Exploratory Machine Learning Analysis of Real Network Log Data

Brandon Carter

May 2017

Abstract

Intrusion detection systems often rely on hard checks of incoming requests to identify whether traffic is safe or malicious. Various machine learning approaches have been developed to mine large-scale network logs and help to identify anomalous traffic patterns. In this paper, we apply several machine learning approaches to real data from the MIT network. We describe how these methods could be useful both in future research and for improving network monitoring systems.

1 Introduction

When enforcing the security policy on a network, the ability to identify attacks and irregularities is crucial. Traditional intrusion detection systems rely on "signature-based" methods, in which features of traffic are compared against features of known attacks to determine if the traffic is malicious [1]. Unfortunately, these methods fall short as they are not able to protect against zero-day exploits until the attacks are detected and the master lists are updated.

Instead, researchers have turned to various methods from data mining and machine learning to automatically detect intrusions (anomalies) in network traffic. In one perspective, *supervised* anomaly detection, the algorithms are given a training set of normal data (no intrusions) and then later are given some traffic data and determine whether its patterns are "normal" or not. These algorithms typically work by learning a generative distribution of normal data, and then test whether new data would have been likely to come from that distribution. However, it is generally hard to find a training set of data that is large enough to contain a sufficient number of normal patterns but no malicious traffic.

It is also possible that the training set contains both normal data and labeled intrusions, such that the algorithm can identify particular types of intrusions after training. One such widely-studied dataset is the DARPA KDD 99 intrusion dataset (see [2]). Unfortunately, however, while this data has been the basis of much research in this area for nearly 20 years, the dataset has a variety of issues, limiting its usefulness as a training set for anomaly detection (e.g. [2], [3]). Moreover, it is likely that network traffic patterns and types of attacks have changed since the dataset was collected in 1998. Though, there is still very little other network data that is both publicly available and has a sufficient number of features to train anomaly detection algorithms, often a result of privacy concerns with exposing real traffic. It is also extremely difficult and expensive to manually curate labels of which data points are normal and which correspond to network misuse.

The problem of *unsupervised* anomaly detection resolves some of the previous problems by positing that a dataset contains mostly normal traffic but also a small amount of anomaly (i.e. malicious) traffic. [1], [4], and [5] contain examples of approaches that have been proposed to identify anomalous traffic under this assumption. These approaches are much easier to apply to real data that is difficult to label.

The goal of this project was to explore various unsupervised techniques to identify interesting patterns in real-world network data on an MIT subnetwork. Because we[1] had limited time (1 week) to explore this data, we chose to investigate it from a few broad angles that we believe inspire future work. First, we explore geographic patterns in the data. Then, we run an unsupervised anomaly detection algorithm and show that clustering reveals various usage patterns. Finally, we investigate time series patterns and discuss ways we might be able to use this data to predict spikes in traffic. While we unfortunately did not have time to explore each of these directions fully, we hope that our preliminary analysis exposes this data in neat ways.

In Section 2, we describe our dataset. Section 3 discusses our evaluation of geographical patterns in the data. In Section 4, we apply an unsupervised anomaly detection algorithm on our data and show some of the trends it discovered. Section 5 discusses opportunities for network traffic forecasting using this dataset. In Section 6, we describe several directions for future work.

Code written for this project is available in a repository at: `https://github.mit.edu/bcarter/6858NetworkLogAnalysis`, as well as in the provided 6.858 git repository.

2 Dataset

We obtained a dataset of real network traffic on an MIT subnetwork through connection logs stored in the Bro IDS.[2] We wrote a script that parses connection logs and anonymizes IP addresses to preserve privacy. We hashed IP addresses (both origin and destination) and also extracted country information using a free IP geolocation library.[3] MIT IS&T used this script to process connection logs collected in Bro.

As an aside, we realize in hindsight that anonymizing IP data using a simple hash is insecure. Since the IPv4 address space only includes about 4 billion unique addresses, it would be possible to brute-force the IP corresponding to each

[1] "we" is used as a matter of convention. Note that all work was completed individually.

