Discovering Breach Patterns on the Internet of Health Things: A Graph and Machine Learning Anomaly Analysis

Prabin B Lamichhane[1], **Hannah Mannering**[2], **William Eberle**[3]
[1, 3] Tennessee Tech University, Cookeville, TN
[2] Loyola University, Baltimore, MD
pblamichha42@tntech.edu, hemannering@loyola.edu, weberle@tntech.edu

Abstract

Due to the rise in the Internet of Health Things (IoHT), cyber-attacks, particularly data intrusions, have become an issue for security experts. In this work, we analyze the performance of traditional statistical, machine learning, and graph-based anomaly detection approaches in response to this problem. We believe that understanding intrusion *patterns* can aid in the prevention of future attacks. In this work, we use the ARMA model for statistical analysis. We also use several machine learning approaches such as multinomial naïve bayes, random forest, neural networks, XGBClassifier, and support vector machines (SVM). However, while our experiments show that machine learning (ML) techniques have higher precision, accuracy, and F1 score than graph-based techniques, there are aspects to a graph-based approach that could aid security experts in the discovery of certain data breaches by combining the graph-based with the statistical and ML methods. Experiments also show combining different anomaly detection techniques allows for a diverse set of intrusion patterns to be discovered. By recognizing the power of both machine learning and graph-based approaches, we analyze their precision and accuracy while explaining how existing state-of-the-art methods can detect breach patterns. Finally, by identifying the *characteristics* of breach patterns, we present information that security experts can use to prevent future data intrusions.

Introduction

IoHT data intrusions have grown increasingly over the past few years (Floyd, Grieco, and Reid 2016; Perakslis 2014; Ronquillo et al. 2018; Seh et al. 2020). In addition, health information technology is vulnerable to ransomware, and hacking (Abawajy, Choo, and Islam 2017; Ronquillo et al. 2018). These data breaches not only result in complications for security experts but also affect patients and organizations, including the ramifications of data loss, monetary theft, and more (Boddy et al. 2016; Chowdhury et al. 2018; Seh et al. 2020).

Due to these complications and the increasing use of informatics-driven healthcare, security experts need to improve their healthcare data infrastructure to ensure the security of patients (Boddy et al. 2016; Perakslis 2014; Ronquillo et al. 2018). To aid in discovering solutions to data breaches and improve existing healthcare security infrastructure, security experts need to understand the interaction between threats, vulnerabilities, and risks (Ronquillo et al. 2018; Floyd, Grieco, and Reid 2016). Most protection guidelines under the Health Insurance Portability and Accountability Act (HIPPA) rely on standard methods of protecting critical healthcare data (Perakslis 2014). However, many attackers can bypass these types of protections. Understanding current data breach patterns and how attackers bypass security protections can allow security experts to better prepare for these attacks in the future.

Data breaches have been a well-researched problem with many of the proposed approaches focusing on solely machine learning techniques (Abawajy, Choo, and Islam 2017; Chowdhury et al. 2018). For instance, approaches have targeted institutional and medical device vulnerabilities. The IoHT encompasses applications such as health sensors and monitoring devices. As the number of IoHT connections increases, the volume of traffic and transactions will inevitably increase. With the increasing volume of IoHT health data, user privacy is likely to be at a greater risk (Floyd, Grieco, and Reid 2016; Perakslis 2014; Ronquillo et al. 2018; Seh et al. 2020). Due to medical device importance and sensitivity, there is a significant need to review and re-investigate healthcare data vulnerabilities.

Among the IoHT, the following are some examples of the types of cyber-attacks used to breach health devices and enterprises: account hijacking, defacement, DDOS, malware, phishing, SQL injection, targeted attacks, etc. However, existing research is limited when it comes to discovering data breach *patterns* (Chowdhury et al. 2018; Abawajy, Choo, and Islam 2017). In this work, we compare statistical, graph-based, and machine learning techniques to discover IoHT breach patterns. While no one solution can discover all intrusions into an IoHT, we propose that the combination of machine learning and graph-based anomaly detection approaches can be used to efficiently discover many forms of intrusion patterns and anomalous behavior within the IoHT.

