IoT Botnet Detection on Flow Data using Autoencoders

Orestis Kompougias*, Dimitris Papadopoulos*, Evangelos Mantas*, Antonis Litke*,
Nikolaos Papadakis†, Dimitris Paraschos†, Akis Kourtis‡, George Xylouris‡

*Infili Technologies PC, Athens, Greece — †Stratiotiki Sxoli Evelpidon, Vari, Greece —
‡ORION Innovations PC, Athens, Greece

***Abstract*—The rapid growth of the Internet of Things and the proliferation of easily compromisable IoT devices has led to a drastic increase in the occurrence of IoT-based botnet attacks. Hackers are keen on exploiting the vulnerabilities of smart devices, which are seen as easy targets often lacking robust security mechanisms. Identifying botnet activity is an active research topic and remains a challenging task due to the continuous evolution of botnet families that employ a large number of attack vectors. Traditional rule-based approaches which rely on signature matching, heuristics and behavioral profiling are always lagging one step behind the attacker, leading researchers to the development of machine and deep learning methods for the detection of compromised IoT device behaviour. In this paper, we model botnet traffic identification as an anomaly detection task, aiming at establishing a baseline of benign traffic, in order to detect unusual behavior using Netflow data. We propose a feature engineering and deep learning-based detection framework based on two Autoencoder architectures: (i) a vanilla implementation of a deep Autoencoder and (ii) GANomaly which has never been used in the context of network traffic analysis before. We validate the performance of the proposed methodology on the CICIDS2017 dataset which has been widely used for cybersecurity benchmarks and show that it is possible to induce highly accurate unsupervised learning models to detect previously unseen botnet behaviour.**

I. Introduction

The threat landscape of the cyber world evolves rapidly, and new sophisticated threats emerge making attacks undetectable by conventional cybersecurity-related countermeasures and techniques. Threats such as the Emotet botnet [1], one of the most significant botnets of the decade, have proved the need for quick and reliable detection tools within the network infrastructure of an organisation or enterprise. According to ENISA [2] a botnet is a set of computers infected by bots, pieces of malicious software that get commands from a central node, the bot master. The malicious acts carried out by botnets can range from DDoS attacks to illegal file hosting and remote code execution (RCE).

IoT devices can also process data and behave as basic computers with specialized functions. However, not all of these devices had adequate security to ward off cyberattacks based on their specifications when they entered the market. Therefore, IoT gadgets went online in homes and businesses without the same level of security awareness that PCs have gained over time, making them a prime target for IoT botnets and other dangers. IoT botnets are a collection of smart devices hijacked by cybercriminals with the same end purpose as traditional botnets, which use networks of hacked computers to execute assaults or harmful activities like cryptocurrency mining.

In our work, we concentrate on IoT botnet detection using a combination of feature engineering methods with two distinct Autoencoder architectures: (i) a vanilla version of the deep Autoencoder and (ii) GANomaly, which is a combination of Autoencoders with conditional Generative Adversarial Networks (GANs), originally implemented for computer vision tasks and re-purposed for botnet detection. To the best of our knowledge, this is the first adaptation of GANomaly on network traffic analysis. We leverage a popular benchmark dataset to train our models using only benign traffic and try to detect botnet activity on unseen data.

This paper is structured as follows: We present related work on IoT attack detection in Section II. We describe the methodology and rationale behind our preprocessing feature engineering techniques and deep learning models in Section III and elaborate on the experimental setup (benchmark dataset and training framework) in Section IV. We discuss the results of our models' performance in Section V, while Section VI concludes our methodology and gives directions for future research.

II. Related Work

The configuration of IoT botnets generally resembles the one of traditional botnets, comprising two main modules as shown in Figure 1:

1) a command and control server from which a threat actor sends commands and controls the botnet and,
2) a number of infected devices (i.e. bots, zombies) that are individually hijacked as part of a larger network of similarly infected bots.