[2] `https://www.bro.org`

[3] `http://dev.maxmind.com/geoip/geoip2/geolite2/`

hash. While we did not attempt to exploit this "vulnerability" in practice, we found that we could compute 1 billion IP hashes in Python in about 90 seconds, meaning that it would take around 6 minutes to find the IP corresponding to a particular hash.

We realize that privacy-preservation is extremely important when analyzing real network data so we propose several fixes for this issue. First, in newer version of Python (3.3+), it seems that the default hash function is randomized per-session, so just having a database of hashed IPs would be harder to reverse. It is also important to keep in mind that the IPv4 address space is constrained, and so there may be ways to cleverly reverse engineer the seed of the hash. Another way to increase privacy would be to salt the hashes (per unique IP) and then discard the salts after anonymizing the data. In practice, a daily connection log contained about 1-2 million entries, so it would be easy to maintain a mapping of salts for each IP. Lastly, we could map each unique IP to some other randomly chosen IP in the IPv4 address space such that observing an anonymized IP gives us no information about the original IP (it could have come from anywhere in the address space, uniformly at random). Overall, we note that even minor tasks like anonymizing logging data prior to analysis can be challenging in practice, and even harder to reason about guarantees.

In total, we obtained traffic data spanning from February 3rd to April 12th, 2017. The data includes almost 42 million log entries. For most of our analysis tasks, we work with a subset of the data from early March (chosen arbitrarily) due to performance constraints.

We extracted the following features directly from the connection logs: hashed origin IP, origin IP country, origin request bytes, response bytes, protocol (e.g. tcp, udp), service (e.g. http, ssh, telnet, ftp), and connection status indicator[4]. Then, we compute a variety of time-based features for each entry. In [6], the authors perform feature selection for the 41 features available in the KDD 99 intrusion dataset and show which are most statistically significant. We chose to compute a variety of the more prominent features. Using a two-second window before each request, we compute: the number of times the same origin attempted a connection, the portion of connections using the same service as the request, the portion of connections using other services, the portion of connections contacting the same destination machine (e.g. to indicate DoS attacks), the portion of connections contacting the same destination machine on the same service, the portion of connections contacting the same destination machine but on different services, the portion of connections using the same service but on different hosts, and the portion of connections with a "S0" flag (indicating connection attempt seen, no reply).

3 Geographical Patterns

In this section, we explore several geographical trends in our traffic data.

[4]`http://ryesecurity.blogspot.com/2012/04/solving-network-forensic-challenges_27.html` contains a description of the possible `conn_state` values in Bro IDS.

3.1 Country Prediction

We first thought it would be interesting to investigate whether there are any geographical patterns underlying the traffic data. Figure 1 shows the distribution of traffic originating from different countries (note the log scale).

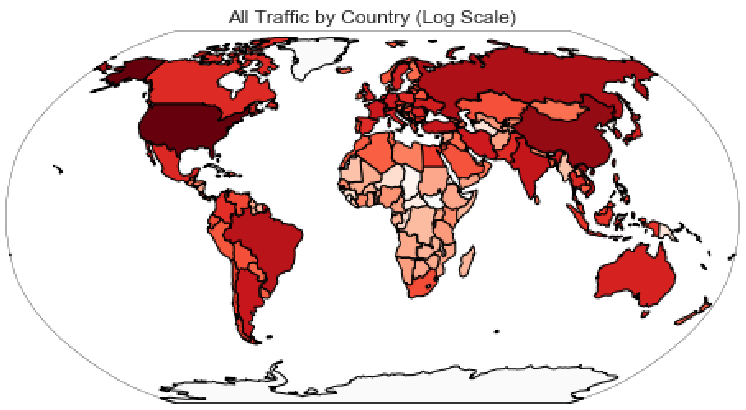

Figure 1: Heatmap of originating traffic by country. Note that intensities are computed on a log scale.