Related Works

This work focuses on anomaly detection techniques for analyzing breach patterns within IoHT. In this section, we discuss different approaches and explain the shortcomings of

existing security solutions on (Bhatia et al. 2020b; Boniol et al. 2020; Paudel, Harlan, and Eberle 2019).

Previous research has shown that machine learning can effectively uncover targeted healthcare record breaches by identifying attacker behavior (Abawajy, Choo, and Islam 2017; Boddy et al. 2016; Chowdhury et al. 2018). In traditional approaches, the dynamic analysis compares the behavior of unknown malware with that of known malware. Then, normal and abnormal files are fed into a data classifier to train specific machine-learning algorithms (Chowdhury et al. 2018; Agrawal and Agrawal 2015). However, one of the challenges in using machine learning algorithms is obtaining the informative and independent factors that can be used to discover *structure* and relationships between entities (Boniol et al. 2020; Paudel, Harlan, and Eberle 2019).

One approach to analyze healthcare data vulnerability is HEKA, which can be used to monitor personal medical device traffic and detect attacks (Newaz et al. 2020). HEKA connects to personal medical devices to learn the packet information sequence and detect irregular patterns using an n-gram-based approach and several machine learning techniques. Recent research that used a testbed of medical devices found that HEKA can effectively detect different attacks on personal medical devices with an accuracy of 98.4% and F-measure of 98%. However, this approach explicitly targets a single medical device, not the IoHT.

Recent research has explored patterns associated with how data can be exploited (Ronquillo et al. 2018; Floyd, Grieco, and Reid 2016). One method of pattern detection is through data mining and visualization. One such approach identifies patterns by analyzing the number of records stolen per breach, the frequency of each breach type, and the type of data stolen from each breach (Floyd, Grieco, and Reid 2016). Visualization techniques used to identify anomalous behavior include Gephi, GraphPrism, IPMatrix, and YifanHu, each of which allows for network exploration, analysis, and visualization (Boddy et al. 2016; Floyd, Grieco, and Reid 2016). In using these visualization techniques, breaches are visually identified as looking different and standing out compared to normal data patterns. Another visual graph-based approach for anomaly detection is GraphAn GUI, in which the user can change values, subsequently visualizing the embedding space. In the graph visualization, crucial information is highlighted to enable users to detect and group abnormal subsequences (Boniol et al. 2020). There are several existing graph-based anomaly detection approaches. Graph-based approaches like MIDAS (Bhatia et al. 2020a) and TF-IGF (Lamichhane and Eberle 2021) use Count-Min (CM) sketching technique to detect anomalous behavior. Another approach to anomaly detection is NorM, a novel approach that detects anomalies based on their dissimilarity to a model representing normal behavior. NorM detects anomalies based on their distance from the Normal Model sequence (Boniol et al. 2020).

Most anomaly detection methods use supervised approaches and require an information baseline from which training is performed. However, the issue with these approaches is that data must be labeled and known in advance. In addition, other algorithms have been used to identify outliers in datasets. In this case, discovering anomalies is from a statistical standpoint to identify underlying trends within a dataset.

In this study, we will be using machine learning methods, a time-series approach, and a graph-based anomaly detection approach to complement each other in the discovery process. First, for the machine learning approaches, we utilize a variety of models such as multinomial naive bayes. Second, we use ARMA, which uses time series decomposition and statistical metrics (e.g., median) to identify anomalies within a network. We use the time series approach to analyze the impact of detecting anomalies based on a known time stamp. The time-series approach is sequential, utilizing data to train and find the general behavior of data based on time-stamps (Pincombe 2005; Wang, Wang, and Luo 2009). Third, we utilize two graph-based anomaly detection approaches: GBAD and MIDAS. GBAD detects anomalous structures using the Minimum Description Length (MDL) principle to discover both the dataset's normative patterns and subsequently the anomalous substructures (Paudel, Harlan, and Eberle 2019; Dumagpi, Jung, and Jeong 2020). Previous studies have shown that graph-based anomaly detection can exhibit high true positive rates, low false-positive rates, and detect complex structural anomalies in real-world domains. We also utilize MIDAS, a micro-cluster anomaly detection algorithm. MIDAS has shown to outperform baseline approaches by 42-48% accuracy and processes the data 162-644 times faster than baseline approaches (Bhatia et al. 2020a).

