This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3116575, IEEE Access

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Network Anomaly Detection Using Exponential Random Graph Models and Autoregressive Moving Average

MICHAIL TSIKERDEKIS[1], (SMIEEE), SCOTT WALDRON[2], AND ALEX EMANUELSON.[3]
[1]Western Washington University, Computer Science Department, Bellingham, WA 98225 USA (e-mail: Michael.Tsikerdekis@wwu.edu)
[2]Western Washington University, Computer Science Department, Bellingham, WA 98225 USA (e-mail: waldros2@gmail.com)
[3]Western Washington University, Computer Science Department, Bellingham, WA 98225 USA (e-mail: alexeemanuelson@gmail.com)

Corresponding author: Michail Tsikerdekis (e-mail: Michael.Tsikerdekis@wwu.edu).

ABSTRACT Network anomaly detection solutions are being used as defense against several attacks, especially those related to data exfiltration. Several methods exist in the literature, such as clustering or neural networks. However, these methods often focus on local and global network indicators instead of network structural properties, such as understanding which devices typically communicate with other devices. To address this literature gap, we propose a method that uses exponential random graph modeling to integrate network topology structure statistics in anomaly detection. We demonstrate the effectiveness of our method using real-world examples as a baseline for experiments on domain name system (DNS) data exfiltration scenarios. We highlight how our method provides better insight into how network traffic may alter network graph structure and how this can assist cybersecurity analysts in making better decisions in conjunction with existing intrusion detection systems. Finally, we compare and contrast the accuracy, false positive rate and computational overhead of our method with other methods.

INDEX TERMS graph, ARMA, detection, anomaly, ERGM, network, exfiltration

I. INTRODUCTION

Data exfiltration is a common type of cyberattack that an attacker uses to extract data from a network once unauthorized access to private and possibly sensitive data has been gained. The exfiltrated data can include personally identifiable information (PII), private financial information, usernames with associated passwords and information involving strategic decisions. There exist many communication channels from which data can be exfiltrated. Techniques include but are not limited to HTTP and HTTPS data transfer, email, peer-to-peer file-sharing, SSH, stenography, and protocol tunneling. Every system's communication channel is potentially vulnerable and can be used to extract data out of a network. Although the techniques and methodology for data exfiltration are known to cybersecurity professionals, high-profile incidents of data exfiltration (e.g., DNS exfiltration) can still be observed.

The cost of a data breach has consistently risen in recent years, indicating a need for additional cybersecurity measures. According to the 2019 independently conducted report by Ponemon Institute, the global average of a data breach rose 6.4 percent to $3.86 million, while the cost per leaked or stolen record increased by 4.8 percent to $148 [1]. Companies are currently designating an average budget of $3.6 million to cybersecurity with an increased preference for automated cyber-resilience. From 2017 to 2019, 57% of organizations reported a cybersecurity incident that resulted in significant disruption to the internal processes, while 55% reported a data breach of more than 1,000 sensitive and confidential records. There is no single detection method that is capable of detecting and thwarting all techniques of data exfiltration. By utilizing several methodologies and techniques, companies and professionals can aim to lower their security risk and ultimately increase their ability to ensure data confidentiality. An approach often taken that increases the likelihood of detecting data exfiltration is the use of a variety of anomaly detection methods.

Simple techniques exist for detecting data exfiltration. For example, one such technique measures the total data transfer of a network and detects unusual spikes in volume. This method however has its drawbacks. If an attacker plans to transfer a large amount of accessible data, they can ex-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3116575, IEEE Access

filtrate by progressively increasing their data transmission rate over the span of multiple weeks and therefore evade detection. As a result, more complex anomaly detection methods are required. A recent survey paper on anomaly detection techniques includes several approaches, such as clustering, statistical methods, correlation analysis and neural networks [2]. Proprietary systems such as Darktrace and McAfee Network Threat Behavior Analysis are likely to be using a combination of these approaches. However, none of these anomaly detection approaches use network topology in the form of graphs as a method for determining network anomalies. Arguably, a graph-based view of the network can better describe the state of the overall network and how changes in traffic may alter the structure of the network graph.

A. MOTIVATION AND CONTRIBUTIONS

In this study, we introduce a graph-based statistical inference anomaly detection approach that focuses on detecting data exfiltration on a network. The method's motivation stems from the lack of previous methods that used network topology statistical inference as a means for network anomaly detection. It is tailored for small- to medium-sized networks in which traffic patterns do not change frequently. These types of networks include public and private entity networks (e.g., municipalities and NGOs). The dataset that we used for this study came from one of these entities through the security monitoring nonprofit, PISCES [3]. The aim is for the method to provide an additional point of view of the network to consult when a potential alert occurs.

Our method leverages network topology alongside data flow information to identify anomalous events based on specific user-defined time intervals. This awareness of network topology provides significant advantages over other techniques. For example, anomaly detection techniques in the literature [2] focus on local or global characteristics rather than relations between network devices. The intent is that the model will be trained based on a specific time period (e.g., one week) and beyond that predicts any deviations between currently observed traffic and predicted traffic. The main limitation of the approach lies in the stationarity assumption for data, which can be difficult to achieve using limited temporal datasets. However, this also presents a realistic challenge since network trends often vary significantly if longer time windows are considered.

The main contributions of our approach are described as follows:

- The method is topology aware by leveraging a graph representation to summarize a network's structural properties in the statistical detection of an anomaly, in contrast to measuring activities for individual or global network components (e.g., single or whole network gigabytes of traffic). To the best of our knowledge, this is the first study to use exponential random graph models for network anomaly detection.
- The method is efficient and can be implemented with existing data collection practices for security monitoring, requiring no new data as long as common sensor data (e.g., netflows) are collected.
- The method can be incorporated into the workflow of cybersecurity analysts, enabling both quantitative and qualitative assessments of network conditions and providing an intuitive way of explaining traffic in the network through the use of further graph visualization tools.

B. OUTLINE

The rest of the paper is structured as follows. Section II presents related work on anomaly-based detection for data exfiltration and related background information. Section III describes our proposed methodology. We discuss how networks can be analyzed using exponential random graph models followed by time-series modeling using autoregressive integrated moving average methods. Section IV presents our experimental design that focuses on evaluating our anomaly-based detection approach. Section V categorizes our results on empirical and experimental data and evaluates the validity of our findings. Section VI lists some limitations for our approach that can assist cybersecurity professionals who aim to implement this approach in their daily workflow. Finally, in section VII, we highlight some future challenges and opportunities.

II. RELATED WORKS

In this section, we highlight the most common network attacks that involve data exfiltration, describe a specific type of data exfiltration attack, DNS exfiltration, and introduce background knowledge on network topology analysis and time series models that were used in anomaly detection in this paper.

A. NETWORK ATTACKS AND DATA EXFILTRATION

Common attack scenarios for organizational networks involve exposed database servers or compromised web servers. For example, databases have been found to be accessible through the internet, often with default credentials. However, a more typical example involves an exploitable web server that provides an attacker with a staging area from which he or she can eventually access confidential data on a database server. Insider threat [4], [5] is another attack scenario where an agent inside the network (i.e., an employee of an organization) can seek to extract confidential data out of the network. Furthermore, attackers that pose an advanced persistent threat may seek to hide their communications in the network to avoid being detected by cybersecurity professionals. The threat of confidentiality breach through data exfiltration is further exacerbated by the exponential growth of the number of mobile and IoT devices that are used by organizations. Data exfiltration attacks can involve mobile phones (e.g., exploiting iOS pairing service [6]), 3D printers (e.g., exploiting and stealing intellectual property through a printer's

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3116575, IEEE Access

network functions [7]), and IoT devices (e.g., data exfiltration of casinosâĂŹ participant records via a compromised IoT thermometer [8]).

