshold, we are able to reach the required gains for all attack scenarios within 2.5 hours. The smallest gain value, which is 380 kW, is reached within 5 minutes, and gains of 570 kW and 2,280 kW are attained within 10 minutes. The time to reach the larger gain values varies by threshold type, with yearly threshold achieving it the fastest within 27 minutes.

V. Dynamic Thresholding using Spring System

The attacker can exploit the gap between the scores and the thresholds to perform the attack such that the demand is increased or decreased, but the anomaly score lies below the threshold line, thus staying undetected. The drawback of these thresholds is that as it depends on the consumption values, it does not detect the attack entirely and starts adapting to the anomalous observations. After analyzing the thresholds used for attack generation, we put together the following characteristics of a thresholding mechanism that will help identify attack instances: (i) it should detect the start of any anomalous situation, (ii) it should detect when the anomaly scores are constant as such anomalies are contextual and may not necessarily exceed the threshold, (iii) it should detect collective anomalies, (iv) it should detect anomalies during any season, and (v) it should have low false positives when any prediction method with an anomaly score is used. With these characteristics in place, we will discuss the methodology for the proposed novel thresholding approach.

TABLE II. Wattage requirement from 114 devices for MadIoT attacks

Attack effects	Wattage from 114 IoT devices
Generators tripping	380 kW
Disrupting a black start	570 kW
Cascading failure	11,400 kW
Failure of tie lines	7,600 kW
Increase in operating costs	2,280 kW

A. Methodology

Consider a traditional spring system consisting of a spring attached to a fixed surface. The spring is in its resting position when no force is applied. Every spring has a resistance, which is a reaction of the spring due to an external force. There are some properties of the spring system that we used to design the dynamic thresholding model. The first property is non-linear resistance which suggests that the spring system has a non-linear relationship between the force applied on the spring and the displacement of the spring. This principle enforces that the higher we want to compress the spring, the more difficult it becomes. This ensures that the thresholding mechanism does not adapt to increase in the anomaly scores. The second property is varying resistance which indicates that a constant force applied to a spring for a long period of time is unable to retain the same level of compression. In other words, the resistance of the spring can be viewed to increase such that the displacement occurs in the opposite direction of the force. This enforces that the longer a score is constant, the more difficult it becomes to sustain a threshold. This is required as the consumption always has peaks and troughs and is unlikely to be constant.

1) Resistance Model: Initially, the spring is in its resting position and we want to apply some force to reach a target threshold. However, the spring is compressed only until a value less than the target threshold, which is considered to be the actual threshold due to the resistance model of the spring. We will represent the resistance function $R(x)$ with displacement x as follows.

$$R(x) : R^+ \rightarrow [0, 1] \tag{4}$$

When we try to reach a target threshold $T_{target}(t)$, the resistance model $R(x)$ is applied such that the spring is displaced until a smaller actual threshold $T_{actual}(t)$ as follows.

$$T_{actual}(t) = \int_0^{T_{target}(t)} (1 - R(x))\, dx \tag{5}$$

We set the target threshold at each time t to be the twice the 99^{th} percentile of past anomaly scores in the most recent 15 minutes time window $W_{as-recent}$ (i.e. $t-1$ to $t-15$).

$$T_{target}(t) = 2\ Q(0.99|W_{as-recent}) \tag{6}$$

When a spring system does not have any resistance, then the resistance function will be $R(x) = 0$, thus the resulting equation of actual threshold will be $T_{actual}(t) = T_{target}(t)$. Similarly, when the spring system has infinite resistance, $R(x) = 1$, and we get $T_{actual}(t) = 0$. This means that there is no displacement at all.

We model the resistance function $R(x)$ as the cumulative distribution function (CDF) of the anomaly scores in a time window as the CDF also has a range of $[0, 1]$. Let X be a random variable of anomaly scores, x represents some anomaly score and W_{as} be a window of anomaly score observations in the past one week. Then the resistance function $R(x)$ is represented as follows.

$$R(x) = Pr(X < x|W_{as}) \tag{7}$$

where $Pr(X < x|W_{as})$ represents the probability of the variable X to be less than x in the learning window W_{as}.