Anomaly Detection Methods

Time-Series Approach

In this study, we implement an auto-regressive moving average model (ARMA) to analyze *non-graph-based* approaches for detecting anomalies (Pincombe 2005; Wang, Wang, and Luo 2009).

ARMA is an auto-regressive moving average model that fits time series data to predict future points in the series. Anomaly detection is done by building an adjusted model using outlier points and using t-statistics to check if these points are a better fit than the original model. One reason that we chose ARMA is that it considers both independent and dependent variances. The non-seasonal ARMA parameter AR(p) represents the order of auto-regressive, and MA(q) represents the moving average. Building our ARMA model consists of the following four steps: assessment of model, estimation of parameters, diagnostic checking, and prediction. In the first step, we use the autocorrelation function (ACF) and partial autocorrelation function (PACF) to verify that the mean, variance, and autocorrelation of the time series were consistent throughout the established interval. ACF is a statistical metric that calculates whether the initial values are related to the latest values or not, while the PACF value of the correlation coefficient is between the time-lag and specified variable.

It is essential to determine if a Gaussian distribution is present in time-series approaches. If a Gaussian distribution is present, then the stationarity of data can be examined by

splitting the time series into two or more partitions and comparing the mean and variance of each group. If the mean and variance difference is statistically significant, the time series is likely non-stationary.

We perform a Shapiro-Wilk test at a p-level of greater than 0.05 to evaluate if the data is Gaussian. After determining if the data is non-Gaussian, we perform an Augmented Dickey-Fuller test to confirm if the time series is stationary or non-stationary. The Augmented Dickey-Fuller test is a statistical test known as a root test. This statistical test uses an autoregressive model and optimizes criteria across multiple lag values. In this statistical test, the null hypothesis is that a time series can be represented by a unit root and that it is not stationary. The alternate hypothesis suggests that the time series does not have a unit root, which means that it does not have a time-dependent structure (Mushtaq 2011; Harris 1992; Paparoditis and Politis 2018). We interpret the results using the p-value from the test. A p-value below a threshold of 0.05 suggests that we reject the null hypothesis (stationary). Otherwise, a p-value above the threshold indicates we fail to reject the null hypothesis (non-stationary). To determine if our dataset was stationary, we apply this approach to the individual variables of time, length, type, type of attack, protocol, source, destination, and info.

Machine Learning Approaches

Our study compares six different machine learning approaches' accuracy, precision, and recall to discover anomalies within a healthcare dataset. The approaches include the following: Multinomial Naive Bayes, Logistic Regression, Random Forest, Neural Network, XGB classifier, and Support Vector Machine. Research shows that all these methods efficiently detect anomalies within an IoT environment (Agrawal and Agrawal 2015; Bhattacharya et al. 2020).

Graph-based Approaches

We then utilize the GBAD software to detect anomalies within the dataset. By implementing GBAD, we use an MDL approach that determines the best substructure(s) followed by any anomalous substructures (Eberle and Holder 2007). We also use the MIDAS approach (Bhatia et al. 2020a), which uses streaming data structures within a hypothesis testing-based framework. This study will compare GBAD and MIDAS to the machine learning approaches (previously mentioned) to discover the most accurate method for finding data breach patterns within the IoHT and then combine them as an ensemble-type approach.

Experiments

This section performs experiments on an IoHT dataset using various anomaly detection techniques.

Dataset

In this study, we evaluate the proposed approach using a publicly available dataset that contains known data breaches. The dataset is related to network traffic and contains Address Resolution Protocol (ARP) Spoofing, Denial of Service (DoS), Nmap Port Scan, and Smurf attacks. Edith Cowan University (ECU) in Australia performs the collection of data. The IoHT environment ECU created reflects different attacks that exploit various vulnerabilities. ECU created this environment using the MySignal kit (http://www.mysignals.com/) from Libellium, a development platform for medical devices and eHealth applications. The researchers used different ethical hacking tools to conduct vulnerability analysis and then captured the network traffic using Wireshark. ECU created the data with 111,208 entries from February 24, 2020, to June 6, 2020, to help healthcare security experts analyze attack behavior and allow them to develop countermeasures. The IoHT environment contains information on the time, source, protocol, length, information, and type of data traffic within the system. Furthermore, the dataset was captured in 3 hours and the default time in Wireshark is in seconds. The accuracy and relevance of our dataset can be verified by looking at real-time analysis of data breaches. This dataset is publicly available (ro.ecu.edu.au/datasets/48/).