B. DATA EXFILTRATION

Regardless of the attack goal, attackers will make strategic decisions to obfuscate their data communications through the network. Network transmissions through common avenues (e.g., SSH connection) are likely to raise alerts for network operators and cybersecurity analysts. As such, attackers often seek to use a covert channel through which they can obfuscate their data exfiltration process. All network protocols become possible vectors for data exfiltration between the compromised machine and the attacker [4]. Examples of such behavior include malware (e.g., botnets) [4], [9]–[11], where compromised machines must communicate with a command and control center to receive further instructions or update their configuration. Communication is often attempted via a common network protocol channel (e.g., DNS request or TCP reserved bit space utilization). Similarly, in a targeted attack that is focused primarily on data extraction, the data must travel through the network of the compromised machine before arriving at the desired endpoint [5].

Most defenses for data exfiltration focus on analyzing network traffic flow through the corresponding flow-based metrics. This is an important avenue of research because it is difficult to obfuscate the signatures relating to data exfiltration through network communication protocols. In other words, the attackers must abide by the rules of the communication protocols to transmit their data through them.

C. DATA EXFILTRATION THROUGH DNS

The DNS protocol is a channel that has been used for data exfiltration and has been the subject of study over the past decade. There exist multiple attack vectors that leverage weak points in the DNS protocol, aiming primarily to send data through an unsolicited channel. This method is known as tunneling. The approach is very attractive because most firewalls do not block DNS queries, which nullifies the first line of defense on a victim's machine. More specifically, DNS tunneling has been researched with a variety of methods to analyze possible exfiltrations [11], [12]. There are also DNS exfiltration malware that use a variety of targeted techniques (e.g., DNSChanger, OilRig) [13]–[15].

Existing research in DNS data exfiltration detection falls under two large categories: byte-level analysis of DNS packet flows and analysis of domain name strings in packets. The most rudimentary examples in this category focus on raising an alert past a number of DNS requests over some time window or by measuring domain name entropy as a crude metric for detecting random strings [16] [17]. However, more comprehensive solutions also exist.

The work presented in [18] details a byte-level analysis of DNS packet flow in a network. Analyzing only DNS traffic and its corresponding protocol, three separate attack vectors were explored and analyzed. These attack vectors represented data exfiltration via a file transfer, an interactive session to mimic command and control communications, and a scenario of web browsing over the DNS protocol. A temporal analysis was applied to the data, which helped analyze the behavior of DNS communications for these attack events over time. Various detection methods were then discussed as a means to effectively combat these attack vectors. One such detection method uses a *time-of-the-day-dependent* threshold (i.e., less traffic is expected during night hours, so the detection threshold should be lowered), while an alternative approach looks at various time windows to determine increases or decreases in the average network DNS flow as a means of raising an alert. If the data collected in this work had been analyzed as a time series, both detection methods could be seen to represent aspects present in a typical time series of data where seasonality and a general trend might exist. Similar methods have been used to detect denial-of-service attacks using autoregressive integrated moving average (ARIMA) models [19]. ARIMA models are used to fit time-series data for either predictions or to confirm whether the current observations match our expectations based on past traffic. However, limitations for these methods also exist. Strictly analyzing the byte flow for DNS traffic leaves the detection method open to a boiling frog poisoning attack [20], where the data are exfiltrated slowly, aiming to progressively raise DNS network traffic and in turn alert thresholds.

Alternate approaches for detecting DNS exfiltration that do not focus on the sum of bytes also exist. Namely, character frequency analysis of the DNS domain and subdomain names have provided an alternate approach to detecting a possibly compromised communication channel [12]. Anomalies were quickly discovered when tunneled traffic was compared to legitimate domain traffic by analyzing the unigram, bigram, and trigram frequencies of characters. This approach is effective because it has empirically been shown that domain names follow Zipf's law [21], which expects that the frequency of a word is inversely proportional to its position in the frequency table. Using natural language processing, these domain name "words" can be categorized and identified. The study demonstrated that by using $n - gram$ frequency analysis, they can detect English language exfiltration. This approach is novel in that it does not consider bytes of DNS flow as a predictive factor for data exfiltration but instead compares likely domain signatures with patterns in the English language. A shortcoming of this approach is the lack of robustness when considering multiple languages. Additionally, an attacker could obfuscate data extraction, leading to a decreased anomaly detection performance for this method.

D. NETWORK ANOMALY DETECTION USING GRAPH ANALYSIS

Graph-based anomaly detection methods have been used for computer networks in the past [22]. For example, traffic dispersion graphs have been used to detect anomalies [23]. The approach offers graphical representation advantages since each IP is represented as a node and the packet exchanges

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3116575, IEEE Access

with other IPs are captured in the graph. Another study used time windows to establish snapshots of network traffic as a graph and subsequently used outlier detection to detect anomalies [24]. Compression of network traffic has also been applied using matrix decomposition [25].

Two large surveys examined all available approaches that were found in past studies [22], [26]. Approaches vary from community detection algorithms to edge detection. Further attention is paid to extracting graph information since the number of nodes and edges make such approaches computationally expensive. Most methods have high ROC curves (e.g., 90% accuracy); however, they also tend to have a high false positive rate (e.g., 50%) [22]. None of the past large survey papers have identified any studies that have previously used exponential random graph models as a means for detecting anomalies in computer network traffic.

E. ERGM: STATISTICAL ANALYSIS OF NETWORK TOPOLOGY

Analysis of network topology is effectively an analysis of graphs. The exponential family of functions that are used for statistical analysis of graphs covers a broad range of applications. Exponential random graph models (ERGMs), a subset of this exponential family, characterize graph topology to model and analyze the structural properties of networks [27]. A statistical model of a network is used to capture regularities and recognize uncertainties. The outcome is represented as log-odds (a coefficient). The general probability formula is as follows [28]:

$$P(Y = y|\theta) = \frac{exp\{\theta' s(y)\}}{c(\theta)} \quad (1)$$

where Y is a random network of n vertices, y is the observed network, and θ is a vector of parameters associated with $s(y)$, which are similar to regression coefficients. We assume an observable graph's structure y can be explained with a statistic vector $s(y)$ depending on vertex attributes and the observable network. Finally, c is a normalizing constant to ensure probabilities sum to 1. More specifically, $c(\theta)$ can be written as:

$$c(\theta) = \sum_{\forall y \in Y} exp\{\theta' s(y)\} \quad (2)$$

Each network tie is a random variable, and as such, we can make a probabilistic prediction for a tie between vertices i and j. The network statistics to be added and counted (e.g., edges, 2-star, triangle) must be known a priori and specified by the user. For each additional term, the model must generate an additional statistical parameter influencing the graph. Figure 1 provides simple examples of types of ERGM terms, which describe structural properties in the network.

In a directed graph of size n, there are $2^{n(n-1)}$ possible permutations. Due to the size of the domain, calculating statistics of all possible graphs grows exponentially with the number of vertices in the network. Instead, ERGM leverages Markov chain Monte Carlo (MCMC) [29], a family of probability distribution sampling algorithms, to estimate the statistical probabilities. MCMC is heavily relied upon when computing models requiring integration over a multitude of unknown statistical parameters.

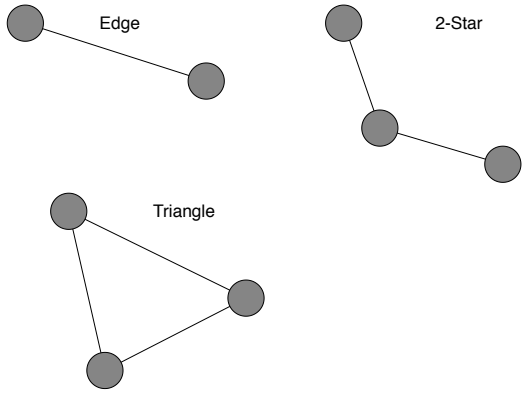

FIGURE 1: An example of network terms to include when calculating probabilities.

In layman's terms, ERGM provides the statistical probability for user-specified structural properties of an observed network based on a large sample of networks that are equal in size to the observed network. For example, we can estimate the probability of triangle formation in an observed network, which in turn informs us of how rare or common that graph property is.

