Fraudulent Resource Consumption EDoS Attack Detection

1st Darpan Shrivastava
Institute of Cybersecurity and Privacy
University of Georgia
Athens, GA
write2darpan[at]gmail[dot]com

2nd Hunter McGarity
dept. of Computer Science
University of Georgia
Athens, GA
hunter.mcgarity[at]uga[dot]edu

3rd Saurabh Vijay Surana
dept. of Computer Science
University of Georgia
Athens, GA
iakonrelic7[at]gmail[dot]com

***Abstract*—Fraudulent Resource Consumption (FRC) attacks are characterized by an attacker's sustained use of cloud services over a variable length of time. When a cloud service client operates on a pay-by-resource model, such sustained traffic offers no economic benefit to the client, but rather deals economic harm to them by increasing their costs in continued cloud service usage. For this reason, an FRC attack is designated as an Economic Denial of Sustainability (EDoS) attack. Given that FRC attacks are comprised of low-intensity HTTP requests from the attacker, the attack itself differs in intent from legitimate user traffic, but is virtually identical in content. For this reason, a sophisticated detection method is required to adequately detect FRC attacks and prevent them from causing substantial economic harm to cloud service clients. Such an approach is explored in this paper, wherein LSTM deep learning neural networks are used to detect FRC attacks with a sufficient success rate.**

***Index Terms*—FRC Attack, EDoS Attack, and LSTM neural network.**

I. Introduction

To understand the harm that Fraudulent Resource Consumption (FRC) attacks can cause, one must first understand the FRC attack in nature. Many Cloud Service Providers (CSP), such as Amazon Web services, AT&T, or IBM offer a payment model commonly called "pay-by-resource" wherein the Cloud Service Client (CSC) is billed according to the amount of cloud server resources their operations consume. This means that the more users/traffic a CSC experiences for their services, the more they are charged regularly by their CSP. Such a model is logical and efficient, providing the opportunity for smaller, less-funded companies to get their foot in the door with cloud computing. However, it opens up the opportunity for an FRC attack wherein the attacker, for reasons known only to themselves, can deliberately cost the CSC a much higher sum for their cloud service resource use.

When a legitimate user of a CSC's service (such as a business' website) makes an economic action, like purchasing an item online, then the CSC earns a small amount of revenue from that transaction. With enough legitimate users making purchases or viewing ads on the website, the CSC actually earns a profit against the cost to host their site with the CSP. But when a user does nothing to earn the CSC revenue, the CSC is still charged for the users' site traffic by the CSP. An FRC attack is carried out by sending a "low and slow" stream of HTTP requests to the site such that the attack traffic, viewed externally, is virtually indistinguishable from legitimate user traffic, the primary difference being that the attack traffic takes no action to earn the CSC any revenue. Rather than attacking from a single machine, FRC attackers will typically employ a botnet to carry out the attack traffic.

If sustained over a lengthy period of time, such an attack can result in the CSC being charged much higher rates for the "user traffic" from the FRC attack. This of course can put enormous financial strain on the CSC. Thus, an FRC attack, classified as an EDoS attack, differs from DDoS attacks in that the only party negatively affected by the attack is the CSC itself, rather than other legitimate users.

II. Motivating example

The choice of pursuing such a type of attack originated while working on idea of a network flow analysis tool for the duration of an LDoS(low rate DoS) attack. Taking into account that the "cloud services" have become the defacto standard for hosting sites, accessing stored data due to instant availability, faster and consistent network speeds and latency, ability to accommodate very high user requests while also blocking attack on a cloud consumers site for DDoS attacks. Using a cloud hosting service the cloud consumer doesn't have to worry about resource allocation policies, running out of storage or computing power and such features of the Cloud hosting services opens the path to an Fraudulent resource consumption attack. Abusing these features of the cloud utility pricing model(pay-as-you-go) the cloud consumer can be a victim of extortion, hacktivism, ego, corporate rivalry. As per [10] the popularity of FRC attack will continue to grow as the growth in cloud services grows due to the facts stated earlier.

