Image Processing for Detecting Botnet Attacks: A Novel Approach for Flexibility and Scalability

Aurélien Agniel, David Arnold, and Jafar Saniie

Embedded Computing and Signal Processing (ECAP) Research Laboratory (http://ecasp.ece.iit.edu/)
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago IL, U.S.A.

***Abstract*- Continued adoption of the Internet of Things (IoT) redefines the paradigm of network architectures. Historically, network architectures relied on centralized resources and data centers. The introduction of the IoT challenges this notion by placing computing resources and observation at the edge of the network. As a result, decentralized approaches for information processing and gathering can be adopted and explored. However, this shift greatly expands the network footprint and shifts traffic away from the center of the network, where observation and cybersecurity monitoring tools are frequently located. Further, IoT devices are often computationally constrained, limiting their readiness to deal with cyber-threats. These security vulnerabilities make the IoT an easy target for hacking groups and lead to the proliferation of zombie networks of compromised devices. Frequently, zombie networks, otherwise known as botnets, are coordinated to attack targets and overload network resources through a Distributed Denial of Service (DDoS) attack. In order to crack down on these botnets, it is essential to develop new methods for quickly and efficiently detecting botnet activity. This study proposes a novel botnet detection technique that first pre-processes network data through computer vision and image processing. The processed dataset is then sent to a neural network for final classification. Two neural networks will be explored, a sequential model and an auto-encoder model. The application of image processing has two advantages over current methods. First, the image processing is simple enough to be completed at the edge of the network by the IoT devices. Second, preprocessing the data allows us to use a shallower network, decreasing detection time further. We will utilize the N-BaIoT dataset and compare our findings to their results.**

Keywords: IoT, Edge Computing, Computer Vision, Machine Learning, Auto-encoders, Cybersecurity, Botnets

I. Introduction

The number of Internet of Things (IoT) devices will continue to grow worldwide in the coming years. This growth in connected devices will result in an increase in attack surfaces, and more importantly a rise in the number of existing botnets. IoT devices range from sensors to cameras, leading to a diverse range of computational resources and embedded designs. Further, they are often not powerful enough to host commonplace malware detection toolkits. When compromised, attackers can use these IoT devices as launching and entry points into larger corporate or personal networks. Additionally, they can be reorganized into botnets to launch Distributed Denial of Service (DDoS) attacks on public and private networks [1-3]. For example, the Mirai botnet successfully infected 2.5 million devices within the last quarter of 2016 and further used them to launch targeted DDoS attacks [4]. In addition to DDoS, compromised medical or safety devices can be used to critically injure the user, such as using a pacemaker to deliver a deadly jolt of electricity to the patient. These threats require a shift in defense and protection methods as IoT devices are often more exposed compared to other resources [5, 6]. Real-time detection of attacks is needed to disconnect and secure infected devices before they can spread the botnet or cause lasting damage to the network.

Detection of botnets, such as the Mirai, Hajime, Reaper, and Gafgyt botnets, can be accomplished by using Artificial Intelligence to classify network traffic datasets. For example, the popular N-BaIoT dataset was analyzed using an auto-encoder to detect when a botnet attack was occurring [1]. As an alternative, we propose the introduction of a computer vision pre-processing phase prior to using AI tools. Currently, the use of computer vision in cybersecurity is used to detect hidden malware using binary visualization and can also be used to identify phishing websites [7, 8]. For instance, the graphical representation using a binary visualization technique can highlight clusters of information differentiating benign files and hidden malware. This was used to achieve 99% accuracy for detecting malware in PDFs [7]. Extending these ideas, we will convert the network traffic into an image, process the image, and then feed it into our AI tools. This procedure can be seen in Figure 1. Existing solutions need to