Although originally intended to be used with social graphs (sociographs), ERGM can also be used for network topology analysis since computer networks can be mapped in graphs. For example, the flow of a network can be captured across multiple protocols and can be extended into a graph representation of nodes and edges. Then, this relationship can be analyzed using a variety of statistical user-specified terms.

F. ARMA: TIME SERIES ANALYSIS OF COEFFICIENTS

The aim of time series analysis is to find patterns in noisy data [30]. Typically, such data are empirical (e.g., real-world network traffic) and are in part the result of nondeterministic processes. More specifically, these data may involve events that take place over time and are measured at nonrandom intervals. Examples of time series data include stock market trading data, sport metrics, weather forecasting, etc. The value in modeling and predicting such systems is obvious, but the difficulty lies in discovering a ground truth that can be isolated from the noise. The largest contributor to this difficulty is stationarity. In a time series, the random variable of the distribution is not stationarity if its mean, variance, or covariance change over time. This characteristic is the critical factor in the performance of the model's ability to forecast. It would be extremely difficult for a model to accurately predict values for a system that is truly random. Luckily, there are several methods of data manipulation that can take raw nonstationary data and transform them to become stationary.

The ARMA model is a method for time series analysis that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3116575, IEEE Access

is used for weakly stationary stochastic processes [31], [32]. That is, data with a constant mean, variance and autocorrelation. ARMA models are a combination of two separate time series analysis techniques, namely, autoregression and moving average. The first component of ARMA (AR) is represented by this autoregressive process, where a prediction at time t is influenced by past observations and a random white noise component. An AR model is defined as follows [33]:

$$x_t = \zeta + \varphi_1 x_{t-1} + \varphi_2 x_{t-2} + \varphi_3 x_{t-3} + \dots \varepsilon \qquad (3)$$

where ζ represents a constant (intercept), ϕ_1, ϕ_2, and ϕ_3 are the autoregressive model parameters, and ε is a random noise component.

The second component of ARMA (MA) consists of the moving average process, also known as smoothing. This process is generally used when analyzing data that might have trends or exploring how different time windows might affect trends in the future. The moving average process aims to remove noise from observations to more closely discern the underlying trend of the data. This process can be described as follows [33]:

$$x_t = \mu + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \theta_3 \varepsilon_{t-3} - \dots \qquad (4)$$

where μ is a constant, θ_1, θ_2, and θ_3 are the moving average model parameters, and ε_t, $\varepsilon_t - 1$, $\varepsilon_t - 2$, and $\varepsilon_t - 2$ are random error components at different time instances.

Combining the autoregressive and moving average equations together yields our ARMA model, also written as ARMA(p, q), defined as follows [34]:

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \qquad (5)$$

where φ are the parameters of the autoreggresive component, θ are the parameters of the moving average component, c is a constant and ε is the noise component.

III. PROPOSED METHOD

Our proposed method consists of two main components: the *statistical analysis of the network topology* graph using ERGM and the *time series analysis of the coefficients* using an ARMA model. Figure 2 shows the process through which these elements coalesce.

Implementation of the ERGM component involves identifying the type of traffic that needs to be monitored for anomalies and then representing the traffic (obtained from packet captures or netflows) as a graph consisting of nodes and edges. The graph represents a snapshot in time for the network (i.e., cross-sectional data) that would be typically collected at regular intervals (e.g., hourly or daily). ERGM produces coefficients that describe the local properties of the graph. An example of such a local property is the probability of three devices communicating with one another forming a triangle in the graph. These coefficients form a temporal dataset that can then be validated with a seasonally trained ARMA model to classify an anomaly outside of a desired threshold value. For example, if over the course of a week, triangle formation has a low probability of occurring in our observed network and the probability suddenly changes beyond a certain threshold (e.g., standard deviation), then a network anomaly has occurred requiring a cybersecurity analyst to investigate further.

The underlying approach assumes that network topology does not change often and is geared toward such types of networks. Traffic fluctuations are also expected to be minimal since most ARMA models expect stationarity to be satisfied. Furthermore, the network size should not be too large, as ERGM models are limited computationally by network size. Most medium-sized networks that are monitored in the public and private sectors typically satisfy these conditions. Then, changes in network topology can be anticipated with a false positive alert, since ERGM and ARMA are bound to detect these changes.

IV. EXPERIMENTAL DESIGN

We tested our proposed method to evaluate its accuracy and computational performance on empirical (real-world) data. Our scenario of choice focused on DNS exfiltration attempts. Although this type of data exfiltration could be detected with reasonably good signatures (e.g., measuring the entropy of domain names or the number of requests to the same FQDN) [12], it was chosen due to its simplicity as a vignette that can demonstrate the efficacy of our method. In practice, our method can be applied in detecting any traffic that can be projected into a graph of nodes and weighted edges. Furthermore, we posit that even though cybersecurity experts are aware of such obvious attacks, large data breaches are still occurring using DNS exfiltration, with the most recent example being Equifax's data breach [35].

DNS requests were captured from a local municipal network to produce a realistic network traffic baseline based on which we could test DNS exfiltration incidents. The experimental design consists of four parts. First, we describe our empirical data collection method as well as the injection of DNS exfiltration traffic in our dataset. Next, we discuss the internal validity of the ERGM model and detail the model configuration to provide insight into network analysis applied to DNS exfiltration. We subsequently describe the experimental setup for our ARMA model, which used the coefficients produced by our ERGM model for anomaly detection. Finally, we detail the process followed for our classification experiments that produced the results of our method's accuracy in detecting anomalies.

A. DATA COLLECTION

The Public Infrastructure Security Collaboration and Exchange System (PISCES) is an incorporated nonprofit that provides free cybersecurity event monitoring for several local municipalities in Washington State [3]. PISCES partners include Global Business Resources, Cyber Range Poulsbo, the Washington State Fusion Center, and the Department of Homeland Security Science and Technology Directorate. The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3116575, IEEE Access

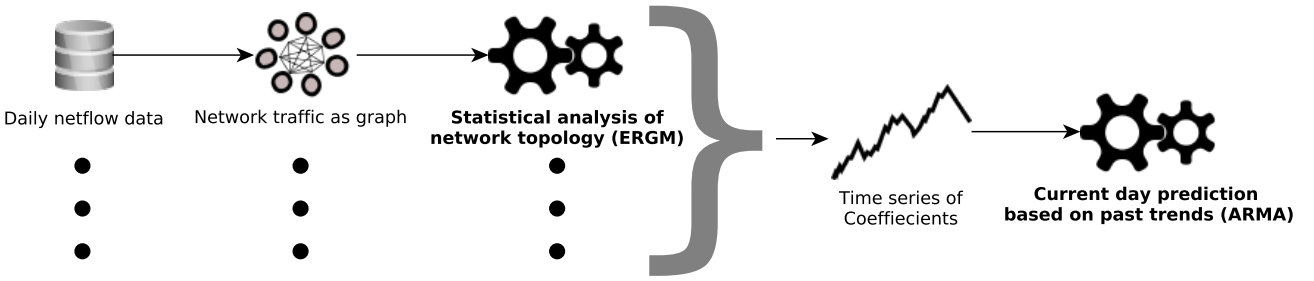

FIGURE 2: A bird's eye view of the components that comprise our topology-aware anomaly detection method.

participating local governments transmit network traffic (in the form of netflows, IDS alerts and DNS records) to be monitored in exchange for no-cost analysis, protection, or best course of action recommendations. PISCES stores the resulting data in an ELK stack.

Since PISCES's beginning of operations in March 2018, petabytes of network traffic data have been collected and organized. However, due to the nature of a deployed system, crashes, bugs, and other oddities occasionally occur during the system's uptime, causing numerous gaps in the collected data. For an ARMA model to provide effective feedback using a small dataset, the sample must be contiguous and must reflect a standard traffic flux void of anomalies. To guarantee this requirement, a week of data for which quality was manually evaluated to be satisfactory was selected, and then proceeding weeks were generated using statistical noise. In this way, we were able to preserve continuity without having to "cherry pick" traffic from different calendar days far apart from one another.