The cloud services providers have made it very easy to host a website on their infrastructure and they handle the resource allocation and offer high scalability. Having a utility pricing model similar to the electricity consumption model at our homes where we pay for only the electricity we've used. Similarly whenever the web servers service a request the cloud consumer charge them per request. The income of the cloud consumer comes from the adverts displayed, sale of items but a bot only with the purpose of sending requests will perform no such activity and thus incur a charge for

TABLE I
AMAZON EC2 PRICING MODEL AS OF MAY 2022 [7]

Data Usage	Pricing per GB
First 100GB	Free / \$0
First 10TB	\$0.09
Next 40TB	\$0.085
Next 100TB	\$0.07
More than 150TB	\$0.05

the services rendered but no income is generated from those services. In the following example it will be demonstrated how easily someone with the intention of attacking someone with the intentions of an FRC attack is possible.

A. The Cost Burden and Threat

The methodology of conducting a FRC attack is similar to how DDoS attacks are generated towards a victim host [8]. A botmaster(the bad intention-ed attacker) can setup a modest size botnet of ten thousand to hundred thousand bots with a simple malware program distributed through software, phishing scams and downloading infected software onto the several victims computers due to the lack of cyber security threats of the general masses. The botmaster then can purpose each such bot to randomly send just 200 requests per day to the host website. This case is the minimum standard compared to the massive reach of the internet and the wide consumer base for attractive sites making good buck. This kind of attack strategy has no impact or cost to the bot master. Taking the medium response size of a http post request to be around 2048KB, the cloud service consumer/host of the website will incur an additional cost of ten thousand dollars for no income earned through services rendered. Thus it is very clear how simple it is for someone with the malicious intentions to overburden a business consumer with compounding effects from such attack scenarios.

B. Our contribution

For the purpose to test FRC attack detection scheme with LSTM neural networks we used a running log from [10], an Austrian property rental site. The log is in Apache CLF and also a running log open for public access. Due to this we had to remove some anomalies like fuzzing attacks by a keyword search for the traces of these fuzzing attacks and filtering them out. The site received around 573 requests per day for a period of one and half year. This was around 5 percent of site traffic compared to the reference NASA webserver logs [11]. Detecting a marginal increase in web traffic to a popular site is vital in case of an FRC attack, which is sustained over several weeks would be an possible case of FRC attack. Our data was very sparse in case of the less popular pages on the website. As a result we focused on developing a single model for such data with heavy requests on the populous pages and almost nothing for the remaining 80 percent of the pages. Thus in our case a single model suffice with almost 8 additional requests made per hour during the attack period and we had accuracy around ninety percent for different attacks in different weeks.

III. TECHNICAL DETAILS

This section outlines the various technical steps that were taken to successfully detect FRC attack.

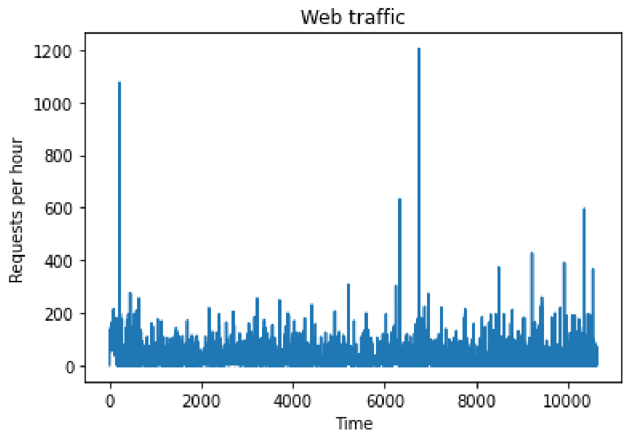
A. Pre-processing of Web Server Logs

In order to detect FRC attack, web server logs were obtained. The logs were then preprocessed in order to remove inconsistencies such as fuzzing and brute force attempts. After the log file was cleaned, the web-server logs are then parsed to create a time series sequence of the total number of requests per hour on a collection of web pages for H hours. The H hours are assumed to be attack free and describe the general pattern of web activity for the website.