To test whether trends differ by country, we trained a classifier to predict the country based on the other features in each log entry using one day's worth of log entries (about 1.4 million total). We used a random forest classifier[5] (both due to performance concerns and presence of categorical features) and performed 5-fold cross validation (splitting the data into five "folds," and training on each set of four while testing on the unseen fifth). We found that the mean accuracy (that the country is predicted correctly) of the classifier is about 42% (though it had notably high variance—the 95% confidence interval is 42 ± 9%). In comparison, the baseline accuracy if we were to always guess the most common country (the U.S.) is about 39%.[6] Note that we did not tune the model hyperparameters for performance, and we would expect that tuning would yield a sightly better model. Still, we were surprised that the classifier was able to predict correctly a fair bit above baseline, especially given it is selecting from a set of almost 200 countries. We suspect that with larger amounts of training data the variance of our accuracy would decrease and the model could predict consistently above baseline. This result suggests that usage patterns may differ geographically (though the effect is minor).

While this conclusion may not be directly relevant from a security perspective, it does suggest that it might be useful to distinguish by country when monitoring network traffic. Traditional network monitoring (as described in

[5]in Python, using the scikit-learn machine learning library

[6]A more sophisticated evaluation metric would utilize the predicted distribution over all countries for each point in the test set, rather than just the most probable one, though using this metric seemed irrelevant for this project.

Section 1) often looks for specific usage patterns to identify threats, so this finding suggests that incorporating geographical IP data based on the origin country might be helpful. We did not have time to explore exactly how the trends differ by country, though this would certainly make for interesting future work.

3.2 Same Origin Countries

We next analyze the particular data points that have very high same origin counts (the same host had made many requests to MIT in the previous two seconds). Looking at the distribution of same origin counts in general, we find that the median number of same host requests in the preceding two seconds is zero, with mean 10.2. The 90th percentile is 25, and the 98th percentile is 112. The maximum is 998.

We look at the distribution of origin country for requests that had ≥ 112 same host requests in the previous two seconds, given in Figure 2. The top country in this distribution is China, whereas the most popular country is the U.S. when not filtering for high same-origin traffic.

The top five countries (in order) in the high-origin distribution are: China, U.S., Netherlands, Seychelles, Czech Republic. (The top five countries without filtering are: U.S., China, Russia, Taiwan, Vietnam.)

We speculate that this result occurs because high-traffic bots often are either located or use VPNs in foreign countries for better anonymity and to circumvent various legal restrictions. We also note that the relative number of countries sending high-traffic requests is quite small, though this may also be caused by only looking at a single day's worth of traffic. In the future, we would check whether similar patterns exist in data from multiple days.

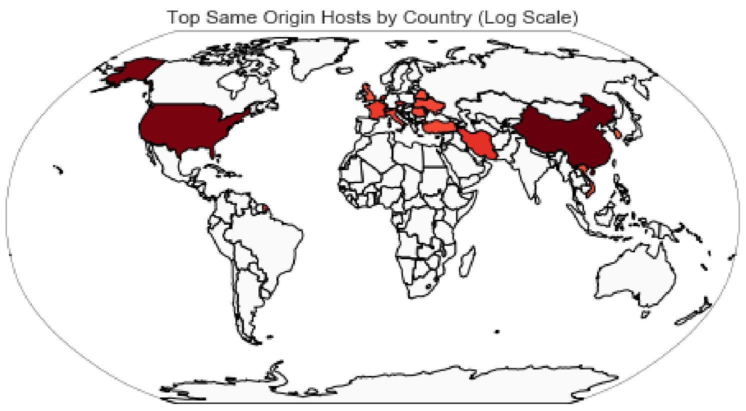

Figure 2: Heatmap of originating traffic by country for hosts with ≥ 112 requests in previous two seconds. Note that intensities are computed on a log scale.

4 Unsupervised Anomaly Detection

We next turned to explore unsupervised anomaly detection using isolation forests [5], a method developed while using network traffic data. We assume that about 1% of the data is anomalous (which seems roughly reasonable—about 99 requests are normal for every abnormal one) and specify this "contamination value" as a parameter in the algorithm.[7] After training, we are given an "normality score" for each datapoint and can identify which points (including future points) are anomalies.