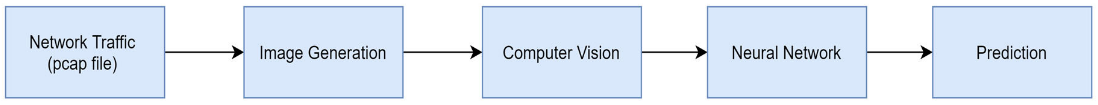


Fig. 1. Proposed botnet attack detection procedure. Network traffic is converted into an image and pre-processed using computer vision. Then it is fed into the desired neural network for predicting whether a botnet attack is occurring.

be trained for each IoT device and need to be trained again when new botnets are discovered. Our solution is advantageous as it produces a shallower network and is transferrable between IoT devices. This grants our model **greater flexibility**. Further, since the image creation is completed at O(1) and the image processing techniques allow us to use shallower networks, we can run our models at the edge of the network on embedded devices such as the Jetson Nano. This affords our model **scalability** when compared to a centralized server structure. As explained above, the profile of IoTs and their uses are numerous. The same sensor will not act in the same way when placed in a home or in a company. This is a "**predictability issue**". The main purpose of this paper is to propose a novel detection method which does not suffer from those issues and that aims to be at least as accurate as the existing ones. We will see that using image processing for botnet detection translates a traffic flow monitoring problem into a pattern recognition issue.

The transformation of the data flow and extracted features into an image allow us to use down sampling filters. We can compress an image without both losing its main characteristics and flooding the network. This can be crucial in bandwidth limited networks. Moreover, to handle multiple IoTs in the same subnet, we can install decision nodes, each containing a trained neural network model per IoT devices or IoT groups. Such a decision node can be seen in Figure 2. This node will act as a kill switch and disconnect the device from the network if it considers it compromised. Indeed, a typical router would not be able alone to both handle that many devices as well as run the neural network checks for each communication. In that case the image compression becomes interesting as it can be dynamically changed according to the context of the traffic flow.

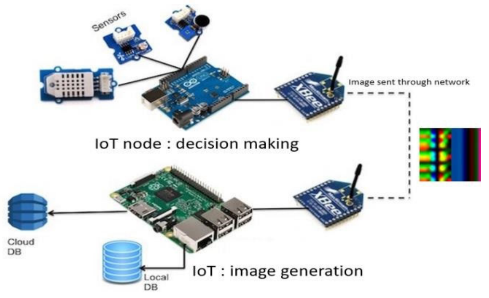


Fig. 2. Scalability: Image processing decentralized from the decision node

The traffic packets would depart from the IoT device and arrive to the router. We extract the features and transform those into an image. We check the load of the network. If the network is overloaded, we wait for more packets to populate our image. The neural network in our case asks for a fixed image input, we thus dynamically implement the compression of our image according to the traffic load. We then compress the image which still contains the overall context of the communication. We then send that image through the network to the decision node. The node will receive the image, run the neural network and decides whether to disconnect the IoT device from the network or not. This model can be adapted to the needs of most networks and be dynamically changed which makes it very flexible. Moreover, because or neural network are shallower, we need less space to store them. Because we do not have access to the neural network deployed in the N-BaIoT research paper it is difficult to qualitatively compare our detection time. We will reimplement the architecture used and compare it to the one we coded in python.

Through the remainder of this paper, we will provide an overview of the auto-encoder solution proposed by the N-BaIoT research in Section II. We will then present our image processing and computer vision methodology that will serve as the basis for our pre-processing in Section III. Finally, we will examine the impact of the pre-processing by using two different Artificial Intelligence approaches in Section IV, first a sequential neural network followed by the auto-encoder model.

II. Related Work

For our purposes, we will be using the N-BaIoT dataset. This dataset focuses on the attack-phase of the botnet and is meant to test security tools that serve as a last-line of defense against attacks. The dataset collected data for nine IoT devices and was stored as .pcap files. There are 115 independent features present for each data point. Artificial Intelligence and Machine Learning are common approaches for tackling this dataset [9-12].

