System Call Processing Using Lightweight NLP for IoT Behavioral Malware Detection

John Carter[1(✉)][0000−0003−4391−0269], Spiros Mancoridis[1][0000−0001−6354−4281], Malvin Nkomo[1][0000−0003−4066−7267], Steven Weber[1][0000−0002−9235−6922], and Kapil R. Dandekar[1][0000−0003−1936−2514]

Drexel University, Philadelphia PA 19104, USA
`jmc683@drexel.edu`

Abstract. Although much of the work in behaviorally detecting malware lies in collecting the best explanatory data and using the most efficacious machine learning models, the processing of the data can sometimes prove to be the most important step in the data pipeline. In this work, we collect kernel-level system calls on a resource-constrained Internet of Things (IoT) device, apply lightweight Natural Language Processing (NLP) techniques to the data, and feed this processed data to two simple machine learning classification models: Logistic Regression (LR) and a Neural Network (NN). For the data processing, we group the system calls into n-grams that are sorted by the timestamp in which they are recorded. To demonstrate the effectiveness, or lack thereof, of using n-grams, we deploy two types of malware onto the IoT device: a Denial-of-Service (DoS) attack, and an Advanced Persistent Threat (APT) malware. We examine the effects of using lightweight NLP on malware like the DoS and the stealthy APT malware. For stealthier malware, such as the APT, using more advanced, but far more resource-intensive, NLP techniques will likely increase detection capability, which is saved for future work.

Keywords: Natural language processing · machine learning · malware detection · Internet of Things.

1 Introduction

IoT devices have quickly progressed from novelty technology, for early-adopting users, to ubiquitous technology that is used by many in everyday life. These devices can range from user-input speech recognition devices such as the Amazon Alexa to various micro electro-mechanical sensor systems for data acquisition. In the wake of tightly-coupled enterprise platforms, Drexel University has championed a multi-modal open-source IoT platform named `VarIoT`, comprising long and short-range wireless communication protocols. This platform can be repurposed for various use-cases due to its highly customizable and modular interfaces. The `VarIoT` gateway offers support for Bluetooth Low Energy (BLE), WiFi, LoRa, Zigbee and LTE/5G cellular transceivers. Other supported interfaces include mmWave radar, ultra-high frequency (UHF), radio-frequency identification (RFID), and sigfox, in order to extend the available application areas. This enables a multitude of sensors to be deployed for various scenarios. The `VarIoT` platform is built on the open-source Thingsboard platform as the gateway for server-side data collection for advanced processing, visualization and storage [22].