Experimental Setup

We perform all experiments on a MacBook Pro with 1.4 GHz Quad-Core Intel Core i5 Processor and 16GB main memory. Machine learning approaches and ARMA were both documented on Jupyter Notebooks. GBAD is implemented in C on a Linux machine, and we use an open-source version of MIDAS, which is implemented in C++. All machine learning algorithms are implemented in python.

Evaluation Metrics

For our ARMA experiments, we use average percent error (APE), mean absolute percentage error (MAPE), mean absolute error (MAE), mean percent error (MPE), and mean scaled error (MASE). We perform all experiments using python, with a target of low APE, MAPE, MAE, MPE, and MASE values. We utilize these metrics because they are known to characterize the performance of a time series approach.

For ML models, we compute the different performance metrics like precision, recall, and F-measure, etc.

For MIDAS, we use false positive rate, true positive rate, and threshold. We analyze performance through precision, accuracy, and F-measure for machine learning algorithms. These metrics are used for all other approaches because they characterize performance on anomaly detection approaches.

Experimental Results

In this section, we evaluate the performance of the various anomaly detection techniques. We aim to answer the following questions:

Q1. Accuracy: How accurately does different techniques detect anomalies?

Q2. Pattern Detection: What patterns do the algorithms detect? Do the different methods discover similar intrusion patterns?

Q3. Attack Characteristics: What are the characteristics of the different breaches? How can security experts use

these characteristics to prevent future data breaches? What types of patterns do the algorithms detect? Do the different methods discover similar intrusion patterns?

Accuracy

Since the primary purpose of using the ARMA model is to discover data breaches, the essential criterion in our experiments is the accuracy of our models. The measurements using ARMA can be found in Fig. 1. Based on the analysis of the error values, the ARMA model forecasts anomalies with a 10% error rate. We define an accurate forecast as any error value less than 0.10. As Figure 1 illustrates, all metrics except MAPE are under 0.05. For instance, MASE measures the accuracy of forecasts determined by the mean absolute error of the forecasts divided by the mean absolute error of the in-sample naive forecast. However, a MAPE forecast of 1 means that the mean absolute percent error between the anomalies and the actual anomalies in our data set is 10%. This means that the ARMA model forecasts anomalies with a 10% error rate. However, experts must interpret the results cautiously because the model can be limited in forecasting extreme values. Furthermore, while ARMA can be adept at modeling trends, outliers are difficult to forecast for the model because they lie outside the general trend captured by the model. It is very important, as these indicators significantly influence identifying characteristics of data breaches, which could be beneficial to implementing measures for creating appropriate security measures to defend against threats.

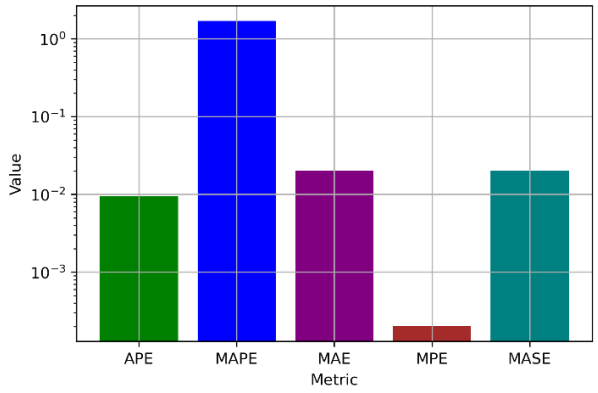

Figure 1: ARMA Evaluation Metrics

Table 1 shows the precision, recall, and F-measure of the machine-learning techniques. We define precision as the percent of relevant instances among the retrieved information. We define recall as the percent of relevant instances that were retrieved. F-measure is a measure of a test's accuracy. We see that the Random Forest and XGB Classifier models are more precise (100%) compared to the other models ($<$ 100%). Compared to the poorest model, there is a 66% precision improvement. Regarding recall, Random Forest and XGB Classifier are also the best. Overall, in terms of the F-measure, the best algorithm is XGB.