Existing solutions have presented an auto-encoder architecture for detecting botnet attacks using the dataset. Auto-encoders are designed using two sequential neural networks, the encoder network and decoder network, along with a bottleneck to link the two networks. An example auto-encoder architecture is shown in Figure 3. During operation, the model learns the features of benign traffic and will attempt to reconstruct it based on the input data. If it fails to reconstruct the traffic, it is identified as malicious. While the existing research indicates that it produces a low False Negative Rate (FNR) and a high True Positive Rate (TPR) when compared to other Artificial Intelligence (AI) models, the models cannot be applied across multiple devices. Additionally, the size of the model is fairly large, with over 40,000 trainable parameters.

III. Image Processing Methodology

Our overall methodology involves 6 keys steps, including 1) data collection, 2) feature extraction, 3) image creation and processing, 4) compression, 5) training, and 6) real-time traffic monitoring. In order for our pre-processing to occur, we must first convert the traffic information into image form. Through our methodology, each image is composed of a height that represents each packet we are

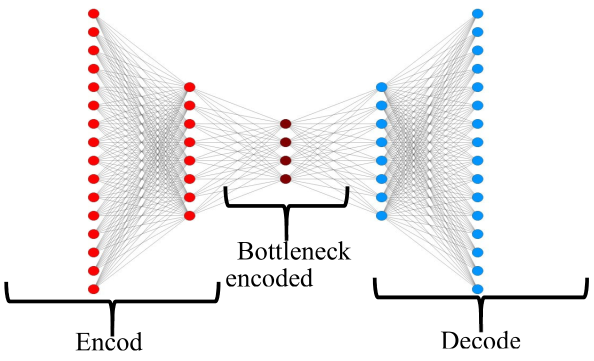


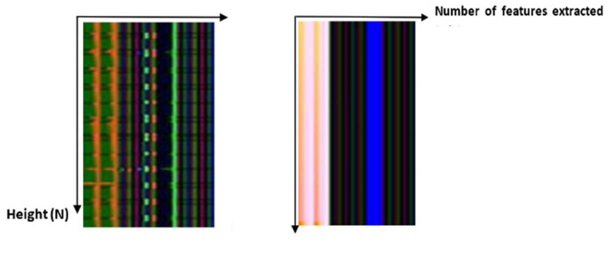
Fig. 3. Classic Auto-Encoder Architecture

examining and a width that represents the features of each packet. For our application, we decided to use a height of 16 packets and took a subsample of the N-BaIoT dataset. The subsample was composed of 23 features and was padded up to 24. Figure 4 provides the feature indexes that were selected for our application. The features then needed to be grouped into three and converted into their pixel format.

L = [0, 1, 2, 15, 16, 17, 30, 31, 32, 33, 34, 35, 36, 86, 85, 84, 83, 80, 81, 82, 65, 66, 67]

Fig. 4. Extraction vector L.

After collecting sufficient data for the desired time frame, we then convert the features into their pixels. The idea is to group data three by three to create each colored pixel. Since we selected 23 features from the overall 115, we still need to pad an additional value to reach a number divisible by 3. This will create three channels, each representing an RGB channel. Additionally, we normalize each channel according to Equation 1. In this case, *x* is the feature from the database, *contrast* is the contrast of the pixel, and *A* is a constant set to either 255 or 1. The result of the normalization ranges from [0:A]. An example of this process can be seen in Figure 5, comparing benign traffic and malicious traffic from a Mirai botnet attack.