Total pages on the website are taken to be N. Let f_{ij} represent the total number of requests on i^{th} page in the j^{th} hour. Then F_i represents the total number of requests for the i^{th} page over total time H.

$$F_i = \sum_{j=1}^{H} f_{ij}$$

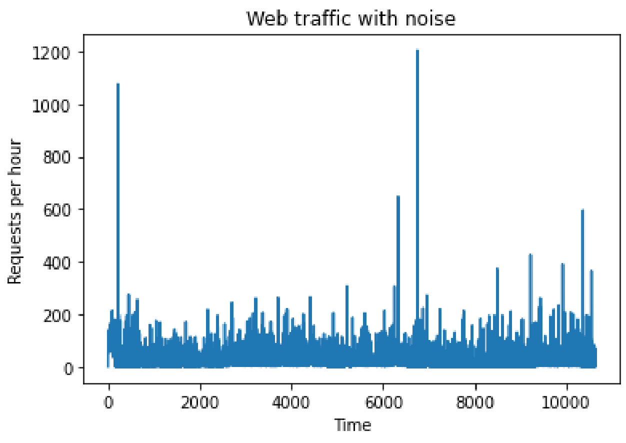
All such F_i are arranged in decreasing order and then combined together to form one time series sequence.



The figure above depicts the number of requests over total number of hours H.

B. FRC Attack Simulation

After the generation of the time series sequence, FRC attack is simulated and added to the sequence. For noise, the mean and variance of the entire data set was obtained and was multiplied to the attack percentage and calculated the normal distribution for those hours. Attack percentage of 5% means that the attack traffic is 5% of the usual traffic. The first H/2 of this sequence is kept attack free and attack is simulated on the remaining part. The period from $9H/10^{th}$ hour to H^{th} hour is kept free again. This ensures that the sequence contains both type of transitions- traffic switching from normal to attack and vica-versa.



The figure above depicts the web traffic with added noise. It is interesting that we cannot see any difference in each of the respective graphs. The traffic associated with FRC easily masks with legitimate traffic.

C. Target Sequence Generation

For training the model to detect attack and non-attack hours, class labels need to be provided to the neural network. A *Targetsequence* is generated which assigns a target value to each hour representing whether noise has been added to that hour or not. The target value t_j for the j^{th} hour of Target sequence is denoted as,

$$t_j = \begin{cases} 0 & \text{if No attack noise is added in } j^{th} \text{ hour} \\ 1 & \text{if Attack noise is added in } j^{th} \text{hour} \end{cases}$$

Hour for which $t_j = 1$ is referred to as an anomalous hour and hour for which $t_j = 0$ is called a normal hour. This sequence is ready to be fed to the neural network. Lastly, a transformation step has to be applied to this sequence for de-noising and removing any transient anomalies.

D. 1-Dimensional Discrete Wavelet Transform

These sequences still have a lot of inconsistencies in the form of sudden peaks and troughs. As these sequences will be used to train machine learning based model later on, it is better to remove such inconsistencies and have a smooth trend of ups and down in our sequences.

The task of removing the high frequency components in the sequence is done using a One dimensional Discrete Wavelet Transform on the sequences. The 1D DWT removes all the anomalies which helps train a consistent machine learning model. The advantage of an discrete wavelet transform is to keep the spatial and temporal characteristics of the input sequences while getting rid of the high frequency anomalies. The Discrete wavelet transform used several layers of high and low pass filters to achieve this. Like a Fourier transform the DWT uses a wavelet and in this case wavelet of the family Daubechies with level 4 was used for empirical reasons. The filters on each step give an orthogonal set of vectors where the output of the high filter pass are null-ed and added with the low pass filter's output. This is done for several times until a smooth and consistent timeseries sequence is generated. With this step the data is preprocessed and ready for supplying to the machine learning model.