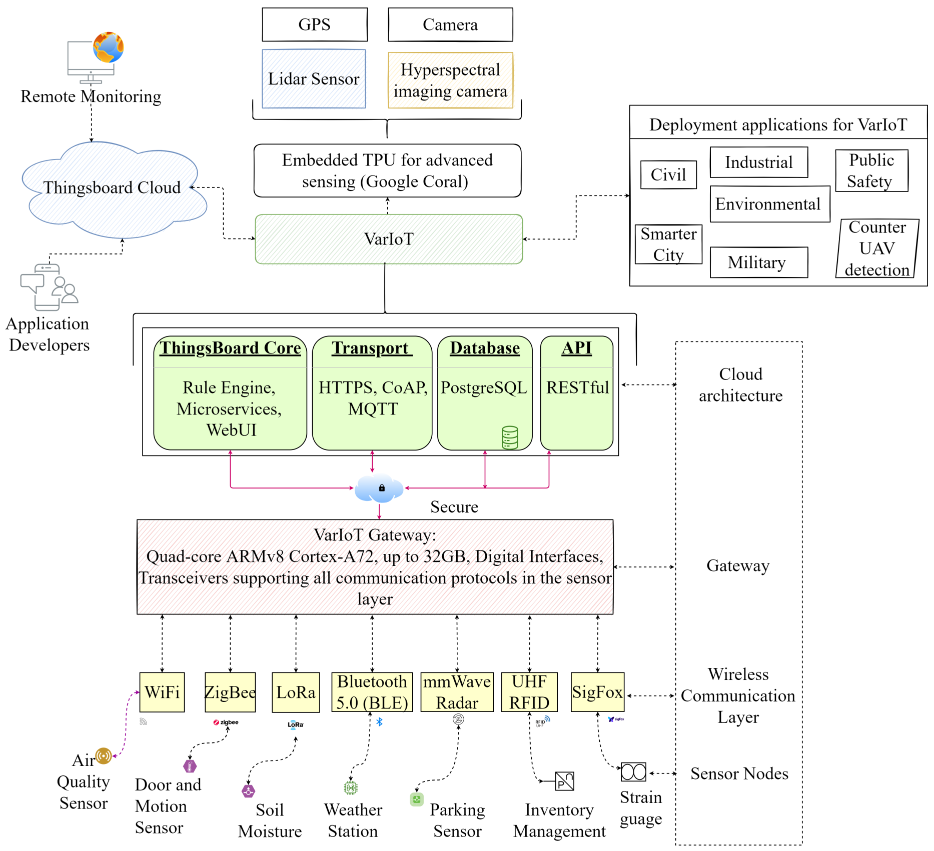

Fig. 1. The VarIoT platform architecture, which includes the IoT devices, the gateway with transceivers, capable nodes and server-side infrastructure.

In this work, we configure the VarIoT gateway to connect to a WiFi-enabled device as the sensor node to gather network data. The node used is the PurpleAir PA-II Air Quality Sensor. This framework is depicted in Figure 2. The Air Quality Sensor communicates with the `VarIoT` server once every minute, and is dormant on the network for the remainder of the time. It uses TLS encryption for its communication, which means all data sent to or from the sensor is not easily changeable, and only the TCP header data is visible from each packet.

This makes the communication less susceptible to many attacks in which the data is corrupted, such as Man-in-the-Middle attacks, but still leaves room for other attacks, such as DoS attacks, which will be explored further in Section 2 [21] [18].

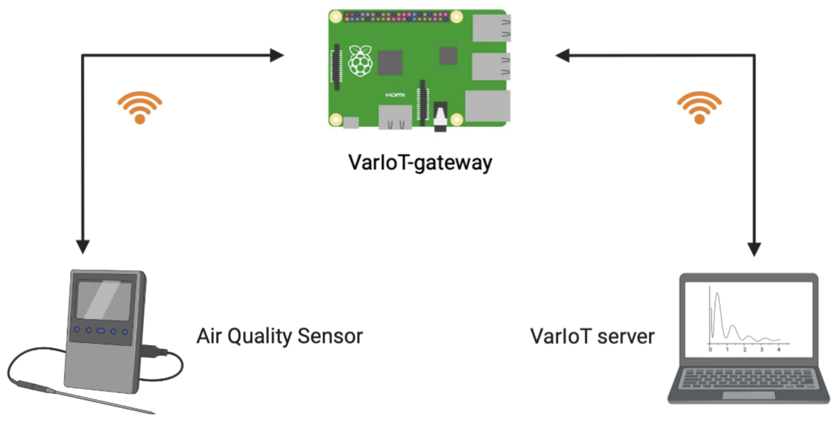

Fig. 2. The specific framework used in this work: an IoT device, the `VarIoT` server, and the gateway connecting them.

Our goal is to deploy malware onto the gateway and detect it using behavioral malware detection, in which kernel-level system calls issued on-device are transformed using lightweight NLP techniques and subsequently fed into two ML models.

Behavioral malware detection has been shown to be an effective method to detect malware that runs on devices, especially on resource-constrained Internet of Things (IoT) devices [2]. Machine learning (ML) models are often able to take a small amount of data, such as OS system call sequences or network traffic information, and produce an accurate classification of whether an observation is malware or not. However, this method relies heavily on what type of data is collected and how the data is processed, such that the data consists of the most explanatory features possible to use to train the models. Since the quality of the data fed to a machine learning model often dictates the model's performance, the data processing step can often be one of the most important steps in determining whether or not an effective model is produced [12].