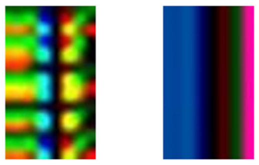


(a) Benign Data (b) Malicious Data

Fig. 5. Image representation of a benign file from the N-BaIoT dataset (a) and a malicious attack from the Mirai botnet (b)

$$f(x) = \left(2 \times sigmoid\left(\left|\frac{x}{contrast}\right|\right) - 1\right) \times A \qquad (1)$$

After converting the data into an image, we then apply a down sampling filter to better highlight different clusters of colorized filters. For our purposes, we explored four filters and examined which led to the best results. The first was the Lanczos filter, which is also known as a sharpening filter. When applied to the image, we increase the differences and jumps between clusters of images. For this case, it means hat we will not lose much packet context when we reduce the image's size. However, the resulting data is no longer normalized. Next, the Bilinear and Bicubic filters smooth the image when down sampling and add pixels by averaging the values of the surrounding pixels. The Bilinear filter is faster, but the Bicubic filer provides better tonal variations. Finally, the Nearest Neighbor algorithm is commonly used in media, such as pixel art and keeps hard edges, but is less precise and produces jagged effects. An example application of the Bicubic filter can be seen in Figure 6.



(a) Benign Data (b) Malicious Data

Fig. 6. Image generation sample. The up-sampling method is bicubic, the height of the image 16. The image generated on the left is issued from the 1.benign.csv file, on the right 1.mirai.scan.csv

IV. Artificial Intelligence Models

To examine the effectiveness of our pre-processing technique, we developed two Artificial Intelligence models. The first was a sequential neural network while the second was a shallow auto-encoder model. Our neural network model was composed of a single fully connected layer that applied the sigmoid activation function. A visual representation is presented in Figure 7.

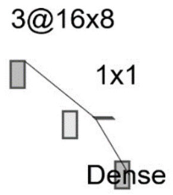


Fig. 7. One-layer sequential model

The auto-encoder was created using an encoder with 3 layers, an input layer, 16 2D Convolutional Layers with a (2,2) kernel and a Max Pooling 2D layer with a (2,2) kernel. The decoder for our auto-encoder had 16 Convolutional 2D

Transpose layers with a (3,3) kernel and 3 Convolutional 2D layers using a (2,2) kernel. This resulted having around 2,500 parameters compared to the N-BaIoT dataset that included 40,000 parameters. However, this does not mean the auto-encoder we created is faster. We will see this is because 2D Convolution layers are extremely slower than Dense Layers.

V. Results and Analysis

A. Sequential Neural Network

We use fully connected layers for this sequential model, therefor we have an input per pixel per color channel. For a colored image of (16x8) we thus have 384 weights since we have 3 color channels: Red, Green, Blue. Those inputs are all connected to a single output between 0 and 1 since we use a sigmoid activation function. It is the most simplistic architecture system we can create. This architecture uses the same Dense layers used in the N-BaIoT research paper, however we have approximatively 100 times les parameters in our network. Thus, we can consider this architecture to perform much faster.

When applying our shallow neural network, we first begin by training the dataset on the smart doorbell. The training dataset is composed of both the benign and Mirai attack sub databases. The concatenated data is shuffled and then split according to 80% training and 20% testing data. For our results, we consider both the TPR and TNR. The TPR represents the probability that an attack will be predicted as an attack, while the TNR represents when a benign datapoint is predicted as a benign datapoint. After training the model and applying it to the testing sample, we achieved a TPR and TNR of 99.99%.

After our success with detecting attacks within the Mirai botnet data, we then examined whether the trained model can be transferred to Gafgyt botnet attacks. For this dataset we introduce the TPRFlex and TNRFlex values, with TPRFlex representing the True Positive Rate when applying the model to the Gafgyt dataset and the TNRFlex representing the True Negative Rate when applying the model to the Gafgyt dataset. Further, we expand the testing dataset to include all nine IoT devices that were found within the dataset. As a result, our TPRFlex and TNRFlex values reflect the flexibility of the model when applied to new botnets and devices. Overall, we end up with **TPRFlex of 99.4%** and **TNRFlex of 98.96%.** When we examine only the benign data, we notice that the security camera benign data is consistently detected as malicious. This is presented in Figure 8. Additionally, when examining the malicious data, we note that the model also has difficulty detecting the Gafgyt scans compared to other attempts at attacking the network. This is presented in Figure 9.