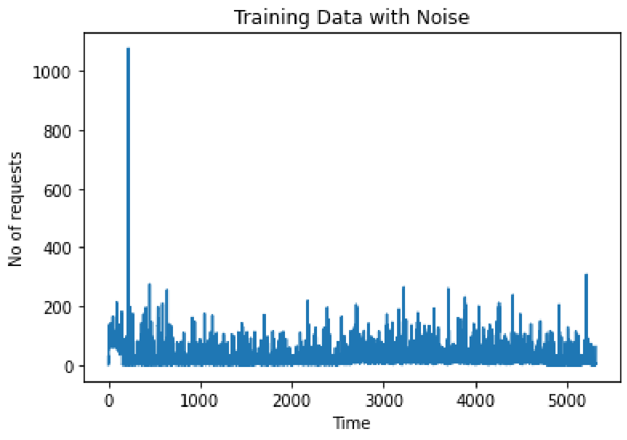
E. Training LSTM Network for Classification

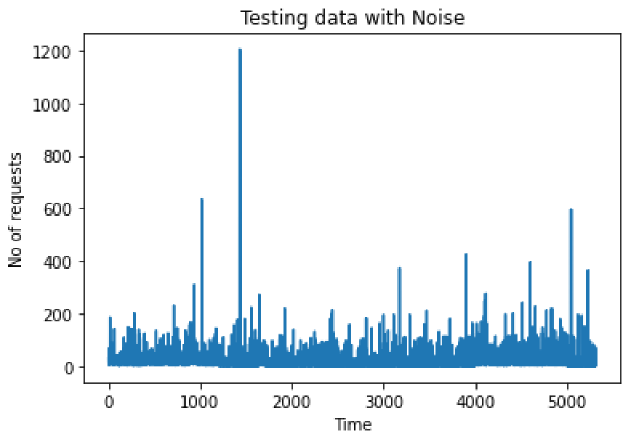
LSTM network is a mathematical model, capable of learning correlation and finding patterns from the given data. It can learn long-term dependencies and is widely used for time series anomaly detection. FRC attacks run for long time duration and hence LSTM network is the appropriate choice for detecting them. LSTM network is engineered to remember a long history of normal web traffic and detect the deviations caused by the attack.

LSTM networks allow connections between the neurons withing the same hidden layer. This allows for historical input information to be stored in the network's internal state, and these are thereby capable of mapping all of the historical input data to the final output.

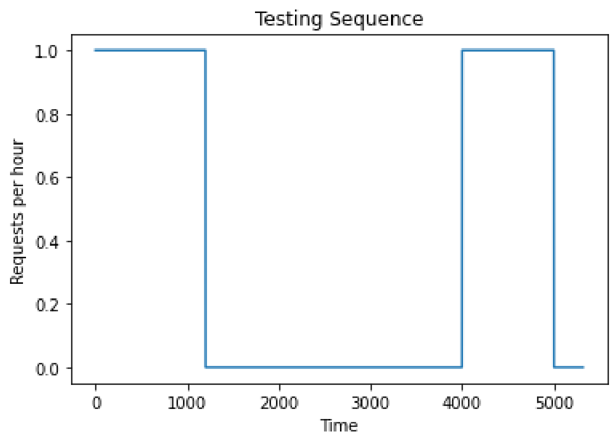
IV. Evaluation

For evaluation, the entire data set was divided into training and testing data with added noise.

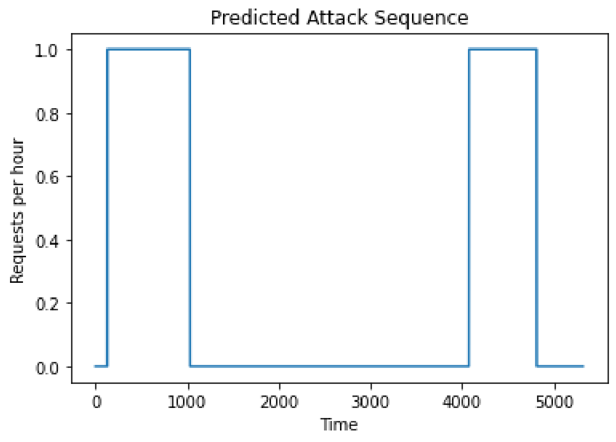




After training the data set on the LSTM network, we are now ready for prediction.