To discover the anomalous instances within our dataset, we created a data pipeline in python that processes what

Table 1: Accuracy Results of Machine Learning Method

Model	Precision	Recall	F-measure
Multinomial Naive Bayes	0.63	0.84	0.73
Random Forest	1.00	1.00	0.99
Keras Neural Network	0.04	0.21	0.07
XGBClassifier	1.00	1.00	1.00
SVM	0.34	0.76	0.47

was identified as data breaches within our dataset. For each IP address, all of its traffic is represented in GBAD as an "XP" in the corresponding input file. This allows us to map GBAD results directly from the reported XP(s) to a corresponding flagrant IP address. The results of GBAD identify edge vertices that correspond to data points within our dataset, as seen by Fig. 2. Here, we can see that the vertex pairs (2393, 6224) and (4434, 7644) correspond to the IP pairs (142.250.66.206, 192.168.43.186) and (192.168.43.186, 192.168.43.200), respectively. After GBAD reports the anomalous occurrence(s) through its output, our approach maps the vertices back to the corresponding IP addresses in the original dataset to see if the identification was indeed marked as anomalous. For example, the traffic between 142.250.66.206 and 192.168.43.186 was identified as anomalous by GBAD and is marked as a Nmap Port Scan attack in our dataset. Likewise, the traffic between 192.168.43.186 and 192.168.43.200 was identified as anomalous by GBAD and is marked as a Smurf attack in our dataset. Therefore, we can verify that the data marked with those traffic are anomalous. In our experiments, GBAD is effective at discovering the Port Scan and Smurf attacks.

Fig. 3 plots the receiver operating characteristic (ROC) curves for MIDAS on the IoHT dataset. Since MIDAS primarily focuses on sudden attacks, we observe that such an approach can detect DoS attacks with the area under the ROC curve (denoted by AUC or AUROC) 0.8170 and Nmap Port Scans with AUC 0.6824. Conversely, MIDAS struggles with detecting DDoS attacks, such as SMURF attacks with an AUC of 0.1521.

Pattern Detection

In Table 2, we can see that GBAD can detect Port Scan and SMURF attacks (i.e., DDoS attacks) within our dataset. While a GBAD approach can be used for DDoS attack detection, a dynamic graph approach like MIDAS is better able to detect DoS and Port Scan attacks. As observed in Table 2, each machine learning approach is only able to detect a single type of attack. Also, the machine learning models could not detect port scan attacks, which can be detected using graph-based models. Similarly, among all the machine learning models, only Neural Networks (NNs) can detect DoS attacks; however, NNs accuracy is pretty low (as shown in Table 1). It implies that MIDAS could be a better model to detect DoS attacks than NNs. Yet, machine learning models like Naive Bayes and SVM can be used to detect Spoofing attacks. The crucial observation from our experiments is that we need to consider both graph-based approaches with traditional machine learning techniques - thus, creating a more effective solution for detecting breaches in network traffic.

Anomalous TCP Instance(s):

from positive example 14:

v 2393 SUB_1

v 6224 "source" <-- anomaly (original vertex: 24 , in original example 24)

d 6224 2393 "TCP" <-- anomaly (original edge vertices: 24 -- 117, in original example 24)

(anomalous value = 0.000218)

8143.350938	142.250.66.206	192.168.43.186	TCP	66	443 > 58768 [FIN, ACK] Seq=129389 Ack=2056 W...	Attack	Nmap Port Scan	142.250.66.206 192.168.43.186	24
8143.353476	142.250.66.206	192.168.43.186	TCP	66	443 > 58760 [FIN, ACK] Seq=25532 Ack=1970 Wi...	Attack	Nmap Port Scan	142.250.66.206 192.168.43.186	24

Anomalous ICMP Instance(s):

from positive example 26:

v 4434 SUB_1

v 7644 "source " <-- anomaly (original vertex: 44 , in original example 44)

d 7644 4434 "ICMP " <-- anomaly (original edge vertices: 44 -- 118, in original example 44)