In the case of IoT devices, bots can also encompass extra features such as a port-scanning functionality, a cryptomining script or a DDoS attack generator [3]. Their widespread disruption can be primarily attributed to the existence of many unpatched IoT devices ranging from TV set-top boxes, DVRs, webcams, smartwatches to medical devices and exploitable car systems [4], as a result of either end-user negligence or lack of vendor support [5].

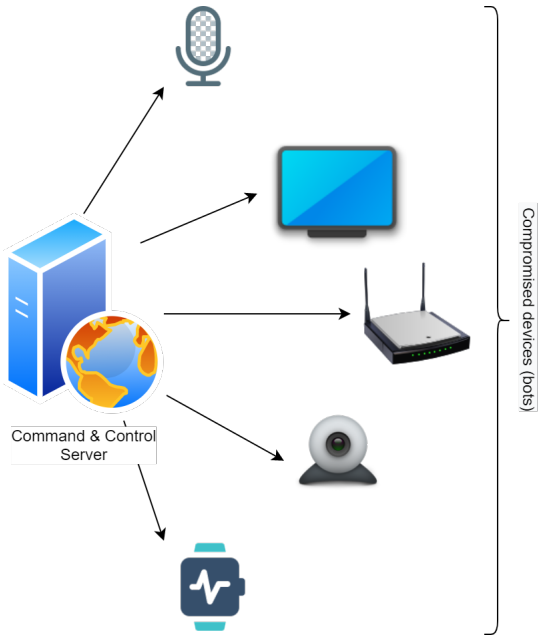

Fig. 1: Example of an IoT botnet.

The need to detect and classify botnet traffic within network flows is ever growing and has been the subject of prior works. According to the work of Feily et al. [6] there are four distinctive approaches to detect botnets: signature-based, anomaly-based, DNS-based and mining-based, with variable success rates depending on the algorithm and the dataset. A variety of different data types have been used for botnet detection and tracking, including DNS data, Netflow data, packet tap data, address allocation data, honeypot data and host data [7].

Traditional detection methods used by Intrusion Detection Systems (IDS), are mostly rule-based and leverage signature matching to detect well-known bots [8] [9]. Modern rule-based approaches also propose DNS blacklisting [10], rule construction on statistical DNS features [11] or the use of Fuzzy Rule Interpolation (FRI) reasoning [12], leading to the generation of more realistic and comprehensive alerts. However the momentum has shifted to the use of machine learning algorithms and techniques due to the vast amount of information of the network headers that the IDS has to process, resulting in a time consuming operation.

Machine learning approaches can be generally classified based on the desired outcome of the algorithm as supervised and unsupervised. The former are trained using labeled data inputs to predict the label of future inputs. In the context of botnet detection, supervised ML methods are commonly used for the implementation of network traffic classifiers, able to distinguish between benign and malicious traffic by assigning the corresponding label to each log or by identifying traffic belonging to different botnets. A number of techniques have demonstrated good detection results in this line of work, including Support Vector Machines [13], tree-based methods [14], and modern neural network-based architectures [15] [16]. However, the inherent weakness of supervised approaches is that they rely on the existence of annotated datasets to train in order to classify network incidents. While they are thriving in cases where a vast history of application signatures exists to get labeled data, their performance is heavily dependent on the training data and falls short in cases of adaptive malware variants (e.g., polymorphic and metamorphic variations [17]) characterized by the ability of blending in with legitimate applications to avoid detection. On the other hand, unsupervised learning methods tend to view botnet detection as an anomaly detection problem that requires no a priori knowledge of bot signatures or labeled traffic in general. The main characteristic of unsupervised, anomaly-based approaches is that they learn a client's trusted behavior by establishing a baseline of what is considered normal/usual traffic, in order to detect any unusual behavior. They are thus more generalizable and effective in cases of 0-day attacks where no labels are available, whilst supervised learning methods are effective only for known attacks. Unsupervised methods such as k-means clustering [18], Self-Organizing Map (SOM) [19], Local Outlier Factor (LOF) [20], One-class-SVM (OCSVM) [21] and more recent Autoencoder architectures [22] have been employed for the detection of botnet traffic.