The upper curve of Fig. 1 has the CDF of the scores. We observe that the cumulative distribution function is very steep and 99 percent of the anomaly scores are less than 5 in the learning window. Hence, we introduce some slack, called resistance factor represented by r_f which is calculated by taking the ratio of 90^{th} and 99^{th} percentile of scores in the learning window W_{as}.

$$r_f = \frac{Q(0.9|W_{as})}{Q(0.99|W_{as})} \tag{8}$$

After introducing the resistance factor, we calculate the resistance as follows.

$$R(x) = Pr(X < x.r_f|W_{as}) \tag{9}$$

By introducing the resistance factor, for a value of x that earlier had a resistance of 0.99, will now decrease to 0.9. This can be observed in the bottom curve of Fig. 1.

In order to accommodate the varying resistance property in the model, we move between two resistance functions given by $r_f = 1$ (upper curve of Fig. 1) and $r_f = \frac{Q(0.9|W_{as})}{Q(0.99|W_{as})}$ (lower curve of Fig. 1). We represent the variable which will control the shift factor between these two functions as s_f, where $s_f = \frac{1}{r_f}$. The range of s_f is $[1, \frac{1}{r_f}]$. To find the shift factor, we use an exponential drop function which is calculated as follows.

$$s_f = C(c_t) = max(1, \alpha e^{\beta c_t}) \tag{10}$$

where c_t is the coefficient of variation of the anomaly scores at time t, $\alpha = \frac{1}{r_f}$, $\beta = \frac{ln(r_f)}{Q(0.001|W_{cv})}$ and W_{cv} is the sequence of coefficients of variation in a rolling window of one day.

From Fig. 2, we observe that majority of values of the coefficient of variation lie between 0.5 and 1 which is why it is very unlikely to be 0. This confirms that the consumption is never constant for a long period of time. We will be using this observation as the basis for finding if there is a constant anomaly score over a long period.

After including the shift factor, (9) changes to the following.

$$R(x) = Pr(X < x.r_f.s_f|W_{as}, W_{cv}) \tag{11}$$

The final resistance model $R(x)$ described in (11) is then used as the resistance in (5) to find the value of the actual threshold at time t. The learning process, which updates r_f every week and controls the shift factor s_f by updating it every fifteen minutes is what makes this mechanism dynamic and will be discussed in the next subsection.

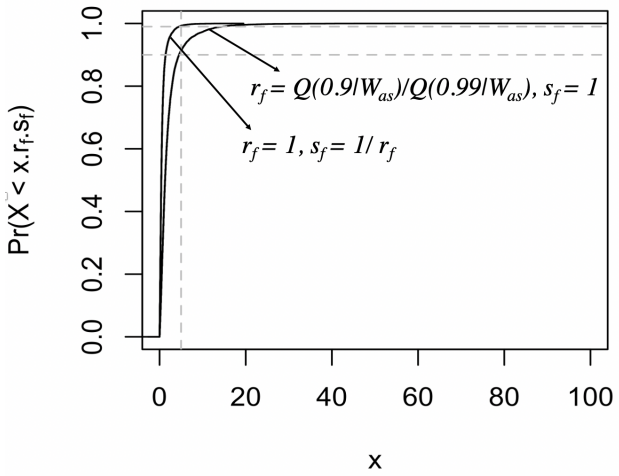

Fig. 1. Original CDF (upper curve) and slacked CDF (lower curve) of the anomaly scores during Week 4 of the year

2) Steps to Learn the Anomaly Scores: The learning process takes place every week, where the learning parameters α and β are updated. We use only non-anomalous past observations for the learning process and are identified by checking which time units have anomaly scores less than their respective dynamic thresholds. Then, we learn the cumulative distribution of these scores and also find the 99^{th} and 90^{th} percentiles of the scores. Using these percentile values, we calculate the resistance factor r_f as shown in (8). Now, we compute and