(anomalous value = 0.000218)

256.9764	192.168.43.186	192.168.43.200	ICMP	162	Echo (ping) request id=0x3fb2, seq=295/9985, ttl=59 (reply in 8343)	Attack	Smurf Attack	192.168.43.186 192.168.43.200	44

Figure 2: Attacks detection by GBAD

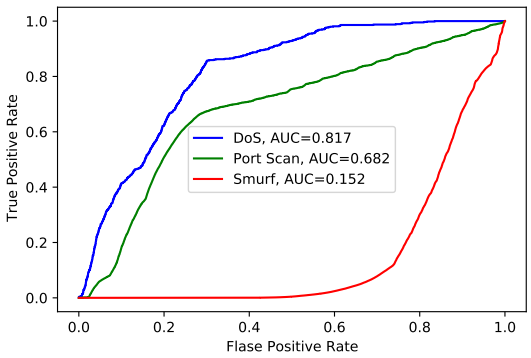

Figure 3: Accuracy Results of MIDAS

Table 2: Data Breaches Detected by Models

Model	Spoofing	DoS	Port Scan	Smurf
GBAD			✓	✓
MIDAS		✓	✓	
Naive Bayes	✓			
Random Forest				✓
Neural Network		✓		
XGBClassifier				✓
SVM	✓			

Table 3: Protocols Detected by Models

Model	TCP	ICMP	ARP
GBAD	✓	✓	
MIDAS	✓		
Naive Bayes			✓
Random Forest		✓	
Neural Network	✓		
XGBClassifier		✓	
SVM			✓

Attack Characteristics

After discovering the various patterns, we want to be able to describe different data breach characteristics. By looking at Table 3, we can see that together, machine learning and graph techniques can detect data breaches with the protocols of TCP, ICMP, and ARP. In addition, the protocols like TLSv1.1, TLSv.1, LLC, OCSP, and HTTP protocols cannot be discovered through our proposed approaches. So, *what is different about these protocols*? It is essential, as these protocols directly correlate to the types of data breaches within our dataset. For example, TCP is used for reliability as packets are processed in a fixed sequence. ICMP is a type of packet used for inter-device communication and is known best for echo requests. ARP maps IP network addresses to the hardware addresses used by a data link protocol. However, our proposed approach is only able to discover attacks regarding individual protocols (Table 3) - hence why LLC, OCSP, and HTTP breaches went undetected. We can identify that data breaches regarding the ARP protocol are best detected by Naive Bayes and SVM methods; ICMP protocol data breaches are best detected by GBAD, Random Forest, and XGBClassifier; and TCP protocol breaches are best detected by MIDAS, GBAD, and NNs (albeit with low accuracy).

Regarding our dataset, SMURF attacks are the most prevalent attack. SMURF attacks are a network layer distributed denial of service attack, which sends a ping record to the host; thus, triggering an automatic response. SMURF attacks also correlate to the protocol of ICMP. So, how can

security experts use this information to discover characteristics to prevent future attacks? By identifying the characteristics of data breach protocols, we identify the protocols that security experts should focus on adding more security. Also, we pinpoint a primary area of concern by identifying that ICMP data breaches are primarily occurring within our IoHT environment. In this case, security experts should design a more efficient methodology to detect attacks that use echo requests.

Conclusion and Future Work

In this work, we evaluate different techniques for anomaly detection within an IoHT environment. In particular, we examined how machine learning and graph-based approaches can both be used to provide a better detection suite. In addition, we were able to identify several patterns representing intrusions, which can ultimately enable a security expert to better identify and defend against attacks. Future work will consider an evaluation of anomaly detection that combines machine learning and graph-based detection to discover intrusion patterns, particularly related to IoHT.

Acknowledgment

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

References

Abawajy, J.; Choo, K.-K. R.; and Islam, R. 2017. *International Conference on Applications and Techniques in Cyber Security and Intelligence: Applications and Techniques in Cyber Security and Intelligence*, volume 580. Springer.