It is important to note that the gathered network traffic is dependent on the days of the week. Weekends experience a much lower volume of DNS requests, while Monday typically has the highest volume. There exists a seasonal trend of network traffic in which weekdays and weekends are distinguishable by their volume of DNS requests and resulting coefficients. Using the gathered ERGM log-odds from the empirical week as a baseline, new data were generated by introducing a randomized multiplicative factor between 0.8 and 1.2. This range was used to observe the range of ERGM values due to the lowest and highest traffic accordingly. The resulting values generated by this process were further evaluated based on whether they reasonably corresponded to a similar empirical day of the week. The result positively satisfied this requirement. Figure 3 shows a time series of an ERGM coefficient that includes the first week based on empirical data and the subsequent generated weeks using the aforementioned process.

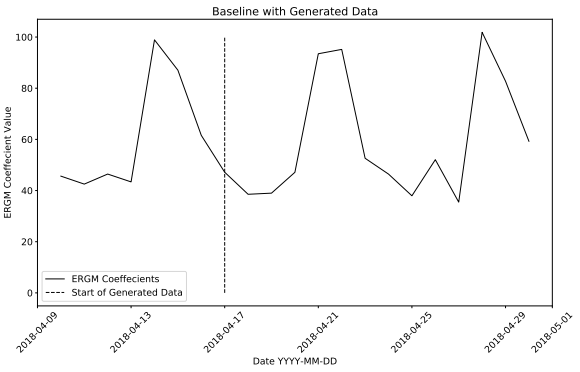

FIGURE 3: One empirical week followed by two generated weeks of data.

B. GENERATING DNS EXFILTRATION

Once a time series of ERGM coefficients based on our baseline network traffic was generated, we proceeded by designing different DNS exfiltration scenarios for various days of the week. In table 1, we can see that a 10 MB DNS exfiltration corresponds to approximately 43,690 additional DNS requests added to edges in the graph. These scenarios are meant to be representative of different conditions that may occur in networks during data exfiltrations. Typical rule-based data exfiltration focuses on over 1 GB of data transfers [36], which, as our results show, are exceedingly detectable using our approach. As such, our experiments focused on determining the lower bound for which the method can still yield satisfactory results. This is due to the limitation that DNS exfiltration has to use the available subdomain space of a domain request, which in practice is less than 253 bytes. For the purposes of our experiment, we assumed that the domain name is 13 characters long and that the attacker has 240 bytes of available space from which to exfiltrate data.

For simplicity, we ignore the fact that Base64 encoding is often used for such purposes (due to non-ASCII compatible characters in the data) that further inflates the number of requests needed. Additionally, because of the presence of DNS relays in network topology, a DNS request may traverse

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3116575, IEEE Access

DNS Volume to Request		
Volume	Approx. No. Requests	2 Hops
10 MB	43,690	87,380
50 MB	218,453	436,906
100 MB	436,906	873,812
500 MB	2,184,533	4,369,066
1 GB	4,473,924	8,947,848

TABLE 1: The approximate number of requests needed for an attacker to exfiltrate volumes of data.

through multiple nodes in a network. In our graph representation of network DNS traffic, we also included the public DNS server, which further increases the number of DNS requests that would appear in the graph if DNS exfiltration were to occur. In fact, as data extraction affects every edge that it travels through, the farther away from the public DNS a request is placed, the more "noise" it would generate in the graph. This in turn will affect the probabilities of a graph's structural properties that are generated by ERGM. Figure 4 shows a simple graph that demonstrates the aforementioned scenario. Clearly, the more isolated a node in the network is, the more "noisy" an exfiltration attempt will be, as more edges are affected by the outflux of requests. The effect of this behavior can be pronounced in larger graphs as the established paths rarely change.

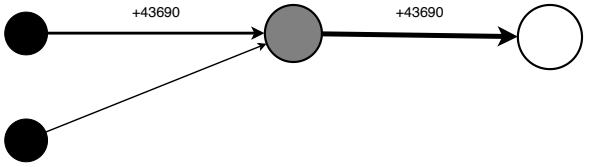

FIGURE 4: An example of a 2-hop flow increase in a network. This is typical DNS traffic where the middle node is a DNS relay node that forwards or DNS requests on behalf of clients to a public DNS server.

Figure 5 shows a graph that includes DNS relay nodes as well as public DNS nodes. Black nodes are associated with IP addresses of hosts with outgoing DNS requests, gray nodes are major internal routers that can also act as DNS relays, and white nodes represent public DNS servers. This cross-sectional graph represents a one-day capture of DNS requests on the traffic of the municipal network monitored by PISCES. This information is produced from a single point of view in the internal network. Noticeably, some hosts directly communicate with public DNSs, whereas most use routers as DNS relays. Additional DNS communications can also be observed between hosts that may have been used by specialized services that leverage the DNS protocol for their applications. The figure demonstrates how even from a single point of view in the network, there are several channels of DNS communication.

C. ERGM CONFIGURATION

The *statnet* [28] package available through R [37] provides the necessary libraries to perform ERGM calculations on

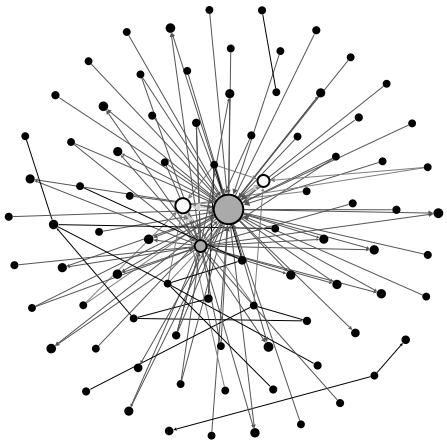

FIGURE 5: A graph representation of DNS requests over a 24-hour window. Black vertices indicate requests being sent, gray vertices are major internal routers, and white vertices indicate public DNS services.

an observed network. In our proposed method for network anomaly detection, the library *ergm.count* in the *statnet* package was used to handle the introduction of valued edges. This allowed us to represent not only the network topology on a graph but also the traffic that traversed through different network paths. The process for estimating coefficients for valued networks is similar, with the only difference being the need to sample estimates for the sampled random networks' edges. The sum of all edge values is polled from one of the user-specified distributions available from the *ergm.count* package (e.g., Geometric, Poisson, Bernoulli).

Since the ERGM methodology is probabilistic, we ran repeated tests to verify that the coefficients produced for the same graph by ERGM are stable (i.e., do not vary substantially for the same graph). With an adequately large, predefined sampling size and a sufficient burnin value (where the initial data generated from MCMC are discarded), coefficients converge to a satisfactory variance. These coefficients were then randomized with the predefined statistical noise described in section IV-A to produce a generated day of traffic for a similar day of the week.

We built the representation of network DNS traffic as a directed graph with valued edges. We used two statistical terms for our ERGM model to maintain a computationally lightweight MCMC estimation. The first term, sum, was used with a geometric distribution to represent probabilities associated with valued edges in a network. This term measures the probability of a single unit increase for a given potential edge in the network. The second term that we used, $atLeast(x)$, measures the probability of finding an edge in the graph with at least x weight. Here, x is the threshold value that is required to determine the log-odds, and we set the value. In the case of our experiment, the threshold value for $atLeast$ was set to the arithmetic mean of all edges in our gathered empirical time series. This hyperparameter is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3116575, IEEE Access

important to consider when deploying this method if this particular ERGM term is used. Additionally, it is important to note that establishing a static threshold value using a normal week of data safeguards against possible adversarial machine learning attacks. For example, in the case of the boiling frog scenario [20], a gradual increase in traffic will not affect the previously defined hyperparameter value unless it is automatically or manually adjusted at time intervals. As a result, this type of attack will eventually be detected as an anomaly. The log-odd coefficients produced by these two terms were then passed onto the ARMA model.