One such data processing method is using lightweight NLP techniques to transform the data into a n-gram language model. n-gram language models make predictions using conditional word frequencies observed in the data [23]. Our work utilizes unigrams, in which the feature set comprises each observed system call issued on the `VarIoT-gateway`. We chose to use system calls because they provide a wider variety of observed behaviors on the device, since each process needs to issue system calls to accomplish its tasks. Although other data sources, such as network traffic, can provide useful behavioral data, system calls provide the largest breadth of device behavior. We also experimented with using system call bigrams, which means the feature set consists of each two consecutive system calls executed during the data collection period. This was not found to yield significantly better results than using the less intensive unigram method.

We show that while a model trained with a simple unigram representation of the data works well for noisier and more disruptive malware, it does not perform as well for stealthier malware. We believe classification problems for stealthier malware necessitate the use of more advanced NLP techniques, such as using a Recurrent Neural Network (RNN) or a Long Short-Term Memory (LSTM) [11], which we save for future work.

2 Previous Work

2.1 Behavioral Malware Detection

Behavioral malware detection seeks to detect malicious activity by learning the functionality and actions of malware during its execution [7] [4]. Today, behavioral malware detection often uses machine learning models to observe malware behavior using system or network data – in our case, kernel-level system calls. In one study by Hasan *et al.* [10], a variety of machine learning models were used, such as Support Vector Machines, Random Forests, Decision Trees, and Neural Networks.

2.2 IoT Background

IoT is extending its impact from simple devices of convenience, such as pedometers and voice-activated home virtual assistants, to integrated home security systems, remote-controlled medical devices, and self-navigating vehicles.

IoT devices are commonly the victims of malware due to their lax security, the number of connected IoT devices, which make them useful for distributed attacks, and the fact that they are rarely monitored directly by humans. One type of malware that often infects IoT devices is DoS malware, which compromise IoT devices with the intent of using them to overload servers and other hosts that cannot handle the load of millions of requests concurrently. The most notable DoS attack is Mirai, which disabled many servers, such as Amazon, GitHub, and Netflix [3] [14]. Another type of malware that can target IoT devices are APT (Advanced Persistent Threat) malware, which can passively surveil the state of the device and exfiltrate potentially sensitive information to remote command and control centers (C&C) [16].

A unique challenge for researchers who want to deploy behavioral malware detectors on IoT devices, or their corresponding ecosystems, is that such devices have computational, memory and energy limitations. Therefore, any malware detection technique that takes these constraints into account must be lightweight, such as the ones we describe in this work.

3 Malware

Two types of malware are deployed on the `VarIoT-gateway`, each of which represents malware families that are prevalent today. One is a quiet, stealthy type of malware, which we represent using an APT malware. The second type of malware is noisy and debilitating, which we represent using a denial of service attack. A DoS attack is one that is commonly carried out by IoT devices due to the scale of attack available. We explain each of these malware types in detail below.