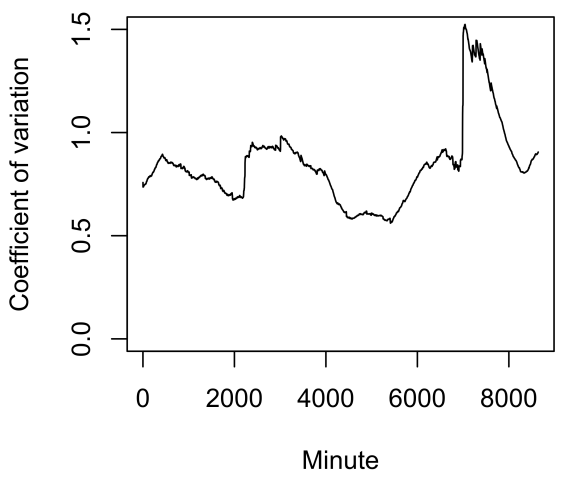

Fig. 2. Coefficient of variation of anomaly scores during Week 4 of the year

store the value of the first learning parameter α which is the reciprocal of r_f. To compute the second learning parameter β, we first create subsets of non-anomalous scores. Each subset is created starting from the first non-anomalous observation until the next anomaly that is observed. This process is then repeated for every occurrence of an anomaly. We then find the coefficients of variation for each subset by using a rolling window of one week and find the 0.1^{th} percentile of all the computed coefficients of variation. This will give the value of the denominator of β, where $\beta = \frac{ln(r_f)}{Q(0.001|W_{cv})}$. As we previously calculated the value of r_f, we compute the value of β and store it. The result of the learning process is the computation of the learning parameters α and β that control the shift factor.

3) Steps to Generate the Threshold: When the dynamic thresholding mechanism runs for the first time, it learns the scores from the previous week. For every time unit in the current week, we set the target threshold to be twice the 99^{th} percentile of the anomaly scores in the most recent fifteen minute window. Equation (6) shows this step where $W_{as-recent}$ is set to the last fifteen minutes. Then, we calculate the coefficient of variation of anomaly scores in the past one week and find the shift factor for the current time unit as shown in (10). Note that, r_f, α and β values are set during the learning process. We find the actual threshold that can be reached for the current time unit by using (11) and (5). This process is repeated for every time unit, where a new threshold is calculated and stored as the actual threshold for that time unit. The learning process is triggered every week, where the resistance factor is updated. Whereas, the shift factor controls the threshold during the current week based on the scores from the most recent fifteen minutes.

B. Alert Level

After a threshold is calculated for a time unit, we see whether the anomaly score for that time unit is greater than the computed threshold. Based on the average number of alerts in the last 24 hours, we assign an alert type to each of these 24 hour rolling windows based on the severity of the anomaly occurrence in that window. There are four alert levels that we use as described below.

1) LOW: Mean alerts is between 0% and 25%
2) MEDIUM: Mean alerts is between 25% and 50%
3) HIGH: Mean alerts is between 50% and 75%
4) CRITICAL: Mean alerts is between 75% and 100%

VI. Results

Dynamic thresholding based on the spring system has shown encouraging results in terms of overall true positive rates as well as detecting anomalies in various contexts that predefined thresholds for different time windows and commercial methods could not detect. In this section, we will evaluate the performance of dynamic thresholding and compare it against predefined thresholds and commercial methods.

A. Overall Performance

Dynamic thresholding performed well in detecting anomalies across all prediction methods, when compared to predefined thresholding methods. Table III shows a comparison of average true positive rates (TPR) and false positive rates (FPR) across the 10 attack profiles between predefined and dynamic thresholds. There is a significant improvement in TPR reaching up to 100% detection rates when using simple exponential smoothing in 7 out of 10 attack profiles. Simple exponential smoothing has performed the best across all prediction methods. The TPR for LSTM has also improved and it is able to detect 99.99% of anomalies in 6 out of 10 attack profiles. Seasonal naïve and Holt Winters' have also performed comparatively better in terms of detection rates. CNN is the only method where a drop in detection rates are observed. This is because CNN itself generates predictions that are significantly different from the actual values with an average mean absolute error percentage (MAPE) of 71% across all the attack profiles for all the months in the year. Due to the normalization of the anomaly score, the entire attack duration is not given high scores except if the prediction is closer to the actual or way off from the usual MAPE, which is why only 51% of the attack duration is detected on an average across all attack profiles, with a maximum detection rate of 88.5% for one of the attack profiles.