We trained an isolation forest[8] using one day of logging data (more would likely be helpful, but training times would increase). We then wanted to identify if there were any particular trends among the anomalous points. First, we visualized the anomaly point space by performing dimensionality reduction using t-SNE [7] on a random subset of the anomaly points to produce two-dimensional embeddings, as shown in Figure 3. We see that there is some inherent structure to these points, so to investigate each one further, we ran standard K-Means clustering on the embeddings.[9]

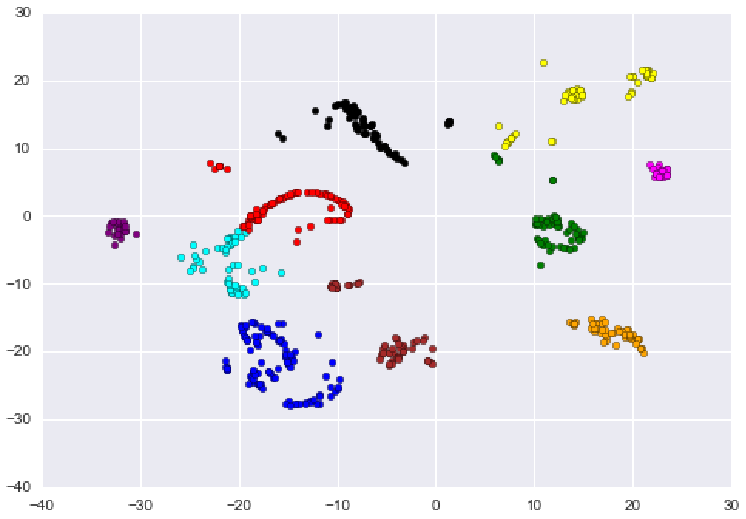

Figure 3: Subsample of anomalous data points as detected by isolation forest. Dimensionality reduction was done using t-SNE. The colored clusters were identified after dimensionality reduction using standard K-Means.

We explore the log entries corresponding to points near the cluster centroids. Interestingly, we find that the green cluster contains DNS traffic from a vari-

[7]We briefly explored other values of this parameter but found 1% to perform best.

[8]using the scikit-learn implementation

[9]Note that normally clustering is performed prior to any dimensionality reduction, though in this case, we noticed that there are clusters of points in the low-dimensional space, and we just wanted to investigate them further for the purpose of exploration.

ety of foreign countries (e.g. China, Romania, Hong Kong, United Kingdom) whose hosts were sending about 100 requests per second to MIT. We suspect that this traffic is not malicious, but rather these are DNS servers forwarding DNS requests to a server at MIT. However, since we do not have access to the destination IP (due to the anonymization), we are not able to verify whether this is the case.

The dark blue cluster contains many ssh requests from the same host (located in the U.S.) which was sending hundreds of requests within the past two seconds. The request and responses across the different requests from this host contain roughly the same number of bytes. If this is malicious, we might speculate that the host is trying to brute-force an ssh login. Though, it might also be that the user has an active ssh session which is sending many updates in real-time. The yellow cluster contains a very similar traffic pattern.

The orange cluster is a series of tcp traffic from a host in Lithuania sending about 15 requests per second. We can see based on the connection state in the log that these requests are rejected (we are not sure why), though this seems to be some sort of malicious traffic. The purple cluster contains a similar traffic pattern but mainly comprised of requests from a host in Singapore.

We see that this method is able to pick out a variety of "anomalous" traffic patterns in our data. Because we are dealing with unsupervised learning, we are unable to say whether this model is detecting potential attacks since we do not have a labeled set of known attacks in the data. We believe that in the future, with sufficient training data, it would be possible to use this method in real-time in conjunction with other intrusion detection systems. For any incoming request, it is quick to extract the input features and use the trained model to predict how anomalous the request is. This approach is advantageous because it might improve detection of zero-day vulnerabilities before other monitoring systems discovered the new traffic pattern. There could be an issue with false-positives (as we found with some of the clusters discussed above), but if this anomaly score is combined with other features from existing intrusion detection systems, we could mitigate this concern (e.g. if the other systems are confident the request is safe, and this method is not overly confident it is anomalous, we don't trigger an alert). Furthermore, we believe that using more training data as well as properly tuning the contamination parameter would lower the number of false-positives.

We also compare the geographic origin of inlier and outlier traffic based on this model. Figure 4 shows the relative distributions of origin for inliers (left) and outliers (right). Note that while it appears that the outlier origin countries are much sparser, there are many fewer outlier points than there are inliers. Because of this effect, we aren't able to make any conclusive statements about whether these distributions are dissimilar.