3.1 Advanced Persistent Threat

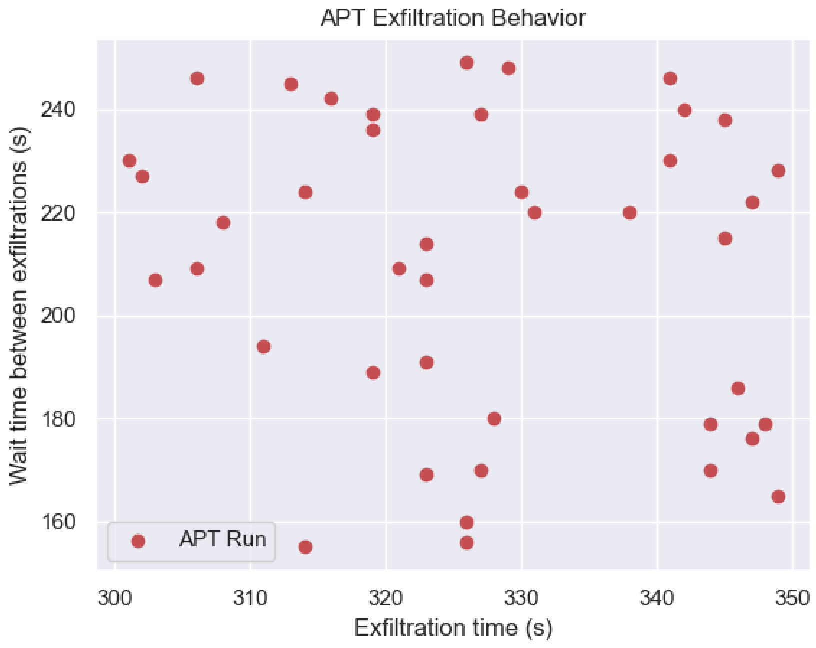

Fig. 3. The exfiltration behavior of the APT during the data collection process.

An APT is a type of malware often used for espionage and spying, sometimes by nation-states and other larger organizations [15]. In this work, the APT is designed to copy and exfiltrate the contents of files to a user-specified remote host.

It is controlled by a C&C server remotely and has the ability to be run in a *random mode*, which randomizes both the duration of each data exfiltration as well as the wait time between exfiltrations. An example of the duration and wait time parameters is depicted in Figure 3. In Figure 3, each data point refers to one APT run, where the x-axis represents the duration of APT exfiltration, and the y-axis represents the amount of time the APT sleeps before the next exfiltration.

Though APTs can be found "in the wild," these often are unusable for a number of reasons, such as that they report to C&C servers that are no longer active. In addition, these APTs are not yet advanced enough to have randomized behavior, which makes their detection an easier task. In contrast, the APT we use in this work is stealthy and much more difficult to detect, which will be discussed further in the Experimental Results section of this paper.

3.2 Denial of Service

Some IoT devices are now becoming sophisticated enough to use encryption for communication with servers. In the case of the Air Quality Sensor, each packet sent to or from the sensor is encrypted using TLS. This leaves fewer options for attackers to use to disable these devices. However, one remaining option is to conduct a Denial of Service attack, which does not require any information from the packet payload but only requires the packet header information, such as the source and destination IP address.

In this work, the Denial of Service attack uses the network utility `netwox` to conduct a TCP Reset Attack on the connection between the Air Quality Sensor and the VarIoT server. `netwox` is first downloaded onto the gateway by our malware using the standard `apt-get` procedure common on Linux machines, and then it is unpackaged and ready to attack the communication between the Air Quality Sensor and the VarIoT server.

A TCP Reset Attack listens to an ongoing TCP connection and then sends a spoofed packet with the "R" flag set to the victim, which will terminate the TCP connection [18]. The `netwox` TCP Reset Attack is known as `netwox` 78, which takes a device name (the desired interface), a pcap filter, and a spoofip parameter, which tells `netwox` how to generate link layer for spoofing [9]. `netwox` then sends many reset packets to the host specified, which disables the communication between the Air Quality Sensor and the VarIoT server.

4 Data Collection and Processing

4.1 Initial Data Processing

The raw data consists of system calls executed on the `VarIoT-gateway` during periods of benign behavior as well as during periods of malware execution on the device. The malware data was collected separately for each type of malware running on the device. In addition, before the benign data was collected, the operating system was reinstalled on the device, ensuring that the environment was free of malware. The data processing step involves grouping the collected data into segments that are more useful for behavioral malware detection, and is shown in detail in Figure 4. After the raw system calls are logged, they are grouped by timestamp using a user-specified window size.