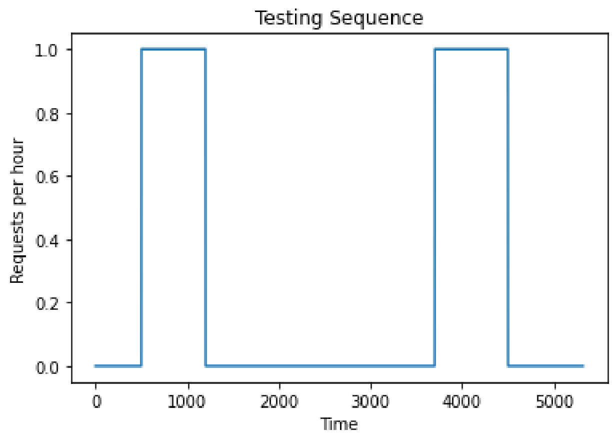


The above figure is the testing sequence for our first experiment.

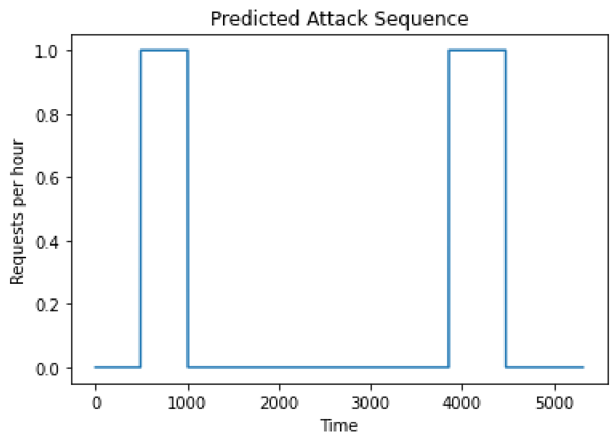


Comparing the testing and prediction sequence, we obtain an accuracy of 91.44%.

For our next experiment we change the position where the noise is added to see how the neural network reacts to it.



The above figure is the testing sequence for our second experiment.



Comparing the testing and prediction sequence, we obtain an accuracy of 92.34%. Hence, we are able to detect FRC attack with a sufficiently high accuracy.

V. Related work

A major contribution for the detection of FRC attacks comes from the works of [1], who've used the P-estimation scheme to detect and subvert an FRC attack. The P-estimation technique is an selection technique where the attack percentage is calculated with respect to the Frequency of least popular pages on the victims website to the current value of the same. This assumes that the data being used for the reference is attack free and is being used as a standard for such calculations. This makes use of the violation of Zipf's law where an attacker isn't privy to the popularity of the web pages and thus the attack will increase the requests to the least popular pages and the percentage increase in such a case is used to estimate the P parameter. On the basis of this P parameter several models are trained for P varying from 2 percent to 25 percent increase in the web activity of the web server.

The researchers have used the web server logs of an Apache based web server for NASA website. The logs have around 3.5 million requests for 1369 hours. The data is assumed to be attack free. It is then preprocessed in 50:50 for training and testing purposes. For both of these data-sets several quantiles are made for the number of requests per hour to a particular web-page. The selection of the number of quantiles is a sensitive parameter. The total number of pages on the website are taken and divided into the quantiles. The first quantile contains the web-pages with the maximum number of requests and the final quantile has the requests per hour for pages with the least amount of requests. The data is of type time-series and every consecutive data point in the quantile has the total number of requests from all the pages in that quantiles for the total number of hours. These results in sequences equal to the number of quantiles and the mean and variance of the sequences is calculated to for generating appropriate amount of noise.

The noise is introduced in 40 percent of the total data hours from 50 percent to 90 percent and the first fifty percent and last ten percent of the sequences are attack free. The attack simulated by noise is generated in accordance to normal distribution with mean and variance parameters calculated from the sequences previously. As a FRC attack is a Low rate DOS attack the mean and variance of the noise is multiplied by the attack percentages ranging from two percent to twenty percent so the noise only marginally increase the requests to the site. This method gives us several models with different attack percentage which will later be used to detect attacks on the basis of the P (attack percentage) parameter.