III. Methodology

A. Feature engineering

Network traffic is inherently a multivariate time series. In order to encapsulate the time aspect of it we need to perform feature engineering on the provided flow characteristics. The alternative would be to use a model that incorporates temporal data such as a Recurrent Neural Network, however their ability to learn temporal relationships is limited in long-term series.

The simplest approach would be to perform feature engineering using a sliding window of some size. For example, we could create a feature that represents the average number of packets sent in the last 10 seconds. The downside to this approach, however, is that when considering the entire network traffic it would be hard to pinpoint which specific IPs created a potential fluctuation to this metric.

Our approach first performs a grouping by IP and then calculates features using sliding windows. This serves our goal of pinpointing malicious traffic since these features represent an underlying distribution that is both easier to learn and more specific to our task.

The implemented feature engineering strategy concludes with the calculation of statistics (mean and standard deviation) over highly representative network features such as packet size, packets sent and header size. In the case of categorical features such as source/destination ports and destination IPs, we implement a counting of the occurrence of these values.

B. Autoencoder variants

For the purposes of this work, we experimented with two Autoencoder variants:

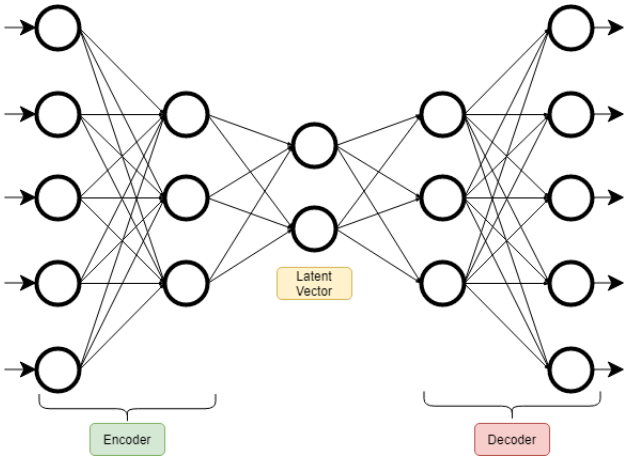

Fig. 2: Autoencoder architecture. The forward pass of data is from left to right. The input is first encoded into a latent vector and then decoded, producing the reconstruction of the input.

1) Deep Autoencoders: The vanilla implementation of Autoencoders is a Neural Network architecture whose purpose is to learn the underlying distribution of data by forcing dimensionality reduction and reconstruction of the original input. To elaborate, Autoencoders receive an input $x \in \mathbb{R}^N$, which gets passed through a series of Neural Network layers that produce progressively smaller outputs (Encoder) as shown in Figure 2. This bottleneck performs dimensionality reduction of the input x to a latent vector $z \in \mathbb{R}^{L<N}$. The latent vector is then passed through a series of Neural Network layers that produce progressively bigger outputs (Decoder) producing the output x' with the goal of reproducing the original input x. Autoencoders for non-binary regression are trained using the Mean Squared Error loss.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - x_i')^2 \tag{1}$$

For the purposes of outlier detection, Autoencoders are trained with normal/benign traffic only and are expected to produce outputs with a high loss when fed with anomalous data. This happens because anomalies do not belong to the distribution of normal behaviour that is learned by Autoencoders.

2) GANomaly: The GANomaly architecture proposed by Akcay et al. [23] is a Generative Adversarial Network (GAN) [24] purposefully built for outlier detection purposes. A GAN is comprised of two neural networks contesting with each other in a zero-sum game, depicted in Figure 3. The Generator network aims to learn the underlying data distribution and produces samples from the learnt distribution. The Discriminator network identifies data as either real or generated by the Generator. After training, the Generator can be used to generate new samples or perform various other tasks.