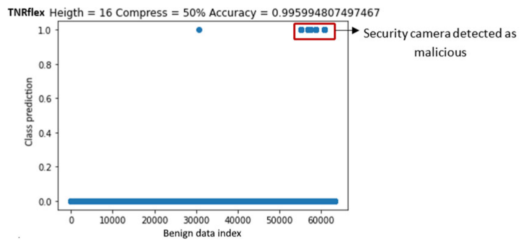


Fig. 8. Class prediction results on the benign data. A benign data is flagged as 0 and a malicious one as 1.

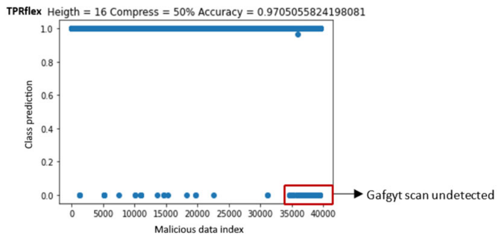


Fig. 9. Class prediction results on the Gafgyt data. A benign data is flagged as 0 and a malicious one as 1.

In addition to providing flexibility across botnets, we can also increase the flexibility of the model when applied across IoT devices. By increasing the height of our image and the compression of each filter, we can achieve a high TPR value. To test this, we trained our model using the Mirai botnet and benign data as before, but this time we adjust the height and compression. Also, we will only apply our trained model on the IoT Baby Monitor (4th device) and the IoT Security Camera (9th device). Our results are presented in Table I. We achieve the best TPR values when using a compression of 6.25 and height of 128.

TABLE I
TPR AND TNR FOR DIFFERENT HEIGHT AND COMPRESSION

	Baby Monitor	Security Camera

Height = 32
Compress = 50
	TNR = 99.96%
	TPR = 98.08%
		TNR = 100%
		TPR = 97.69%

Height = 32
Compress = 25
	TNR = 100%
	TPR = 97.70%
		TNR = 100%
		TPR = 97.51%

Height = 64
Compress = 12.5
	TNR = 100%
	TPR = 98.29%
		TNR = 100%
		TPR = 98.32%

Height = 128 Compress = 6.25	**TNR = 98.75% TPR = 99.73%**	**TNR = 100% TPR = 98.38%**

B. Auto-Encoder

Next, we examined our auto-encoder model using different down sampling filter. We considered the same training and testing methodology as was completed with the sequential neural network. Namely, we concatenated the benign and Mirai attack sub databases, shuffled them and split them 80% training and 20% testing. Further, we examined our Lanczos, Bicubic, Bilinear, and Nearest Neighbor filters prior to providing the dataset to the auto-encoder. Again, we examine the (TPR and TNR. Based on our results, the Bicubic filter performed the best with a TPR of 98.98% and a TNR of 99.98%. The full results for all filters can be found in Table II.

TABLE II
TPR AND TNR FOR DIFFERENT DOWNSAMPLING METHODS

	Lanczos	**Bicubic**	Nearest	Bilinear
TPR (%)	98.81	**98.98**	99.40	98.97
TNR (%)	99.96	**99.98**	78.78	**99.98**

Overall, this architecture performs as quickly as the architecture we implemented from the N-BaIoT research paper despite having much less parameters. This is because we are using 2D convolutional layer instead of Dense. Because we wanted to try and create colored images, we could not use the latter. In the future we could try generating larger black and white images. This would enable us to use those Dense layers and possibly speed up the run time.

VIII. Conclusion and Future Work

Through this project, we examined the potential application of image processing as a pre-processing stage in detecting botnet attacks. Four down sampling filters were tested, including the Lanczos, Bicubic, Bilinear, and Nearest Neighbor filters. Additionally, a sequential neural network and auto-encoder model were applied to the processed data to classify cyberattacks. Overall, the proposed architecture was successful at detecting botnet attacks within the N-BaIoT dataset. Further, our auto-encoder achieved similar TPR and TNR as the N-BaIoT architecture. This is significant as we were able to use a much shallower model, saving computational resources and decreasing the time consumed for running the model. In addition, we showed that when our model was only trained on the Mirai botnet that it was capable of detecting attacks from the Gafgyt botnet as well, showcasing the flexibility of the model.