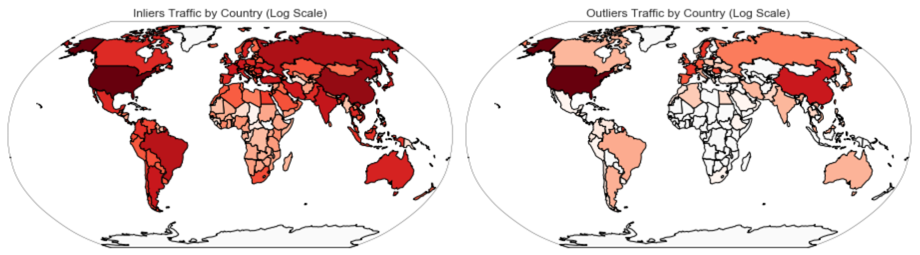
Inliers Traffic by Country (Log Scale)
Outliers Traffic by Country (Log Scale)

Figure 4: Heatmap of traffic origin for inliers (left) and outliers (right) as determined by the isolation forest model. Note that intensities are computed on a log scale.

5 Time series Predictions

We now discuss trends in the time series of connections to this MIT subnet based on the network logs. We binned traffic data to count the number of connections per minute and per second over the course of one day, shown in Figure 5. We see in the per-minute (left) plot that traffic frequency peaks in the middle of the day, as we expect since the U.S. sends the greatest amount of traffic to these servers, compared to other countries. Interestingly, we note that it would be harder to identify this pattern from the per-second plot (right), suggesting that the amount of traffic on a per-second basis is more uniform compared to binning on a per-minute basis. The high-traffic spikes are more concentrated in the mid-day region in the per-second plot, such that when binning into minutes, we can observe that traffic mid-day is generally higher than morning and evening/overnight.

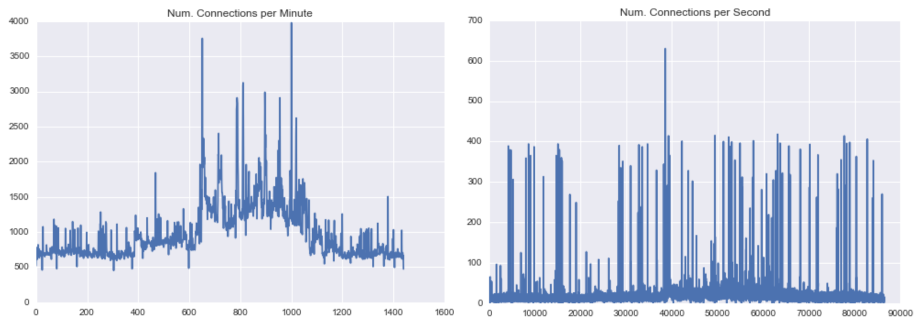

Figure 5: Time series plots of number of requests per minute (left) and per second (right) over one full day.

We think an interesting machine learning task on this data would consider traffic data from the previous n timesteps (e.g. minutes) and predict the amount of traffic at time $n + 1$. Such a model would be useful to help predict attacks involving high-traffic just as they are starting to occur by predicting the impending traffic "ramp-up." It could also be helpful for dynamic server resource management.

We attempted to train a simplistic regression model in which feature vectors contain the number of requests for each of the last five minutes and the target is the number of requests for the sixth minute. We generate features using a sliding window over the binned data. We used a standard ordinary least squares regression model and 10-fold cross validation to find that the mean coefficient of determination[10] (R^2) is 0.12 (with a 95% confidence interval of 0.12 ± 0.65). In certain cases, the model performed very well on test data, and in others, performed very poorly. We explored using different numbers of previous timesteps in the feature vectors (e.g. past 10 or 20 minutes rather than past five minutes). We also used data at the second-level rather than minute-level, thinking that perhaps binning at the minute level is too coarse and misses second-level trends in traffic. However, we were not able to significantly improve the accuracy of the model in these ways.

After further exploration of this time series data, we believe that the traffic jump trends are too noisy/random to perform the task in this manner (we might observe the same trends over the previous few minutes but then have different results at the next minute). We believe that more sophisticated feature engineering would improve the prediction model by looking for other underlying trends that might be indicators for impending traffic increases and decreases. Such features could include various moving averages taken over the previous time points, rate of change analysis, etc. However, trying to engineer features to improve this model could likely serve as a whole separate project on its own!