GANomaly is a GAN that uses an Autoencoder as the Generator. Two extra encoders are also employed, one being the Discriminator and the other being a feature extractor that re-encodes the reconstructed input. The Generator's loss is a weighted sum of three loss functions which all aim to help the Generator learn the underlying data distribution so that any outliers stand out from the other data points.

$$\mathcal{L}_{adv} = \mathbb{E}_{x\sim p_x}||D_E(x) - \mathbb{E}_{x\sim p_x}D_E(G(x))||_2 \tag{2}$$

$$\mathcal{L}_{con} = \mathbb{E}_{x\sim p_x}||x - G(x)||_1 \tag{3}$$

$$\mathcal{L}_{enc} = \mathbb{E}_{x\sim p_x}||G_E(x) - F(G(x))||_2 \tag{4}$$

$$\mathcal{L}_G = w_{adv}\mathcal{L}_{adv} + w_{con}\mathcal{L}_{con} + w_{enc}\mathcal{L}_{enc} \tag{5}$$

$\mathcal{L}_{adv}$ is the adversarial L_2 loss between two outputs of the Discriminator Encoder D_E, one given the input x and the other given the reconstructed input $G(x)$. $\mathcal{L}_{con}$ is the reconstruction loss of the Generator/Autoencoder. $\mathcal{L}_{enc}$ is the Encoder loss measuring the difference between the Generator and feature extractor latent vectors, $G_E(x)$ and $F(G(x))$ respectively.

The Discriminator is trained using the standard GAN binary cross-entropy loss.

$$\mathcal{L}_D = -(ylog(D(x)) + (1-y)log(1 - D(x))) \tag{6}$$

For making a prediction on whether a sample is an outlier or not we obtain an anomaly score $\mathcal{A}$ by calculating the difference between the feature extractor's latent vector and that of the Generator, which we scale to [0,1]. A high anomaly score means a high confidence of the sample being malicious, while a low score means that the sample is predicted as being normal. The decision boundary can be defined on a per-use case basis depending on the precision-recall trade-off that is acceptable.

$$\mathcal{A}(x) = ||G_E(x) - F(G(x))||_1 \tag{7}$$

IV. Experiments

A. Dataset

Numerous works exist in literature trying to address the botnet detection problem with varying results. The work of García et al. [25] focuses on comparing three different detection methods CAMNEP [26], BClus [27] and BotHunter [28] using the same real dataset, concluding that a realistic network dataset is essential to evaluate the accuracy of the detection. In our work we used the CICIDS2017 dataset provided by Sharafaldin et al. [29]. This dataset contains real-world data with a simulated behaviour of 25 users within the network, capturing packets of the most common HTTP, HTTPS, FTP, SSH, and email protocols. There is also a great variety of simulated attack vectors such as Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS. The full list of different network traffic types is shown in Table I.

For our training dataset, we use Monday's traffic, which only contains naturalistic benign background traffic. It should be noted that, while additional normal traffic could be acquired by combining the benign traffic subsets of other days, these

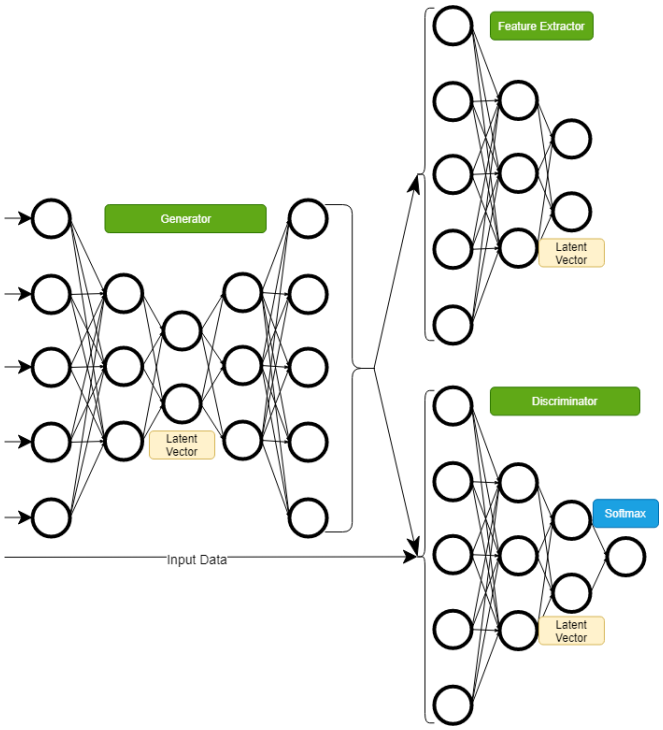

Fig. 3: GANomaly architecture.