D. ARMA CONFIGURATION

We selected ARMA models for our proposed method as a means to raise alerts due to the models having a component specializing in forecasting from past observations and a component attempting to smooth observations by "learning" the underlying trend. Although these models are typically used for forecasting (e.g., a value 7 days into the future), our use of ARMA models focused instead on verifying how our observed traffic fitted our baseline prediction (i.e., is what we are seeing now what we should be seeing). Typical network traffic follows a strict repetitive pattern. This is especially true for many corporate or critical infrastructure networks where the number of computing devices in a network does not change often. Even networks that include public devices (e.g., public wireless) such as those found in municipal networks can have a fairly stable traffic pattern that varies little aside from on and off days [16]. An example of such an observable pattern is the lesser network activity over the weekends, as there are fewer people using a network. Such network traffic is unlikely to deviate from the norm (once established). That is, traffic on a particular Monday is unlikely to significantly differ from traffic on a following Monday.

Once the ERGM estimation phase of data analysis was completed, we utilized the log-odd coefficients produced by the model to form a time series dataset that could be used by ARMA for prediction. Due to the small sample size of observed days in our sample (14 weeks), any amount of significant noise greatly affects the stationarity of the data. To combat this issue, a differencing method was used where a rolling z - $score$ of a seven-day window was subtracted from the observed value. Augmented Dickey-Fuller unit root tests confirmed the stationarity of the transformed time series data within a 99% confidence interval.

The *pmdarima* [38] software package was used to perform grid search hyperparameter optimization for ARMA focusing on the minimization of the model's Akaike information criterion (AIC). Note that this software package can be used to fit ARIMA models, also denoted as $ARIMA(p, d, q)$. However, since data deflation was to be tightly controlled, to achieve stationarity, the differencing term d was omitted.

Although large training data (e.g., a yearâĂŹs worth of ERGM coefficients derived from daily network graphs) may intuitively mean better predictions, we posit that a) it is practically unfeasible and even excessive to collect such a large dataset, and b) it may hinder the prediction ability of an ARMA model due to variance caused by natural changes in computer networks over the course of a year. Most computer networks in organizations experience changes in devices and in services used on these devices frequently. As such, over the long run, the effect of these network changes introduces non-determinism in a time-series model. Hence, we recommend that for a more realistic model evaluation for daily traffic, a training set should consist of a few weeks of traffic if weekly seasonality is assumed. In other words, the ARMA model needs to be trained on a period where network stability is observed. The number of observations in the test set should also be kept small due to the decreasing prediction confidence of the ARMA model as time progresses from the start of a prediction. In practice, our method is expected to be implemented without any forecasting. Instead, a given current observed traffic would be triangulated with the predicted traffic for any current moment.

The alert threshold used in our proposed method is a threshold of varying standard deviation over the training period. Due to the nature of the time series, this method of detection incorporates temporal variations in the threshold. As such, it allows for a higher threshold on days typically associated with higher volumes of traffic. An alert is raised if the observed value is outside the configured threshold from the predicted value. In our experimental design, this translates to the log-odds ERGM coefficient being outside the baseline established by the ARMA model. Figure 6 shows an example of predictions made by a trained ARMA model along with an alert threshold of 1 standard deviation (shown as an error bar). The dashed lines for the final two weeks of the time series represent the observed traffic coefficients. The traffic is considered normal for these days since the observed and predicted traffic does not deviate beyond our alert threshold.

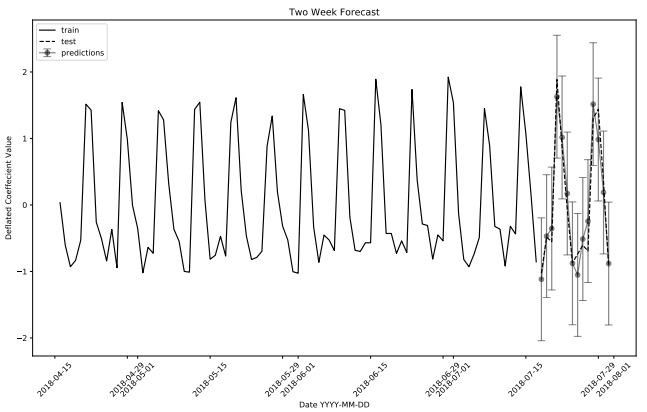

FIGURE 6: ARMA Model fit using approximately 3 months of training data and prediction over a two-week period with 1 std. error bars. The observed traffic is shown in dashed lines for the predicted two weeks. The predicted and observed coefficients do not deviate beyond 1 std. error bar.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3116575, IEEE Access

E. EXPERIMENTAL PROCEDURE

For the purposes of this experiment, we developed three data exfiltration scenarios based on data volume. Low volume data exfiltration was defined as any amount in the range of 10 MB to 100 MB. Medium corresponded to 100 MB up to 500 MB. Last, high volume data exfiltration was any amount of data between 500 MB and 1 GB. In terms of data exfiltration volume, all these categories are relatively conservative. In past incidents of data exfiltration, multiple gigabytes of data have been typically exfiltrated out of a network [8].

Our experiments aimed to identify the classification accuracy of our approach on a) detecting incidents of DNS exfiltrations and b) classifying days with normal traffic correctly. To generate a malicious attempt, the observed network for a given day of the week was injected with requests corresponding to each value of exfiltration in table 1. A two-hop process was used as the basis for DNS exfiltration. This is a typical network path that DNS requests follow in many networks (see figure 4 for an understanding of a two-hop increase). Seven normal week days from our generated set were selected as candidates for testing our method's ability to detect anomalies. These days were then injected with DNS traffic at random. The graphs containing the injected DNS traffic (representing data exfiltration) were then used as ERGM input to calculate the statistical terms (mentioned in section IV-C) and extract the coefficient values.

We trained our ARMA model on 14 weeks of generated traffic that was based on our initial PISCES data. A testing week was randomly generated as normal traffic (see figure 6 for visual depiction of this approach). Subsequently, a random day in the testing week was replaced with a randomized coefficient value relating to the volume of DNS exfiltration based on table 1. Then, the trained ARMA model would predict the testing week. The observed and predicted coefficients are then compared with respect to our alert threshold values. By choosing a week for testing (as opposed to a single day), the model would have to not only accurately predict an exfiltration that occurred over the course of a week but also correctly classify the day that the exfiltration occurred. As such, the validity of the tested accuracy was increased. The results were obtained for three separate alert threshold values to demonstrate how fine-tuning may affect the detection accuracy of our approach. These threshold values were a) one standard deviation, b) one-half standard deviation, and c) one-quarter standard deviation.

This process was repeated 500 times, generating a testing week each time that was comprised of six normal days of traffic and one day in which DNS exfiltration was attempted. Put simply, our experiments contained 3000 normal and 500 abnormal day classification attempts for every alert threshold (e.g., one standard deviation) and every DNS exfiltration category (e.g., low data exfiltration, 10 MB to 100 MB).

V. RESULTS

In this section, we show the results of our experiment for our proposed method. We describe below the accuracy results and highlight the computational requirements for our method.

A. PERFORMANCE

For our evaluation, we utilized the following common classification metrics: accuracy, precision, recall and F_2 score.

We calculated precision to identify the false positive rate for our method. This was defined as follows:

$$Precision = \frac{T_P}{T_P + F_P} \tag{6}$$

where T_P is the number of true positives and F_P is the number of the false positives.

We subsequently estimated recall to identify how well our method will help us avoid false negative incidents. Simply put, recall is implemented to describe how many data exfiltrations the method failed to detect. Recall was defined as follows:

$$Recall = \frac{T_P}{T_P + F_N} \tag{7}$$

where T_P is the number of true positives and F_N is the number of false negatives.

We summarized the interplay of precision and recall using accuracy, which we defined as:

$$Accuracy = \frac{T_P + T_N}{n} \tag{8}$$

where T_P is the number of true positives, T_N is the number of true negatives, and n is the size of the experimental trials (i.e., the number of days that we tried to identify as normal or anomalous).