The machine learning model used for this application is a neural network. Long short term memory (LSTM) neural network is a type of RNN(Recurrent Neural Network) with the addition of a history and forget gate the cells of the neural network. This allows the model to learn from previous data. This network is trained to classify the data of the input sequences to the target sequence. The target sequence is a sequences which denotes where the attack is induced into our data. For the case when no attack noise is added to the input sequences the target sequence has the value zero and one elsewhere. This is a problem of binary supervised learning which the LSTM neural network will be performing.

The loss function used in this case is the Binary Cross Entropy which the neural network will try to minimize through training. Several such models are trained on the basis of the attack percentage (P). After the model has been trained with hyper-tuned parameters the models can be used for the detection scheme. Detection scheme calculated the violation of Zipf's law in terms of the attack percentage and the appropriate models is used to detect if an attack was attempted or not. The models captures the trend of requests for longer periods of times to have an accurate prediction. For every hour that is calculated as under attack if more than 50 percent of such consecutive hours are under attack then the week is said to be under attack for such an threshold amount. In the case the hours are attack free it the new data is used to retrain the model for changing trends of web page popularity. This helps keep the model updated for dynamics change in the websites popularity trends which results in very little false positive attacks.

After an attack is detected for a a particular week the web server logs for those attack hours is used to extract the requests and IP's of the user. A whitelist, greylist and blacklist is generated which gives puzzle challenges to those users, thus not completely denying the use of the website to a particular users abnormal usage. As the only difference between the FRC attacker and normal user is the intentions no harsh policies are used to filter out the user base unlike in the case of the DDoS attack.

The predecessor of the previous approach [2], Rustogi et al. uses Artificial neural networks instead of the LSTM networks. The performance in this approach was significantly increased to capture attacks of very low intensity over a longer undetected period. In an other approach [3] the author calls for the reviewing of the financial bills of the CSC for anomalous increase in the bills while the website earned no such profit or additional income during the particular billing cycle. There also have been several attempts of FRC detection by finding the statistical outliers on the basis of usage characteristics in [4] but these models rely on high computing resources for tracking usage characteristics of each user which is not practical for popular websites. In another approach, [5] it was proposed that the bots on a webpage can be differentiated from the human user on the basis of the time spend and user activity. But considering only a single parameter, the time spent is not a good metric to filter out the bots as shown in [6].

VI. Conclusion and Future Work

In this work, the definition and potential harm of a successful FRC attack are explored. Since an FRC attack mimics legitimate user traffic with seemingly harmless HTTP requests over time, it is difficult to detect with the methods that are used to detect a DDoS attack. An FRC attack is called an Economic Denial of Sustainability attack because its target is the economic well being of the Cloud Service Client that is operating on a pay-by-resource model from their respective Cloud Service Provider. If allowed to run for an extended period of time, an FRC attack can cause serious economic harm for the CSC which could indirectly impact other legitimate users of the site.

Further, we explored a general overview of Neural Networks and their role in data prediction. Specifically, this work discusses the differences between an LSTM network and a Recurrent Neural Network with an emphasis on the advantages offered by the former. LSTM networks are excellent for learning long-term time dependencies due to their design around long-term memory. This makes them well-suited to the detection of FRC attacks, which take a long time to become detectable. For development, training, and deployment of our LSTM networks, we used the Keras API in Python, which is built on top of the TensorFlow library. This API was chosen on account of its ease of use, extensive documentation, and design around rapid deployment.

Input for the LSTM model training was taken from thousands of 200 OK HTTP requests from online web server logs. We then simulated FRC attack noise with a Gaussian Noise Function before de-noising with Discrete Wavelet Transform. Once the LSTM model was trained, we were able to successfully detect FRC attack traffic with a favorable success rate without the use of parameter hyper-tuning.