TABLE I: CICIDS2017 Dataset description

Dataset	Traffic type	# of records
Monday-WorkingHours. pcap_ISCX.csv	Benign	529918
Tuesday-WorkingHours. pcap_ISCX.csv	Benign SSH-Patator FTP-Patator	432074 5897 7938
Wednesday-WorkingHours. pcap_ISCX.csv	Benign DoS Hulk DoS GoldenEye DoS Slowloris DoS Slowhttptest Heartbleed	440031 231073 10293 5796 5499 11
Thursday-WorkingHours-Morning-WebAttacks. pcap_ISCX.csv	Benign WebAttack-Bruteforce WebAttack-SQL Injection WebAttack-XSS	168186 1507 21 652
Thursday-WorkingHours-Afternoon-Infiltration. pcap_ISCX.csv	Benign Infiltration	288566 36
Friday-WorkingHours-Morning.pcap_ISCX.csv	Benign Bot	189067 1966
Friday-WorkingHours-Afternoon-Portscan. pcap_ISCX.csv	Benign Portscan	127537 158930
Friday-WorkingHours-Afternoon-DDoS. pcap_ISCX.csv	Benign DDoS	97718 128027

were not used in our experiments to resemble a more realistic operational setting, where a model's training is usually based on limited historical data. Our test dataset is Friday's morning traffic, which contains normal activities as well as malicious IoT Botnet ARES communications (the only IoT-botnet related attack). The ARES botnet was discovered on Android OS-based Set-Top Boxes (STBs) to be exploiting insecure configuration and misuse of the Android Debug Bridge infrastructure used by a majority of STBs and smart TVs. We preprocess the datasets and perform feature engineering as mentioned in the previous section. Given that our datasets are comprised of features with diverse ranges, we also apply standardisation to resolve any bias caused by the gradient of larger parameters dominating the networks' weights updates.

B. Training

We use the Keras framework of Tensorflow using Python and train on an NVIDIA RTX 2080 Super GPU. For both models, we use Leaky ReLU activations, batch normalization, a batch size of 512, a latent vector size of 25% the original dimensions and the Adam optimizer [30]. We train for 60000 iterations and present the best models in the following section.

1) Autoencoder: For the Autoencoder, we use a learning rate of 0.002 and a Mean Squared Error loss. The Encoder consists of 3 layers having 128, 64 and 32 neurons respectively. The latent vector is of size 21 while the Decoder consists of 3 layers with 32, 64 and 128 neurons.

2) GANomaly: For the training of GANomaly we use a learning rate of 0.0002 for both Generator and Discriminator, Eq.5 as the Generator loss and Eq.6 as the Discriminator loss. The Encoders and Decoders have the same architecture as the ones used for the Autoencoder, described above. For the Discriminator labels, we utilize one-sided label smoothing of value 0.9 for the reconstructed inputs given by the output of the Generator which prevents overconfidence of the Discriminator. We do not use a balancing scheme for training the Generator and Discriminator and train both equally.

V. Results & Discussion

This work is a preliminary investigation on outlier detection using Netflow data intended for an Intrusion Detection System. For this reason, our main goal was focused on achieving high precision, which means detecting the botnet traffic without flagging too much normal traffic as malicious. The intuition behind this rationale is that if the botnet has been detected, the IDS can prevent further traffic. Thus, for this use case, precision is much more important than recall.

After training both models we obtain the precision scores presented in Table II. In our case, precision is defined as the number of true positives (actual botnet flows) over the number of true positives plus the number of false positives. We identify three decision boundaries as potential candidates which we use to compare the two models. These decision boundaries were found by accepting various percentages of the normal data as false positives. All results represent the models that had the highest overall precision across the three decision boundaries.

An interpretation of Table II is as follows: For example, when accepting that 0.001% of normal traffic can be predicted as malicious we get a decision boundary that contains that percentage of normal traffic on the wrong side of it. For that decision boundary, we get a precision of 0.92 given by GANomaly, meaning 92% of the traffic labelled as malicious is indeed botnet traffic.