Agrawal, S., and Agrawal, J. 2015. Survey on anomaly detection using data mining techniques. *Procedia Computer Science* 60:708–713. Knowledge-Based and Intelligent Information Engineering Systems 19th Annual Conference, KES-2015, Singapore, September 2015 Proceedings.

Bhatia, S.; Hooi, B.; Yoon, M.; Shin, K.; and Faloutsos, C. 2020a. Midas: Microcluster-based detector of anomalies in edge streams. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Bhatia, S.; Liu, R.; Hooi, B.; Yoon, M.; Shin, K.; and Faloutsos, C. 2020b. Real-time streaming anomaly detection in dynamic graphs.

Bhattacharya, S.; Maddikunta, P. K. R.; Kaluri, R.; Singh, S.; Gadekallu, T. R.; Alazab, M.; Tariq, U.; et al. 2020. A novel pca-firefly based xgboost classification model for intrusion detection in networks using gpu. *Electronics* 9(2):219.

Boddy, A.; Hurst, W.; Mackay, M.; and El Rhalibi, A. 2016. A study into detecting anomalous behaviours within healthcare infrastructures. In *2016 9th International Conference on Developments in eSystems Engineering (DeSE)*, 111–117. IEEE.

Boniol, P.; Linardi, M.; Roncallo, F.; and Palpanas, T. 2020. Automated anomaly detection in large sequences. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, 1834–1837. IEEE.

Chowdhury, M.; Jahan, S.; Islam, R.; and Gao, J. 2018. Malware detection for healthcare data security. In *International Conference on Security and Privacy in Communication Systems*, 407–416. Springer.

Dumagpi, J. K.; Jung, W.-Y.; and Jeong, Y.-J. 2020. A new gan-based anomaly detection (gbad) approach for multi-threat object classification on large-scale x-ray security images. *IEICE TRANSACTIONS on Information and Systems* 103(2):454–458.

Eberle, W., and Holder, L. 2007. Anomaly detection in data represented as graphs. *Intelligent Data Analysis* 11(6):663–689.

Floyd, T.; Grieco, M.; and Reid, E. F. 2016. Mining hospital data breach records: Cyber threats to us hospitals. In *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, 43–48. IEEE.

Harris, R. I. 1992. Testing for unit roots using the augmented dickey-fuller test: Some issues relating to the size, power and the lag structure of the test. *Economics letters* 38(4):381–386.

Lamichhane, P. B., and Eberle, W. 2021. Anomaly detection in edge streams using term frequency-inverse graph frequency (tf-igf) concept. In *2021 IEEE International Conference on Big Data (Big Data)*, 661–667.

Mushtaq, R. 2011. Augmented dickey fuller test.

Newaz, A. I.; Sikder, A. K.; Babun, L.; and Uluagac, A. S. 2020. Heka: A novel intrusion detection system for attacks to personal medical devices. In *2020 IEEE Conference on Communications and Network Security (CNS)*, 1–9. IEEE.

Paparoditis, E., and Politis, D. N. 2018. The asymptotic size and power of the augmented dickey–fuller test for a unit root. *Econometric Reviews* 37(9):955–973.

Paudel, R.; Harlan, P.; and Eberle, W. 2019. Detecting the onset of a network layer dos attack with a graph-based approach.

Perakslis, E. D. 2014. Cybersecurity in health care. *N Engl J Med* 371(5):395–397.

Pincombe, B. 2005. Anomaly detection in time series of graphs using arma processes. *Asor Bulletin* 24(4):2.

Ronquillo, J. G.; Erik Winterholler, J.; Cwikla, K.; Szymanski, R.; and Levy, C. 2018. Health it, hacking, and cybersecurity: national trends in data breaches of protected health information. *JAMIA open* 1(1):15–19.

Seh, A. H.; Zarour, M.; Alenezi, M.; Sarkar, A. K.; Agrawal, A.; Kumar, R.; and Ahmad Khan, R. 2020. Healthcare data breaches: Insights and implications. In *Healthcare*, volume 8, 133. Multidisciplinary Digital Publishing Institute.

Wang, G.; Wang, Z.; and Luo, X. 2009. Research of anomaly detection based on time series. In *2009 WRI World Congress on Software Engineering*, volume 1, 444–448. IEEE.