In terms of false positive rates, dynamic thresholding has 30% for LSTM which is the highest of all prediction methods followed by 27% for Holt Winters'. It has comparatively increased for simple exponential smoothing as well having 18% when using dynamic thresholding. However, for CNN the false positive rates have reduced from 11% to 3%, again because of the consistent errors between predicted and actual values that anomaly score is well adapted to. We will address the reason behind the increase in false positives by using alert levels in the next section.

When looking at the performance of commercial methods, it is observed that these methods are unable to detect collective and contextual anomalies. Moreover, these methods could not detect some clearly visible point anomalies representing the

TABLE III. Comparison of true positive and false positive rates rates between predefined thresholding mechanism (P) and our proposed dynamic thresholding mechanism (D) across all prediction methods

Method	TPR (P)	TPR (D)	FPR (P)	FPR (D)
Seasonal naïve	44.65%	47.04%	7.79%	7.53%
SES	47.98%	97.30%	1.18%	18.05%
Holt Winters'	48.22%	67.80%	9.67%	27.47%
LSTM	67.25%	83.74%	21.88%	30.42%
CNN	67.75%	51.35%	11.02%	3.42%

sudden increase in the consumption. Table IV shows the TPR values for commercial methods which do not have significant detection rates when compared to predefined or our proposed dynamic thresholding mechanism. However, the FPR values are very small for most of the methods, thus having less false alarms.

TABLE IV. Comparison of true positive and false positive rates for commercial methods

Commercial Method	TPR	FPR
HTM Studio	0.16%	0.08%
Anomalize	0.35%	0.90%
Twitter's AnomalyDetection	0.45%	0.00%
Arundo's ADTK	4.98%	5.00%
Facebook's Prophet	39.43%	16.34%

B. Evaluation of Results

In this subsection, we will evaluate how our dynamic thresholding system identifies cases which the commercial methods and predefined thresholds could not capture.

Out of the 50 total attack profile results from all the prediction methods, 44 of them are detected within the first minute when using predefined thresholds. With dynamic thresholding, 46 of them are detected in the first minute and all of the 50 were detected within the same day. Hence, our mechanism was able to detect more number of attack profiles earlier than the predefined thresholds.

The attacks are generated such that the anomaly scores are always below the predefined threshold in a detection system that uses weighted average. This results in having constant anomaly scores which allowed us to generate undetected attacks using predefined thresholds. However, with our dynamic thresholding mechanism, there is an initial rise in the threshold in order to adapt to the increased consumption, and when it is detected to be constant, there is a drop in the threshold, detecting the rest of the attack period successfully.

Commercial methods fail to detect collective and contextual anomalies. Predefined thresholds are also able to detect only a proportion of collective anomalies based on the seasonality and the granularity of the threshold calculation range. Our thresholding mechanism is able to detect majority of the attack periods comparatively. Fig. 3 shows an example of an attack profile whose scores are calculated using Holt Winters' method. The predefined daily threshold is able to detect only half of the anomalies whereas our proposed method detects it entirely. It is observed that the predefined thresholds and commercial methods are unable to detect significant amount of contextual anomalies during seasonal transition months resulting in lower detection rates. Our proposed dynamic thresholding mechanism aids in detecting attacks during these times as well.

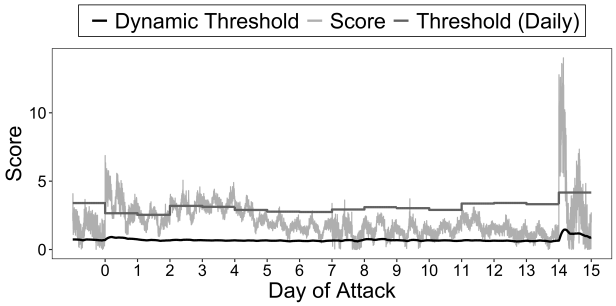

Fig. 3. Detection of collective anomalies in a single attack case by our proposed thresholding mechanism and a predefined daily threshold