We also present the underlying measurements of true positives and false positives in Table III at the same decision boundaries for both models. For example, when accepting 0.006% of normal traffic to be predicted as malicious we get 11 false positives, 13 true positives by the Autoencoder and 20 by GANomaly.

TABLE II: Precision scores of both models at various decision boundaries. The decision boundaries represent the percentage of normal traffic that has been predicted as malicious.

Model	**Precision**		
Decision Boundary	*0.001%*	*0.006%*	*0.011%*
Autoencoder	0.75	0.54	0.44
GANomaly	0.92	0.65	0.58

TABLE III: Number of true/false positives at various decision boundaries. The decision boundaries represent the percentage of normal traffic that has been predicted as malicious. Total outlier samples after preprocessing are 1947 and normal traffic samples are 188955.

Measurements	**# of occurrences**		
Decision Boundary	*0.001%*	*0.006%*	*0.011%*
FP (both models)	1	11	20
Autoencoder TP	3	13	16
GANomaly TP	11	20	28

Overall, the GANomaly model has proven to be more effective at learning the underlying data distribution. This can be seen both numerically in the tables above and visually when comparing the scatter plots of the testing dataset between the two models in Figures 4 and 5. In these figures it is apparent that GANomaly is much more confident in the data distribution since the blue points, representing normal traffic, have an anomaly score that is on average closer to zero compared to their reconstruction error when passed through the Autoencoder. It can also be observed that the red points, representing malicious traffic, achieve better separation from the blue points with GANomaly which is caused by a better modelling of the data distribution, making outliers stand out more. It is however, worth noting that the Autoencoder eventually achieves lower average reconstruction losses when trained for more iterations but performs considerably worse for outlier detection. This further points to the fact that the training scheme of GANomaly results in a more accurate approximation of the underlying data distribution.

VI. Conclusions

In this work, we proposed a methodology for the efficient detection of IoT botnet activity on Netflow traffic. To this end, we applied a combination of feature engineering techniques

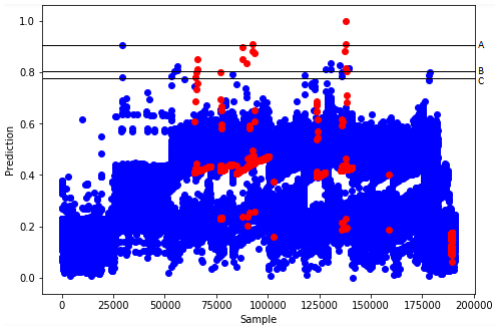

Fig. 4: Scatter plot of Autoencoder. Red points are malicious samples, blue points are normal traffic. The prediction (y axis) is the reconstruction error of each sample scaled to the range [0,1]. This represents the confidence of a sample being malicious. Decision boundary A predicts 0.001% of normal traffic as malicious while B and C predict 0.006% and 0.011% respectively.

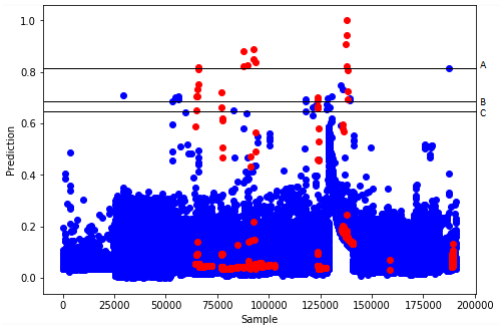

Fig. 5: Scatter plot of GANomaly. Red points are malicious samples, blue points are normal traffic. The prediction (y axis) is the anomaly score (Eq.7) scaled to the range [0,1]. This represents the confidence of a sample being malicious. Decision boundary A predicts 0.001% of normal traffic as malicious while B and C predict 0.006% and 0.011% respectively.

to extract robust features that better represent the underlying problem, to improve our predictive performance on unseen data. We performed anomaly detection using two state-of-the-art deep-learning architectures, a deep Autoencoder and GANomaly (which was re-purposed for the traffic analysis task) by training them on benign traffic and testing them on a mixture of normal and botnet network flows. Our models were successful in establishing a baseline of normal network behaviour, outputting a higher reconstruction error when fed with botnet flows which was used as an indicator of malicious activity given a predefined decision boundary.