In our future work we would like to further experiment feature extraction methods that would better fit the image processing method. It should also be noted that a single image creation method with a fixed number of packets has been tested here. An image with a dynamic height might provide better results. Ultimately those methods should further define the concept of predictability of an IoT.

At last, the image processing method is very experimental. We would like to optimize the hyper parameter we used throughout our experiment and create a shallow but performant neural network. That would also mean testing different image generation algorithms.

Acknowledgements

First and foremost, we would like to thank our friend **Emile Barbier-Renard**, currently a PhD student in image processing, for his help on the image processing to be applied. We would also like to thank IIT Chicago and the framework established by Professor **Jafar Saniie** without which we would never have done this study. At last, we want to thank the research team that conducted the paper on the N-BaIoT dataset on which this experiment is based.

References

[1] Y. Meidan et al., "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Auto-encoders," in IEEE Pervasive Computing, vol. 17, no. 3, pp. 12-22, Jul.-Sep. 2018, doi: 10.1109/MPRV.2018.03367731.

[2] C. Kolias, G. Kambourakis, A. Stavrou and J. Voas, "DDoS in the IoT: Mirai and Other Botnets," in Computer, vol. 50, no. 7, pp. 80-84, 2017, doi: 10.1109/MC.2017.201.

[3] J. Margolis, T. T. Oh, S. Jadhav, Y. H. Kim and J. N. Kim, "An In-Depth Analysis of the Mirai Botnet," 2017 International Conference on Software Security and Assurance (ICSSA), 2017, pp. 6-12, doi: 10.1109/ICSSA.2017.12.

[4] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal and B. Sikdar, "A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures," in IEEE Access, vol. 7, pp. 82721-82743, 2019, doi: 10.1109/ACCESS.2019.2924045.

[5] E. Bertino and N. Islam, "Botnets and Internet of Things Security," in Computer, vol. 50, no. 2, pp. 7679, Feb. 2017, doi: 10.1109/MC.2017.62.

[6] M. Gromov, D. Arnold and J. Saniie, “Tackling Multiple Security Threats in an IoT Environment,” in *2022 IEEE International Conference on Electro Information Technology*, 2022.

[7] Baptista, S. Shiaeles and N. Kolokotronis, "A Novel Malware Detection System Based on Machine Learning and Binary Visualization," 2019 IEEE International Conference on Communications Workshops (ICC Workshops), 2019, pp. 1-6, doi: 10.1109/ICCW.2019.8757060.

[8] L. Barlow, G. Bendiab, S. Shiaeles and N. Savage, "A Novel Approach to Detect Phishing Attacks using Binary Visualisation and Machine Learning," 2020 IEEE World Congress on Services (SERVICES), 2020, pp. 177-182, doi: 10.1109/SERVICES48979.2020.00046.

[9] S. Hojjatinia, S. Hamzenejadi and H. Mohseni, "Android Botnet Detection using Convolutional Neural Networks," 2020 28th Iranian Conference on Electrical Engineering (ICEE), 2020,pp.1-6,doi:10.1109/ICEE50131.2020.926067 4

[10] D. McDermott, F. Majdani and A. V. Petrovski, "Botnet Detection in the Internet of Things using Deep Learning Approaches," 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1-8, doi: 10.1109/IJCNN.2018.8489489.

[11] Supranamaya Ranjan. Machine learning based botnet detection using real-time extracted traffic features, March 25 2014. US Patent 8,682,812.

[12] Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: An Ensemble of Auto-encoders for Online Network Intrusion Detection. Proceedings 2018 Network and Distributed System Security Symposium. https://doi.org/10.14722/ndss.2018.23204