Having detected majority of the anomalies in various prediction methods, dynamic thresholding came with a price of having false positives higher than predefined thresholds and commercial techniques with a maximum of 30% across all prediction methods. We will analyze the occurrence of false positives based on how long an alert level was sustained. Fig. 4 shows the alert levels for each 24 hour rolling window during the year. We can see that the attack periods are highlighted in the month of July, and the alert levels for these attack times move gradually from LOW to CRITICAL. Most of the attack periods are at CRITICAL level until the end of the attack proving the ability of our dynamic thresholding method to sustain the CRITICAL alert level throughout the attack. There are some false alerts observed throughout the year. While looking into the average consumption in 24 hour rolling windows, we observed that the consumption reached a peak value mid-February which is why a gradual increase in alert level is also observed during that time. It is observed that due to some peaks and troughs in the consumption until mid-summer, the alerts also move to different levels based on the consumption. These peaks and troughs in consumption occur due to varying temperatures and seasonal changes. We accessed the snowfall data in West Massachusetts from National Centers for Environmental Information and observed that though the frequency of snowfall decreases while transitioning to Summer, the days or times when there is a snowfall until the end of May affects the consumption patterns as well, which is why we see a sudden increase or decrease in consumption during these times. Additionally, Table V shows the top three sustained periods in minutes in decreasing order for each alert level. For CRITICAL level, the first entry shows how long this level is sustained during the attack period. The remaining two entries denote the times from the winter season approximately sustained for around 4 days because after a single day of high snowfall, the temperature usually stays low during the following days. Hence, the false positives do not occur contiguously unless there is an attack or a variability in consumption due to environmental factors.

In summary, our novel dynamic thresholding technique

is able to detect cases that are not detected accurately by predefined thresholds and commercial methods resulting in higher detection rates. Though an increase in the false positive rates is observed compared to the other methods, we conclude that their occurrence is not sustained for a long time except when there is an increased power usage due to snow days.

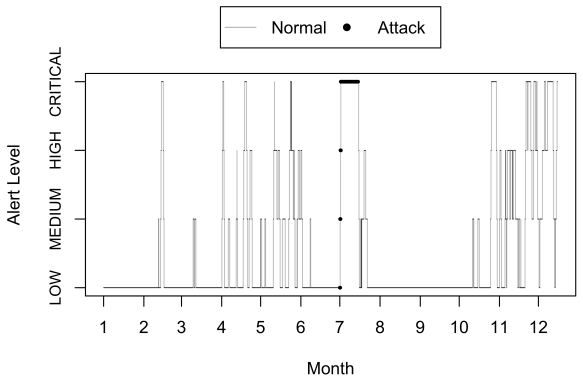

Fig. 4. Alert level for each 24 hour rolling window during the year

TABLE V. Top three sustained times in minutes for each alert level

Alert level	#1	#2	#3
CRITICAL	19782	6244	5562
HIGH	2678	2517	2314
MEDIUM	2864	2667	2614

VII. Conclusion and Future Work

Demand in the power grid can be manipulated by an attacker when they have access to multiple compromised high-wattage devices leading to power line failures, increase in operating costs, or even a blackout. We generated such attack profiles and provided a comparative evaluation of the performance of commercial methods and anomaly detection systems that use a combination of a prediction method and an anomaly score along with predefined thresholds for detecting demand manipulation anomalies. We observed that they were unable to adjust to seasonal changes and contextual changes in demand leading to lower detection rates. Our proposed novel dynamic thresholding mechanism was able to detect attacks taking place during these situations, and also allows gradual changes in the threshold such that it can adapt to seasonal transitions as well as to genuine increase or decrease in demands.

Adversarial machine learning was performed in this work to generate attack profiles that exploit the flexibility retained by a detection technique by not overfitting to historical data. This area needs future work to be performed in the application of the power grid. We observed that we were having false positives due to the external factors like seasonal changes. This requires us to carry out the learning process such that it can accommodate these seasonal transitions as well. Collective anomaly detection is also a possible future direction for sustaining the detection for long periods of anomalous situations.