Throughout our work, we acknowledged that outlier detection in network traffic by using heuristic-based approaches such as neural networks is an inherently complex problem since the activities of each user in a network can vary dramatically making the underlying distribution hard to model. Nonetheless, highly representative features can be the catalyst to this problem. We also observed that proper feature

extraction can potentially be more important than even the choice of model and its parameters. This points to the fact that for these types of tasks, focusing heavily on the dataset statistics and feature extraction can prove extremely beneficial. Furthermore, having access to data that is more detailed than the Netflow format, such as full packet captures, can further improve performance as even more meaningful features can be extracted from them. To this end, our approach could have practical applications as part of a hybrid threat intelligence system that exploits multiple data sources to deliver context-agnostic technical indicators [31].

Future work could investigate the effectiveness of our approach on other botnet families or different network attacks. We would also like to further finetune the implemented feature extraction strategy, by modifying the characteristics of the extracted features using different sliding windows for the calculation of statistics. Finally, another line of work could be targeted towards the optimisation of our method for the online (real-time) analysis of flow streams.

VII. Acknowledgement

The work described in this article has received funding by the European Union Horizon 2020 research and innovation programme, supported under Grant Agreement no. 883335.

References

[1] Emotet botnet disrupted in global operation (Jan. 2021).
URL https://www.welivesecurity.com/2021/01/28/emotet-botnet-disrupted-global-operation/

[2] Botnets.
URL https://www.enisa.europa.eu/topics/csirts-in-europe/glossary/botnets

[3] B. Vignau, R. Khoury, S. Hallé, 10 years of iot malware: A feature-based taxonomy, in: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), IEEE, 2019, pp. 458–465.

[4] M. Frustaci, P. Pace, G. Aloi, G. Fortino, Evaluating critical security issues of the iot world: Present and future challenges, IEEE Internet of things journal 5 (4) (2017) 2483–2495.

[5] I. Yaqoob, E. Ahmed, M. H. ur Rehman, A. I. A. Ahmed, M. A. Al-garadi, M. Imran, M. Guizani, The rise of ransomware and emerging security challenges in the internet of things, Computer Networks 129 (2017) 444–458.

[6] M. Feily, A. Shahrestani, S. Ramadass, A survey of botnet and botnet detection, 2009, pp. 268–273. doi:10.1109/SECURWARE.2009.48.

[7] P. Amini, R. Azmi, M. Araghizadeh, Botnet detection using netflow and clustering, Advances in Computer Science: an International Journal 3 (2) (2014) 139–149.

[8] M. Roesch, et al., Snort: Lightweight intrusion detection for networks., in: Lisa, Vol. 99, 1999, pp. 229–238.

[9] A. Ramachandran, N. Feamster, D. Dagon, et al., Revealing botnet membership using dnsbl counter-intelligence., Sruti 6 (2006) 49–54.

[10] P. Sharma, S. Kumar, N. Sharma, Botmad: Botnet malicious activity detector based on dns traffic analysis, in: 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), 2016, pp. 824–830. doi:10.1109/NGCT.2016.7877524.

[11] K. Alieyan, A. Almomani, M. Anbar, M. Alauthman, R. Abdullah, B. Gupta, Dns rule-based schema to botnet detection, Enterprise Information Systems 15 (4) (2021) 545–564.

[12] M. Al-Kasassbeh, M. Almseidin, K. Alrfou, S. Kovacs, Detection of iot-botnet attacks using fuzzy rule interpolation, Journal of Intelligent & Fuzzy Systems (Preprint) (2020) 1–11.

[13] P. Barthakur, M. Dahal, M. K. Ghose, A framework for p2p botnet detection using svm, in: 2012 international conference on cyber-enabled distributed computing and knowledge discovery, IEEE, 2012, pp. 195–200.