Our work is focused on LSTM-enabled detection of FRC attacks, but there are other paths to take in combating FRC attacks. Once an FRC attack has been detected, the CSC needs to find some way to mitigate the attack. FRC mitigation is an important research topic that we could explore in future work, and would be focused on eliminating the economic harm done by an FRC attack once it has been detected. There are other FRC mitigation techniques that other researchers have proposed. In [1] Agarwal, et al. proposed a mitigation scheme that is built around the concepts of IP whitelist, blacklist, and greylist for "containing" FRC attacks. The scheme also relies on presenting the site user with human challenges akin to CAPTCHA tests to filter through bot devices that may be under the direction of the FRC attacker. While this mitigation technique has its advantages, it also carries the significant drawback of using human challenges, which dramatically impact the user experience on the site.

Another approach suggested by Idziorek et al. [3] is focused on user statistics and flagging users as anomalous if they do not meet the standards of said statistics. An apparent downside to this approach is a high false positive rate because legitimate users can proliferate different traffic patterns on the site depending on their browsing habits, without necessarily instigating an attack. A good direction to take for expansion of this work would be development of an FRC mitigation technique that yields a low false positive rate while also prioritizing the user experience by avoiding human puzzle challenges.

Our LSTM model is called a *stacked LSTM*, meaning that multiple hidden layers make up the model as a whole. This is what separates our LSTM model as a *deep learning* model apart from a standard single-layer LSTM. A stacked LSTM offers better input abstraction than a single-layer LSTM. As per the specifications given in the experiment detailing in [1], our model uses an input layer followed by three hidden layers and finally, a Dense output layer. Each layer has 4, 512, 256, and 64 neurons, respectively. The Dense output layer also has a single neuron. With this model, we were able to attain favorable success rates at FRC attack detection, but our process of training did not include parameter hyper-tuning. LSTM parameter hyper-tuning is a process by which we first specify different model parameters to tune, then run an automated method of compiling a model, training a model, and testing a model, each time with different tuned parameters. This is done in order to find the "ideal" set of parameters. The team worked to learn hyper-tuning via the Keras Tuner, which allows tuning on the metric of data loss or prediction accuracy.

Ultimately, hyper-tuning did not yield favorable results for our work, as the highest accuracy rating we attained for a hyper-tuned model was approximately 87%. The exact cause behind this is unknown, but future modifications to the LSTM model, perhaps in the development of an attribution/mitigation technique, could open the door for effective parameter hyper-tuning.

Acknowledgment

We extend our gratitude to professor Kyu Hyung Lee in the Department of Computer Science at the University of Georgia for guidance and feedback throughout the project's conceptualization and realization.

References

[1] Abhishek Agarwal, Ayush Prasad, Rishabh Rustogi, Sweta Mishra, Detection and mitigation of fraudulent resource consumption attacks in cloud using deep learning approach, Journal of Information Security and Applications, https://doi.org/10.1016/j.jisa.2020.102672.

[2] Rustogi R, Agarwal A, Prasad A, Saurabh S. Machine learning based web-traffic analysis for detection of fraudulent resource consumption attack in cloud. 2019 IEEE/WIC/ACM international conference on web intelligence (WI). IEEE; 2019. p. 456–60.

[3] IdziorekJ,TannianMF,JacobsonD.The insecurity of cloud utility models. ITProf 2013;15(2):22–7.

[4] IdziorekJ,TannianM,JacobsonD.Attribution of fraudulent resource consumption in the cloud. 2012 IEEE Fifth international conference on cloud computing. IEEE; 2012. p. 99–106.

[5] Koduru A, Neelakantam T, et al. Detection of economic denial of sustainability using time spent on a web page in cloud. 2013 IEEE international conference on cloud computing in emerging markets (CCEM). IEEE; 2013. p. 1–4.

[6] Neal A, Kouwenhoven S, SA O. Quantifying online advertising fraud: ad-click bots vs. humans. Tech. Rep. Oxford Bio Chronometrics; 2015.

[7] Amazon EC2 pricing model https://aws.amazon.com/ec2/pricing/on-demand/

[8] Botnet-based Distributed Denial of Service (DDoS) Attacks on Web Servers: Classification and Art. (2012)

[9] Bawa PS, Manickam S. Critical review of economical denial of sustainability (EDoS) mitigation techniques. J Comput Sci 2015;11(7):855.

[10] http://www.almhuette-raith.at/

[11] https://data.world/shad/nasa-website-data