The standard F_1 score is the harmonic average of both precision and recall, weighing both of these values equally. In the case of our results, an F_2 score is the preferred method of displaying performance. An F_2 score weighs more heavily on recall, providing a better performance metric when evaluating models to detect anomalous behavior. The F_2 score was defined as follows:

$$F_{\beta=2} = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall} \tag{9}$$

where β is 2 in this example, indicating an F_2 score.

The results of all our experiments are summarized in table 2. There exists a positive correlation between the threshold value and accuracy. That is, as the threshold value decreases, accuracy follows. When the threshold value decreases, the model is stricter about the temporal pattern of the data, therefore classifying more false positives and fewer true negatives. Precision can be observed to have a positive correlation with the threshold value, as we would expect. If the system classifies an anomaly, precision details how often that classification is correct. As the threshold value decreases, we increase the classification of false positives attributed to the drop in precision.

Possibly, a more important metric to consider is recall. Recall is representative of a system's "completeness." If a data exfiltration event were to occur, recall details how often

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3116575, IEEE Access

DNS Exfiltration Detection Performance (n=3500)						
Volume	Threshold	Precision	Recall	F_N	F_2	Accuracy
Low	1 std.	0.644	0.294	70.4%	0.330	**0.876**
	1/2 std.	0.433	0.524	47.6%	0.503	0.834
	1/4 std.	0.209	0.78	22%	**0.504**	0.548
Medium	1 std.	0.847	0.622	37.8%	0.657	**0.93**
	1/2 std.	0.549	0.918	8.2%	**0.809**	0.88
	1/4 std.	0.263	1.0	0%	0.641	0.6
High	1 std.	0.931	0.712	28.8%	0.747	**0.951**
	1/2 std.	0.546	1.0	0%	**0.857**	0.881
	1/4 std.	0.262	1.0	0%	0.640	0.597

TABLE 2: Performance metrics across different volumes of exfiltration with varying threshold values.

it would be caught by the system. In the case of medium data exfiltration, using a threshold of one-half standard deviation provided a recall score of .918 (or 91.8%). By restricting the threshold to one-quarter standard deviation, there is not much improvement in recall, and in fact, precision decreases. As such, this threshold value is an important hyperparameter to consider when implementing our method.

In practice, this indicates a point of diminishing returns in defining a lower threshold value. By increasing the number of false positives, the usefulness of anomaly detection is decreased. As such, an ideal "golden" threshold can be found where recall remains high without substantially influencing precision. In terms of our evaluation metrics, accuracy can be rather deceptive since it focuses on the overall improvement of either precision or recall (similar to the F_1 score, which also represents the harmonic mean between these two metrics). However, in practice, we want to prioritize recall first in anomaly detection to avoid security incidents that are costly to organizations.

Detecting DNS exfiltration that is below 100 MB (a rare event for data exfiltration) is difficult for our method unless a more aggressive threshold is selected at the expense of increasing the false positive rate. However, our method can identify almost all DNS exfiltration attempts above the 100 MB data exfiltration threshold (medium and high data exfiltration) with a reasonable number of false positive alerts.

In fact, as the volume of DNS exfiltration increases, the accuracy and F_2 score also increase, indicating that any exfiltration beyond the upper bound of our experiment (1 GB) is almost definitively detectable .

B. COMPUTATIONAL REQUIREMENTS

Network topology is the main factor affecting the ability of our method to scale. Given that ERGM estimates an exponential number of graph permutations, increasing the number of nodes (i.e., increasing the number of network devices) would exponentially increase the number of possible permutations that need to be computed. However, organizational networks or those of a government building rarely exceed a reasonably small upper bound (e.g., a few hundred to a few thousand network devices). In our demonstrated experiment, the local municipality had 120 devices. One ERGM estimation of the network using the experimental dataset required approximately 20 minutes of computing time on an *Intel Core* i7-7700 @ 3.60 GHz. As such, this method can be reasonably implemented on an hourly or daily basis. Typically, that is a reasonable time frame for a cybersecurity analyst to investigate an incident. Faster infrastructure could potentially allow for minute-by-minute classification of traffic; however, it is unclear whether that time window may contain enough meaningful information (device communications and seasonality) for our method. Future research will aim to investigate the application of our method to microscale events.

A further requirement relates to ARMA requiring a minimal time window to predict and train reliably. Additionally, the model only needs to be trained periodically to predict new trends in the network flow. If a data point is known to be void of anomalous behavior, it can be quickly added to the training dataset, and retraining is not computationally expensive. Furthermore, ARMA predictions can potentially be made in near real time.

C. COMPARISONS WITH SIMILAR APPROACHES

We provide comparisons between our method and other anomaly detection methods that have been used in the past in table 3. These are not meant to be exhaustive but rather representative of some of the options that organizations have at their disposal. Due to variations between the parameters and experiments presented for these methods, ranges or averages are presented for comparison purposes. Hence, while we are presenting a large false positive range for our method, most implementations will optimize the parameters to fit closer to the lower end of the range.

Overall, there is an inverse relationship between false positive rates and overall accuracy with all methods. In labeled datasets, neural networks tend to perform much better [39]. However, supervised models tend to perform well in 10-fold cross-validations (99% accuracy with less than 1% false positives); however, in unknown types of attacks, the performance degrades substantially (64% accuracy with 18% false positives) [40]. The study demonstrated that unsupervised models (e.g., clustering algorithms) dealing with unlabeled data tend to perform better at detecting anomalies (80.15% accuracy with 21.14% false positives), which is a substantial advantage since labeled training sets are not required, making implementation much easier.

Furthermore, although computational overhead relates to the algorithm used, certain techniques are substantially more expensive, such as clustering approaches. Models that require training prior to prediction (including our method) tend to have a substantial cost for training but much faster performance when predicting whether an anomaly is present in a testing set. A further variance in the overhead presented in table 3 exists due to the number of nodes considered in each approach. Networks with many nodes make training infeasible for graph-based approaches; however, it is attainable for other supervised models due to the nature of how data points are structured in those methods. As such, when one considers the feasibility of implementing a method in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3116575, IEEE Access

an anomaly detection workflow, the size of the network is a significant consideration that will influence computational overhead.

	Statistical Detection [18]	Clustering [40]	Neural Networks [39]	Our Method
Accuracy	16.7% - 87.5%	57.81% - 80.15%	88.64% (avg)	87.6% - 95.1%
False Positives	Inverse as accuracy increases	20%	10% (avg)	6.9% - 79.1%
Computing Overhead	Low	Medium-High (linear to quadratic [41])	Low (after training is completed)	Low (after ERGM / ARMA training is completed)

TABLE 3: A comparison of our method with similar approaches.

Overall, our method has comparable computational efficiency to other methods and can yield similar results in terms of accuracy for anomaly detection. As such, it can serve as an additional decision system for larger artificial intelligence systems that analyze network behavior for anomalies.

VI. LIMITATIONS

There are a few limitations that we need to highlight in regard to our method. These relate to constraints on the initial data, problems that may arise from ERGM calculations, and assuring the best model performance for anomaly detection.

First, the collected input data for our method must be contiguous and devoid of abnormalities. If the input data are noncontiguous, another time series analysis model (other than ARMA) will need to be used that can account for data gaps. The training data must also be representative of the network traffic norm. In other words, the network traffic must be devoid of any extreme abnormalities such as an outage that makes computers inoperable. This would greatly affect model performance, as the model needs to learn the proper network data trend. When training data are rotated (e.g., updated every 7 days with more recent data), the effect of these abnormalities can be minimized. However, it is still worth noting that such network traffic abnormalities will raise an alert as an anomaly. Additionally, other methods may need to be used to increase the stationarity of the time series (a requirement by ARMA). Ultimately, this decision would need to be made a priori, as there are multiple detrending methods for time-series models [42], [43].