[14] M. Injadat, A. Moubayed, A. Shami, Detecting botnet attacks in iot environments: An optimized machine learning approach, in: 2020 32nd International Conference on Microelectronics (ICM), 2020, pp. 1–4. doi:10.1109/ICM50269.2020.9331794.

[15] S. Y. Yerima, M. K. Alzaylaee, Mobile botnet detection: A deep learning approach using convolutional neural networks, in: 2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), 2020, pp. 1–8. doi:10.1109/CyberSA49311.2020.9139664.

[16] P. Torres, C. Catania, S. Garcia, C. G. Garino, An analysis of recurrent neural networks for botnet detection behavior, in: 2016 IEEE Biennial Congress of Argentina (ARGENCON), 2016, pp. 1–6. doi:10.1109/ARGENCON.2016.7585247.

[17] S. Alam, I. Traore, I. Sogukpinar, Current trends and the future of metamorphic malware detection, in: Proceedings of the 7th International Conference on Security of Information and Networks, SIN '14, Association for Computing Machinery, New York, NY, USA, 2014, p. 411–416. doi:10.1145/2659651.2659670.
URL https://doi.org/10.1145/2659651.2659670

[18] S.-H. Li, Y.-C. Kao, Z.-C. Zhang, Y.-P. Chuang, D. C. Yen, A network behavior-based botnet detection mechanism using pso and k-means, ACM Trans. Manage. Inf. Syst. 6 (1) (Apr. 2015). doi:10.1145/2676869.
URL https://doi.org/10.1145/2676869

[19] W. Chen, X. Luo, A. N. Zincir-Heywood, Exploring a service-based normal behaviour profiling system for botnet detection, in: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 947–952. doi:10.23919/INM.2017.7987417.

[20] W. Wang, Y. Shang, Y. He, Y. Li, J. Liu, Botmark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors, Information Sciences 511 (2020) 284–296. doi:https://doi.org/10.1016/j.ins.2019.09.024.
URL https://www.sciencedirect.com/science/article/pii/S0020025519308758

[21] H. Attak, M. Combalia, G. Gardikis, B. Gastón, L. Jacquin, D. Katsianis, A. Litke, N. Papadakis, D. Papadopoulos, A. Pastor, et al., Application of distributed computing and machine learning technologies to cybersecurity.

[22] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, Y. Elovici, N-baiot—network-based detection of iot botnet attacks using deep autoencoders, IEEE Pervasive Computing 17 (3) (2018) 12–22. doi:10.1109/MPRV.2018.03367731.

[23] S. Akcay, A. A. Abarghouei, T. P. Breckon, Ganomaly: Semi-supervised anomaly detection via adversarial training, CoRR abs/1805.06725 (2018). arXiv:1805.06725.
URL http://arxiv.org/abs/1805.06725

[24] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks (2014). arXiv:1406.2661.

[25] S. García, M. Grill, J. Stiborek, A. Zunino, An empirical comparison of botnet detection methods, Computers Security 45 (2014) 100–123. doi:https://doi.org/10.1016/j.cose.2014.05.011.
URL https://www.sciencedirect.com/science/article/pii/S0167404814000923

[26] M. Rehak, P. Celeda, M. Pechoucek, J. Novotny, Camnep: multistage collective network behavior analysis system with hardware accelerated netflow probes, in: CERT FloCon Workshop, 2009.

[27] S. Garcia, M. Grill, J. Stiborek, A. Zunino, An empirical comparison of botnet detection methods, computers & security 45 (2014) 100–123.

[28] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, W. Lee, Bothunter: Detecting malware infection through ids-driven dialog correlation., in: USENIX Security Symposium, Vol. 7, 2007, pp. 1–16.

[29] A. H. L. Iman Sharafaldin, A. A. Ghorbani, "toward generating a new intrusion detection dataset and intrusion traffic characterization".

[30] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
URL http://arxiv.org/abs/1412.6980

[31] E. Mantas, et al., Practical autonomous cyberhealth for resilient micro, small and medium-sized enterprises, in: 2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom) (IEEE MeditCom 2021), Athens, Greece, 2021.