**AAI-530 Final Project: IoT Botnet Attack Detection**

Desmond Turner

Shiley-Marcos School of Engineering: University of San Diego

AAI-530: Data Analytics and IoT

Dr. Ying Lin

February 28, 2022

**Introduction**

The dataset I chose recorded network traffic across nine IoT devices. IoT devices are more prone to hacking than personal computers (Meidan, et al., 2018). Many are deployed without changing the factory settings which leaves their credentials open to brute-force password cracking. Once hacked, the IoT device becomes a part of a botnet, and can be used for malicious activity by the central controller or the infected peers. The rapid deployment of vulnerable, internet connected IoT devices is the perfect resource for criminals conducting distributed denial of service attacks. An enterprise with hacked devices suffers because the extra networking/computing demands from the botnet are difficult for the already tightly constrained IoT devices to manage. Performance can take a hit as a result. To solve this problem I developed a semi-supervised classification method for determining whether IoT devices are infected with malware using anomaly detection. Several transformations and principal component analysis prepare the data for local outlier factor models which successfully classify each device using novelty detection.

**Dataset**

Packet information for nine IoT devices was captured using port mirroring. Each device contains five to eleven .csv files, each with hundreds of thousands of rows. Each device has one benign file and several malicious files. The reason for there being multiple infected files is because the original authors conducted several types of attacks using both Mirai, and Gafgyt botnets. Within each botnet category there are several specific attack types which enables multivariate classification. Mirai and Gafgyt are amongst the top botnet malwares. The dataset captures a variety of Mirai/Gafgyt attacks including udp flooding, tcp flooding, brute-force scanning, junk, and combination attacks for every device, each receiving its own .csv file. Devices themselves included a Danmini doorbell, Ecobee thermostat, Enmio doorbell, Philips baby monitor, two Provision security cameras, two SimpleHome security cameras, and a Samsung webcam. Although multivariate classification was possible, I stuck to binary classification due to time constraints. The authors of the dataset intended to detect attacks stemming from devices in real time using deep autoencoders. For whatever reason the time stamps that describe when each attack takes place are not included in the dataset. For this reason, I designed a method to classify devices as a whole since the original problem is impossible to solve given the absence of labelled data.

The application reads each .csv file into its own data frame. Features were engineered using various statistics computed on packet streams. Weight, mean, standard deviation, radius, magnitude, and covariance were computed on stream aggregations. *H* aggregation describes recent traffic from the packet’s host. *HH* describes recent traffic going from the packet’s host to the destination’s host. *HpHp* describes recent traffic going from the packet’s host and port to the packet’s destination’s host and port. Finally, *HH\_jit* describes the jitter of traffic going from the packet’s host to the destination’s host. Jitter is the variation in delay between received packets. The data was difficult to get an intuitive feel for because it came in the form of summaries of packets whose original form is already abstract enough as it is. Despite the uninterpretability of this complex dataset, the preprocessing steps make it perfectly decipherable for machine learning.

Chart, histogram

Description automatically generated

**Transformations**

The data conveniently comes prepackaged with features engineered ahead of time. 115 high quality features are perfect for deep learning. For the purposes of this application, the data needs additional transformations to make it suitable for traditional machine learning. The model I chose for classification was a local outlier factor (LOF). Training the local outlier factor on the original input size took an indeterminable amount of time, a process swiftly abandoned.

To reduce the number of inputs, principal component analysis (PCA) is implemented to cluster the variance into a select number of components. PCA captures 60% to 80% of the total variance of each .csv when selecting the top five components which gives the best results. Originally eight components were used because they captured 80% to 95% of the total variance, and I naively assumed more variance was better. Not only did reducing 115 features to five components speed up the training/testing process, but it improved performance as well. Training a model on the reduced components drastically reduces false positives, a key aspect of this project since the detection system should not falsely alert its users to investigate/dispose of an uncompromised devices. Using less components with less variance allows the model to generalize far greater on unseen data. Further, variance does not necessarily lead to predictive value, it could be nothing more than noise (Walker, 2019). PCA presumably removes much of this noise along with some useful variance, but in the end finds a favorable balance between the two.