The size of the graph is a large limiting factor for our proposed method. As stated in section II-E, the number of permutations of the graph grows exponentially with the number of nodes added (devices in the network). Therefore, there is a limit to the size of the network where the computation time required by the ERGM process becomes prohibitively excessive.

The model relies on stable networks where the network structure does not frequently change. Our experimental results do not include cases where nodes may have moved (e.g., dynamic IP) or have otherwise been removed (e.g., system fault) or added to the network. All of these are factors that should be tested in future work and may affect the accuracy, as presented in the current model. In general, small- to medium-sized networks do not experience such frequent changes, but the domain of application typically defines this behavior.

Finally, the configuration of the ARMA model will more than likely require human input. The model will need to be validated before it is deployed. This can ensure that there are enough data for the model to learn the network traffic norm and that model predictions are within reasonable expectations. A final challenge that requires further investigation relates to networks that change structure very frequently. This does not affect the majority of applications of the method, but highly dynamic networks will require additional considerations. A radically dynamic structure of a network will reflect a dynamic graph, which in turn will lead to ERGM coefficients potentially becoming unreliable. In turn, ARMA's predictions would not be able to identify a proper network trend. The end result is likely to create an anomaly detection method that is either highly insensitive to rapid network changes or highly sensitive (depending on the thresholds that are defined for anomaly alerting).

VII. FUTURE WORK

Future work should explore the method's detection accuracy for data exfiltration in a real-time implementation. We anticipate that the mean and variance of any real-time time series is likely to change over time and become nonstationary, which is an important condition for the goodness of fit of ARMA. Studies need to investigate how further manipulation of data and relationships should be approached in such incidents.

Furthermore, real-world traffic introduces more network noise, requiring a more finely tuned approach of parameter values for the models. For example, a network's constant traffic growth (e.g., more workstations added to the network) will result in a steady increase in network traffic, leading to an increase in the mean and variance of transmitted bytes. This would result in a decrease in the stationarity of the training data used in the ARMA prediction model. One approach that can resolve this challenge is the process of retraining the ARMA model periodically. This approach can help predictions remain accurate in relation to the actual underlying network topology and developing trends.

A further focus of our work aims to adapt the methodology to network compromises other than data exfiltration over the DNS protocol. For example, attackers can utilize protocols such as HTTP and HTTPS traffic for command and control communication as well as data exfiltration [44]. Network graphs for HTTP and HTTPS netflow relations are different from DNS traffic. In HTTP (or HTTPS) traffic, the graph of observable vertices is substantially larger if external IP addresses were to be included. In turn, ERGM coefficient estimation becomes prohibitively computationally expensive (e.g., 30-hour run time). A potential approach that can reduce

the size of these graphs (e.g., derived by HTTP traffic) is by classifying any address outside the observable local area network as a single supernode (i.e., aggregating external network nodes into one) or by clustering these external subnets based on some other networking factor (e.g., autonomous system number).

VIII. CONCLUSION

In conclusion, companies and organizations are becoming invested in building cyber-resilience as data exfiltration attempts become increasingly frequent [1]. New methods for anomaly detection can help cybersecurity analysts secure the infrastructure of organizations. The method proposed in this paper serves as an additional tool for analysts that can assist in detecting data exfiltration. We have demonstrated the efficacy of our method in an example scenario of data exfiltration, and further exploration will determine the accuracy of the method in other domains of application. Our hope is that our methodology can be useful in reducing the possible workload of cybersecurity analysts by allowing them to only investigate truly anomalous network events.

REFERENCES

[1] Ponemon Institute, The 2019 Study on the Cyber Resilient Organization, Tech. rep. (2019).
URL https://www.ibm.com/downloads/cas/GAVGOVNV

[2] M. Ahmed, A. Naser Mahmood, J. Hu, A survey of network anomaly detection techniques, Journal of Network and Computer Applications 60 (2016) 19–31. doi:https://doi.org/10.1016/j.jnca.2015.11.016.
URL http://www.sciencedirect.com/science/article/pii/S1084804515002891

[3] M. Hamilton, CI Security Partners with PISCES to Provide a Public Option for Cybersecurity Monitoring (2018).
URL https://ci.security/news/article/ci-security-partnership-pisces-cybersecurity-monitoring

[4] A. Giani, V. H. Berk, G. V. Cybenko, Data exfiltration and covert channels, Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense V 6201 (May 2006) (2006) 620103. doi:10.1117/12.670123.

[5] Y. Liu, C. Corbett, K. Chiang, R. Archibald, SIDD: A Framework for Detecting Sensitive Data Exfiltration by an Insider Attack (2011) 1–10doi:10.1109/hicss.2009.390.

[6] C. J. D'Orazio, K. K. R. Choo, L. T. Yang, Data Exfiltration from Internet of Things Devices: IOS Devices as Case Studies, IEEE Internet of Things Journal 4 (2) (2017) 524–535. doi:10.1109/JIOT.2016.2569094.

[7] Q. Do, B. Martini, K. K. R. Choo, A Data Exfiltration and Remote Exploitation Attack on Consumer 3D Printers, IEEE Transactions on Information Forensics and Security 11 (10) (2016) 2174–2186. doi:10.1109/TIFS.2016.2578285.

[8] O. Williams-Grut, Hackers once stole a casino's high-roller database through a thermometer in the lobby fish tank (2018).
URL https://www.businessinsider.com/hackers-stole-a-casinos-database-through-a-thermometer-in-the-lobby-fish-tank-2018-4

[9] A. Al-Bataineh, G. White, Analysis and detection of malicious data exfiltration in web traffic, Proceedings of the 2012 7th International Conference on Malicious and Unwanted Software, Malware 2012 (2012) 26–31doi:10.1109/MALWARE.2012.6461004.

[10] T. W. Fawcett, C. J. Cotton, W. D. Sincoskie, Detection of Data Exfiltration Using Entropy and Encryption Characteristics of Network Traffic, Ph.D. thesis, University of Delaware (2006).

[11] N. M. Hands, B. Yang, R. A. Hansen, A Study on Botnets Utilizing DNS, in: Proceedings of the 4th Annual ACM Conference on Research in Information Technology - RIIT '15, ACM Press, New York, New York, USA, 2015, pp. 23–28. doi:10.1145/2808062.2808070.
URL http://dl.acm.org/citation.cfm?doid=2808062.2808070

[12] K. Born, D. Gustafson, Detecting DNS Tunnels Using Character Frequency Analysis, SciencePrimer.com (2010) 1arXiv:1004.4358.
URL http://scienceprimer.com/boyles-lawhttp://arxiv.org/abs/1004.4358

[13] Y. Nadji, R. Perdisci, M. Antonakakis, Still Beheading Hydras: Botnet Takedowns Then and Now, IEEE Transactions on Dependable and Secure Computing 14 (5) (2017) 535–549. doi:10.1109/TDSC.2015.2496176.
URL http://ieeexplore.ieee.org/document/7312442/

[14] P. Paganini, Analyzing OilRig's malware that uses DNS Tunneling (2019).
URL https://securityaffairs.co/wordpress/84125/apt/oilrig-dns-tunneling.html

[15] A. Nadler, A. Aminov, A. Shabtai, Detection of malicious and low throughput data exfiltration over the DNS protocol, Computers Security 80 (2019) 36–53. doi:10.1016/j.cose.2018.09.006.
URL https://linkinghub.elsevier.com/retrieve/pii/S0167404818304000

[16] M. P. Collins, Network security through data analysis from data to action, 2nd Edition, O'Reily Media Inc., 2017.
URL https://www.oreilly.com/library/view/network-security-through/9781491962831/

[17] C. Sanders, J. Smith, Applied Network Security Monitoring: Collection, Detection, and Analysis, Syngress, Waltham, MA, 2014.
URL https://books.google.com/books?id=TTIDAQAAQBAJ

[18] W. Ellens, P. Żuraniewski, A. Sperotto, H. Schotanus, M. Mandjes, E. Meeuwissen, Flow-Based Detection of DNS Tunnels, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 7943 LNCS, 2013, pp. 124–135. doi:10.1007/978-3-642-38998-6_16.
URL http://link.springer.com/10.1007/978-3-642-38998-6_16

[19] A. H. Yaacob, I. K. Tan, S. F. Chien, H. K. Tan, ARIMA Based Network Anomaly Detection, in: 2010 Second International Conference on Communication Software and Networks, IEEE, 2010, pp. 205–209. doi:10.1109/ICCSN.2010.55.
URL http://ieeexplore.ieee.org/document/5437603/

[20] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, J. D. Tygar, Adversarial machine learning, in: Proceedings of the ACM Workshop on Security and Artificial Intelligence (AISec '11), ACM Press, New York, New York, USA, 2011, p. 43. doi:10.1145/2046684.2046692.