Another direction to model this time series data would be using recurrent neural networks (RNNs), which can find underlying, perhaps non-linear, trends in the data without explicitly specifying these functions. Much similar work has been done using RNNs in financial forecasting and other time series, for example, where the observed data are often noisy, non-stationary, and have non-linear dependencies (e.g. [8], [9]). Neural networks allow for more robust modeling compared to conventional methods. Unfortunately, we were not able to explore using neural networks for traffic pattern modeling, though we believe that it would be a promising future direction.

We also identified a variety of published methods on the problem of network traffic forecasting (see [10] for a review of some common approaches). Some work (e.g. [11]) has been done showing that neural networks seem to work well on this problem. The authors of [11] discuss that internet network traffic contains some statistical features (self-similarity and non-linearity) that cannot be captured by classical models, confirming the difficulties we faced with our simplistic attempt at traffic prediction.

[10]the closer R^2 is to 1, the better the fit of the model

6 Conclusions and Future Work

In this paper, we explore using machine learning methods for several tasks on a dataset of network traffic logs from an MIT subnetwork. From this work, we reached several conclusions. First, we note that running unsupervised algorithms on real-world (noisy) data is challenging. It is particularly difficult to evaluate results (i.e. identify attacks) without labeled data. Instead, we show that we are able to identify various traffic patterns in the data. Second, we found that details like privacy-preservation can be tricky. We realized that hashing IP addresses does little to provide anonymity in network logs. Finally, we identified several methods that could be useful components of an intrusion detection system, such as an unsupervised anomaly detection model that can identify zero-day threats, as well as a traffic monitoring model that can predict future levels of traffic. While these models would take significant refinement before they could be used practically, we believe that our exploratory results and discussion enlighten their potential.

We describe several opportunities for future work. First, we were subject to performance and time constraints. It would be helpful to retrain these models with significantly more training data and also compare results across different days to see if the traffic patterns are similar. In addition, more careful feature engineering might help the learning tasks. We chose a subset of features used in the KDD 99 dataset and elected to use the same two-second window for computing time-based features, though other features might also be useful. Finally, one may want to consider these problems in a *semi-supervised* setting, perhaps learning about attack traffic using some labeled data (e.g. KDD 99) but then applying it to this larger amount of unlabeled data.

Acknowledgements

We would like to thank Prof. Zeldovich for guidance on this project. We would also like to thank Jessica Murray and Mark Silis at MIT IS&T for gathering and contributing the dataset used in this project.

References

[1] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, “A geometric framework for unsupervised anomaly detection,” in *Applications of data mining in computer security*, Springer, 2002, pp. 77–101.

[2] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the kdd cup 99 data set,” in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, IEEE, 2009, pp. 1–6.

[3] M. Sabhnani and G. Serpen, "Why machine learning algorithms fail in misuse detection on kdd intrusion detection data set," *Intelligent data analysis*, vol. 8, no. 4, pp. 403–415, 2004.

[4] K. Leung and C. Leckie, "Unsupervised anomaly detection in network intrusion detection using clusters," in *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*, Australian Computer Society, Inc., 2005, pp. 333–342.

[5] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, IEEE, 2008, pp. 413–422.

[6] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "Selecting features for intrusion detection: A feature relevance analysis on kdd 99 intrusion detection datasets," in *Proceedings of the third annual conference on privacy, security and trust*, Citeseer, 2005.

[7] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[8] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 240–254, 1994.

[9] C. L. Giles, S. Lawrence, and A. C. Tsoi, "Noisy time series prediction using recurrent neural networks and grammatical inference," *Machine learning*, vol. 44, no. 1, pp. 161–183, 2001.

[10] H. Feng and Y. Shu, "Study on network traffic prediction techniques," in *Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on*, IEEE, vol. 2, 2005, pp. 1041–1044.

[11] G. Rutka, "Neural network models for internet traffic prediction," *Elektronika ir Elektrotechnika*, vol. 68, no. 4, pp. 55–58, 2015.