The data needs to go through several transformations before principal component analysis so that the variation within each component is distinct. PCA assumes stationarity, meaning there is no positive or negative trend in the data. Of all the methods to ensure trend-stationarity differencing is selected because of its simplicity. Every signal was differenced because it was too computationally expensive to test every column (115), on every file (89) with an augmented Dickey-Fuller test (2.5 seconds per column), which would have taken around seven hours to complete. An augmented Dickey-Fuller test simply tests a signal for trend-stationarity. Differencing was an essential step. A nonessential step that nonetheless improved results was standardizing the data. Standardizing data ensures that each column is on a comparable scale by bringing its mean and standard deviation as close to zero and one as possible. This helps to stabilize variance. In tandem, the data can be fed to PCA in a manner that allows it to separate the components in a way that does not muddle contrasting information.

PCA components

Chart, bar chart, histogram

Description automatically generatedChart

Description automatically generated

**Training and Optimization**

The procedure used to train the classification method is unorthodox due to the nature of the problem. Each device behaves extremely differently from one another, so a single model would struggle to understand the patterns amongst them all simultaneously. Instead, a semi-supervised learning approach is used that creates an individual model for each individual device. When each model understands the typical behavior of its corresponding device, it can use anomaly detection to pick out abnormal behavior that might be indicative of a botnet attack. For this reason, the models are trained only on the benign device data to obtain the proper baseline. When testing, the goal is to find anomalies within the infected device’s packets, thus determining them to be infected. This semi-supervised approach to anomaly detection is called novelty detection.

To train each model, benign data is divided into three equal size subsets. A training subset, an optimization subset, and a testing subset. This is because of a well-known principle that the optimization and testing data should be separate to avoid biased results. Each model is trained on its training data to obtain the baseline for the device’s behavior. Local outlier factor algorithms compare the local density deviation of a value with respect to its neighbors and considers outliers to be those with a considerably lower density. This gives the advantage of finding local and global anomalies which is fitting for the nature of botnet attack detection. Grid-search optimize searches dictionaries of possible values to find the parameters that perform best according to a score obtained from testing on the optimization data (scikit, 2022). The metric for the score in this case rewards minimizing false positives, as it is most important to ensure uncompromised IoT devices are calculated as such. Once the best hyperparameters are found, each model is tested on the (benign) test subset with the hope that zero false positives are predicted.

**Results**

After the negative data is tested, positive data follows. The 80 infected device files are tested individually on their respective models. If a single anomaly is found the device is predicted to be infected. This is another reason why minimizing false positives was so important during the training and optimization phase. All test results are then concatenated and compared with their true labels to compute true positives, false negatives, etc., which are then used to compute metrics for the performance of the overall classification method. When using the ideal number of components (5) and the chosen hyperparameters from grid-search the model performs exceptionally well. Accuracy is 0.989, precision is 0.988, sensitivity is 1, specificity is 0.889, resulting in an F1 score of 0.994.

To deploy this model in an enterprise setting, devices free of malware could be used to train models within a cloud-based application. Streams of sniffed packets from untested devices could be sent to the cloud and analyzed. The specificity of 0.889 would need to be much higher and tested on a larger set of benign devices than nine. The application would work best in the cloud using port mirroring as opposed to in the devices themselves, because those that are tightly constrained could not handle the additional computational costs of running the local outlier factory.

Chart

Description automatically generated Diagram

Description automatically generated with medium confidence

**References**

Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, D. Breitenbacher, A. Shabtai, and Y. Elovici 'N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders', IEEE Pervasive Computing, Special Issue - Securing the IoT (July/Sep 2018).

*Gafgyt botnet absorbs code from notorious Mirai Strain: SaferNet VPN*. SaferNet VPN | Creating A Safer Internet. (2021, May 20). Retrieved March 1, 2022, from https://www.safernetvpn.com/gafgyt-botnet-absorbs-code-from-notorious-mirai-strain/

*Novelty detection with local outlier factor (LOF)*. scikit. (n.d.). Retrieved March 1, 2022, from https://scikit-learn.org/stable/auto\_examples/neighbors/plot\_lof\_novelty\_detection.html#:~:text=The%20Local%20Outlier%20Factor%20(LOF,lower%20density%20than%20their%20neighbors.

Walker, B. (2019, December 31). *PCA is not feature selection*. Medium. Retrieved March 1, 2022, from https://towardsdatascience.com/pca-is-not-feature-selection-3344fb764ae6