[21] L. A. Adamic, B. A. Huberman, Glottometrics, Glottometrics 3 (1) (2002) 143–150.

[22] L. Akoglu, H. Tong, D. Koutra, Graph based anomaly detection and description: a survey, Data Mining and Knowledge Discovery 29 (3) (2015) 626–688. doi:10.1007/s10618-014-0365-y.
URL https://doi.org/10.1007/s10618-014-0365-y

[23] D. Q. Le, T. Jeong, H. E. Roman, J. W.-K. Hong, Traffic Dispersion Graph Based Anomaly Detection, in: Proceedings of the Second Symposium on Information and Communication Technology, SoICT '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 36–41. doi:10.1145/2069216.2069227.
URL https://doi.org/10.1145/2069216.2069227

[24] C. R. Harshaw, R. A. Bridges, M. D. Iannacone, J. W. Reed, J. R. Goodall, GraphPrints: Towards a Graph Analytic Method for Network Anomaly Detection, in: Proceedings of the 11th Annual Cyber and Information Security Research Conference, CISRC '16, Association for Computing Machinery, New York, NY, USA, 2016. doi:10.1145/2897795.2897806.
URL https://doi.org/10.1145/2897795.2897806

[25] J. Sun, Y. Xie, H. Zhang, C. Faloutsos, Less is More: Sparse Graph Mining with Compact Matrix Decomposition, Statistical Analysis and Data Mining: The ASA Data Science Journal 1 (1) (2008) 6–22. doi:https://doi.org/10.1002/sam.102.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.102

[26] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, N. F. Samatova, Anomaly detection in dynamic networks: a survey, WIREs Computational Statistics 7 (3) (2015) 223–247. doi:https://doi.org/10.1002/wics.1347.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.1347

[27] G. Robins, P. Pattison, Y. Kalish, D. Lusher, An introduction to exponential random graph (p*) models for social networks, Social Networks 29 (2) (2007) 173–191. doi:10.1016/j.socnet.2006.08.002.
URL https://linkinghub.elsevier.com/retrieve/pii/S0378873306000372

[28] M. S. Handcock, D. R. Hunter, C. T. Butts, S. M. Goodreau, M. Morris, statnet: Software Tools for the Representation, Visualization, Analysis and Simulation of Network Data Mark 24 (1) (2008) 1–11.

URL http://www.jstatsoft.org/v24/i01/paper%5Cnpapers2://publication/uuid/8CCE9B1B-3345-4D85-87FE-B2A8D6E8F50E

[29] C. J. Geyer, Practical Markov Chain Monte Carlo, Statistical Science 7 (4) (1992) 473–483.
URL http://www.jstor.org/stable/2246094

[30] J. Contreras, R. Espínola, F. J. Nogales, A. J. Conejo, ARIMA models to predict next-day electricity prices, IEEE Transactions on Power Systems 18 (3) (2003) 1014–1020. doi:10.1109/TPWRS.2002.804943.

[31] H. Zhang, S. Zhang, P. Wang, Y. Qin, H. Wang, Forecasting of particulate matter time series using wavelet analysis and wavelet-ARMA/ARIMA model in Taiyuan, China, Journal of the Air Waste Management Association 67 (7) (2017) 776–788. doi:10.1080/10962247.2017.1292968.
URL https://www.tandfonline.com/doi/full/10.1080/10962247.2017.1292968

[32] R. H. Shumway, D. S. Stoffer, ARIMA Models, Springer International Publishing, Cham, 2017, pp. 75–163. doi:10.1007/978-3-319-52452-8_3.
URL https://doi.org/10.1007/978-3-319-52452-8_3

[33] J. L. Torres, A. García, M. De Blas, A. De Francisco, Forecast of hourly average wind speed with ARMA models in Navarre (Spain), Solar Energy 79 (1) (2005) 65–77. doi:10.1016/j.solener.2004.09.013.
URL http://www.sciencedirect.com/science/article/pii/S0038092X04002877https://linkinghub.elsevier.com/retrieve/pii/S0038092X04002877

[34] C. Peng, M. Xu, S. Xu, T. Hu, Modeling multivariate cybersecurity risks, Journal of Applied Statistics 45 (15) (2018) 2718–2740. doi:10.1080/02664763.2018.1436701.
URL https://doi.org/10.1080/02664763.2018.1436701https://www.tandfonline.com/doi/full/10.1080/02664763.2018.1436701

[35] D. Williamson, Protecting networks from DNS exfiltration (2017).
URL https://www.helpnetsecurity.com/2017/10/02/dns-exfiltration/

[36] IBM, Data Exfiltration (2020).
URL https://www.ibm.com/support/knowledgecenter/SSKMKU/com.ibm.extensions.doc/r_data_exfiltration.html

[37] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria (2018).
URL https://www.r-project.org/

[38] T. G. Smith, pmdarima (2019).
URL https://pypi.org/project/pmdarima/

[39] Y. Yu, J. Long, Z. Cai, Network Intrusion Detection through Stacking Dilated Convolutional Autoencoders, Security and Communication Networks 2017 (2017) 4184196. doi:10.1155/2017/4184196.
URL https://doi.org/10.1155/2017/4184196

[40] I. Syarif, A. Prugel-Bennett, G. Wills, Unsupervised Clustering Approach for Network Anomaly Detection, in: R. Benlamri (Ed.), Networked Digital Technologies, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 135–145.

[41] D. Xu, Y. Tian, A Comprehensive Survey of Clustering Algorithms, Annals of Data Science 2 (2) (2015) 165–193. doi:10.1007/s40745-015-0040-1.
URL https://doi.org/10.1007/s40745-015-0040-1

[42] B. Podobnik, H. E. Stanley, Detrended cross-correlation analysis: A new method for analyzing two nonstationary time series, Physical Review Letters 100 (8) (2008) 1–4. doi:10.1103/PhysRevLett.100.084102.

[43] J. W. Kantelhardt, S. A. Zschiegner, E. Koscielny-Bunde, S. Havlin, A. Bunde, H. E. Stanley, Multifractal detrended fluctuation analysis of nonstationary time series, Physica A: Statistical Mechanics and its Applications 316 (1-4) (2002) 87–114. arXiv:0202070v1, doi:10.1016/S0378-4371(02)01383-3.

[44] J. Parfet, Conducting and Detecting Data Exfiltration [Blog Post] (2018).
URL https://www.mindpointgroup.com/blog/operations/conducting-and-detecting-data-exfiltration/

MICHAIL TSIKERDEKIS is an assistant professor in the Computer Science Department at Western Washington University. His research interests include deception, data mining, cybersecurity, and social computing. Tsikerdekis has a Ph.D. in Informatics from Masaryk University. He is a senior member of the Institute of Electrical and Electronics Engineers. Contact him at Michael.Tsikerdekis@wwu.edu.

SCOTT WALDRON is a recent graduate from Western Washington University. He has a Master of Science degree in Computer Science. Contact him at waldros2@gmail.com.

ALEX EMANUELSON is a recent graduate from Western Washington University. He has a Master of Science degree in Computer Science. Contact him at alexeemanuelson@gmail.com.