References

[1] OWASP, "Internet of Things." [Online]. Available: https://owasp.org/www-project-internet-of-things/

[2] S. Soltan, P. Mittal, and H. V. Poor, "BlackIoT: IoT Botnet of High Wattage Devices Can Disrupt the Power Grid," in *27th USENIX Security Symposium*, 2018, pp. 15–32.

[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[4] N. Sisworahardjo and A. A. Saad, "Spatio - Temporal Context Anomaly Detection for Residential Power Consumption," *International Journal on Electrical Engineering and Informatics*, vol. 9, no. 4, pp. 776–785, 2017.

[5] H. Janetzko, F. Stoffel, S. Mittelstädt, and D. Keim, "Anomaly Detection for Visual Analytics of Power Consumption Data," *Computers and Graphics*, vol. 38, pp. 27–37, 2014.

[6] C. Chahla, H. Snoussi, L. Merghem, and M. Esseghir, "A Deep Learning Approach for Anomaly Detection and Prediction in Power Consumption Data," *Energy Efficiency*, vol. 13, pp. 1633 – 1651, 2020.

[7] W. Cui and H. Wang, "A New Anomaly Detection System for School Electricity Consumption Data," *Information (Basel)*, vol. 8, no. 4, p. 151, 2017.

[8] Y. Jiang, C. Zeng, J. Xu, and T. Li, "Real time contextual collective anomaly detection over multiple data streams," in *SIGKDD Workshop on Outlier Detection and Description under Data Diversity*, 2014, pp. 23–30.

[9] D. B. Araya, K. Grolinger, H. F. ElYamany, M. A. M. Capretz, and G. Bitsuamlak, "Collective Contextual Anomaly Detection Framework for Smart Buildings," in *International Joint Conference on Neural Networks*, 2016, pp. 511–518.

[10] J. David and C. Thomas, "Efficient DDoS Flood Attack Detection using Dynamic Thresholding on Flow-based Network Traffic," *Computers & Security*, vol. 82, pp. 284–295, 2019.

[11] I. G. A. Poornima and B. Paramasivan, "Anomaly Detection in Wireless Sensor Network using Machine Learning Algorithm," *Computer Communications*, vol. 151, pp. 331–337, 2020.

[12] O. Salem, Y. Liu, A. Mehaoua, and R. Boutaba, "Online Anomaly Detection in Wireless Body Area Networks for Reliable Healthcare Monitoring," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 5, pp. 1541–1551, 2014.

[13] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice (2nd Edition)*. OTexts, 2018.

[14] M. C. Hao, H. Janetzko, S. Mittelstädt, W. Hill, U. Dayal, D. A. Keim, M. Marwah, and R. K. Sharma, "A Visual Analytics Approach for Peak-Preserving Prediction of Large Seasonal Time Series," *Computer Graphics Forum*, vol. 30, no. 3, pp. 691–700, 2011.

[15] X. Wang, T. Zhao, H. Liu, and R. He, "Power consumption predicting and anomaly detection based on long short-term memory neural network," in *2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, 2019, pp. 487–491.

[16] C. Xu, J. Wang, J. Zhang, and X. Li, "Anomaly detection of power consumption in yarn spinning using transfer learning," *Computers & Industrial Engineering*, vol. 152, p. 107015, 2021.

[17] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised Real-time Anomaly Detection for Streaming Data," in *Neurocomputing*, vol. 262, 2017, pp. 134–147.

[18] "Arundo's ADTK." [Online]. Available: https://arundo-adtk.readthedocs-hosted.com/en/stable/

[19] Twitter, "Introducing Practical and Robust Anomaly Detection in a Time Series," Jan. 2015. [Online]. Available: https://blog.twitter.com/engineering/en_us/a/2015/introducing-practical-and-robust-anomaly-detection-in-a-time-series.html

[20] D. V. Matt Dancho, "Tidy anomaly detection," Oct. 2020. [Online]. Available: https://cran.r-project.org/web/packages/anomalize/anomalize.pdf

[21] Facebook, "Prophet." [Online]. Available: https://facebook.github.io/prophet/

[22] UMassTraceRepository, "Smart* Data Set for Sustainability." [Online]. Available: http://traces.cs.umass.edu/index.php/Smart/Smart