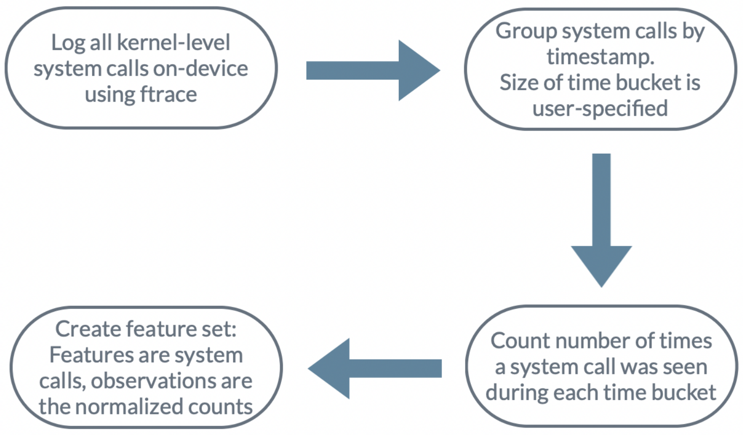
Log all kernel-level system calls on-device using ftrace
Group system calls by timestamp. Size of time bucket is user-specified
Count number of times a system call was seen during each time bucket
Create feature set: Features are system calls, observations are the normalized counts

Fig. 4. The data pipeline. This describes the transformation from ROM raw kernel-level system calls to a usable feature set for a ML model.

This window size can be thought of as a parameter that breaks up the total amount of data collection time into a user-specified number of buckets. Through experimentation, we set the window size to be 10 milliseconds, though this is a tunable parameter. We chose this small window size due to the rapidity of which the system calls are executed, such that that the windows could separate system calls from different processes as effectively as possible.

4.2 Data Transformation using NLP

We then chose to use a lightweight NLP approach of grouping the data into n-grams. Often this approach is applied to written text, in which words are grouped into either unigrams (sequences of one), or bigrams (sequences of two). Longer sequences can also be used, although these are used less frequently due to larger sequences becoming intractable. We used the bag-of-n-grams approach in the groupings [13] [17], where the value of n was user-specified [4]. This means that the feature set is composed of the number of observations of each n consecutive system calls in a particular time window. Through empirical analysis, we found that a value of $n = 1$ is optimal for both performance efficiency and detection efficacy in our dataset. Since we chose a value of $n = 1$, the feature set consists simply of the number of times each system call was observed during each time window. For example, using unigrams, one column of the data could be a system call such as `mutex_lock`, and each of the rows of that column will be the total number of calls to this system call for each time window. The number of observations of the system calls were then normalized using Term Frequency-Inverse Document Frequency (TF-IDF) [20], which normalizes the system call counts instead of using only the total number of times the system call was observed.

5 Experimental Results

Two typical machine learning models were used for evaluation of our data processing techniques: logistic regression and a shallow neural network. Both of these models are lightweight, which make them ideal for use on resource-constrained IoT devices. As shown in Figures 6-7, more available training data (depicted by the x-axis of both figures) yields better models. Although the amount of training data necessary to create useful classifiers can differ between types of malware, in general, as with many ML models, more training data yields a better model. Both models are tested using a larger testing set, comprised of 120 minutes of benign data and 120 minutes of malware data. We explain each of these models in detail below.

The metric we use for evaluating the efficacy of the models is Area Under the Curve (AUC), where the curve is the Receiver Operating Characteristic curve. This metric measures the ability of a classifier to differentiate between classes in the data, and is useful as a summary of the Receiver Operating Characteristic (ROC) curve. An example ROC Curve for the neural network is shown in Figure 5. AUC is an effective metric for evaluating classifiers, and there has been research that suggests it is actually preferable to overall accuracy for some problem domains [6].

Three types of malware are used for evaluation in this work.

1. The first is the stealthy APT malware, which was described in detail previously.
2. The second type of malware is a simple installation and uninstallation script which is responsible for repeatedly downloading `netwox`, unpackaging and installing it, and then removing it from the device. We chose to include this pseudo-malware because it shows how easily these simple ML models can detect the malware before any execution starts and without any execution occurring. This relies on the assumption that the device does not run automatic updates and if it were updated, the detector would not be running during that time. This rapid detection is especially important for zero-day attacks [5], since the only time a user might have to stop a malware attack from damaging their computer is to kill the download process before execution.
3. The third type of malware is the randomized `netwox`, which not only encompasses the installation/uninstallation process, but also executes the `netwox` TCP Reset Attack for a random duration of time.

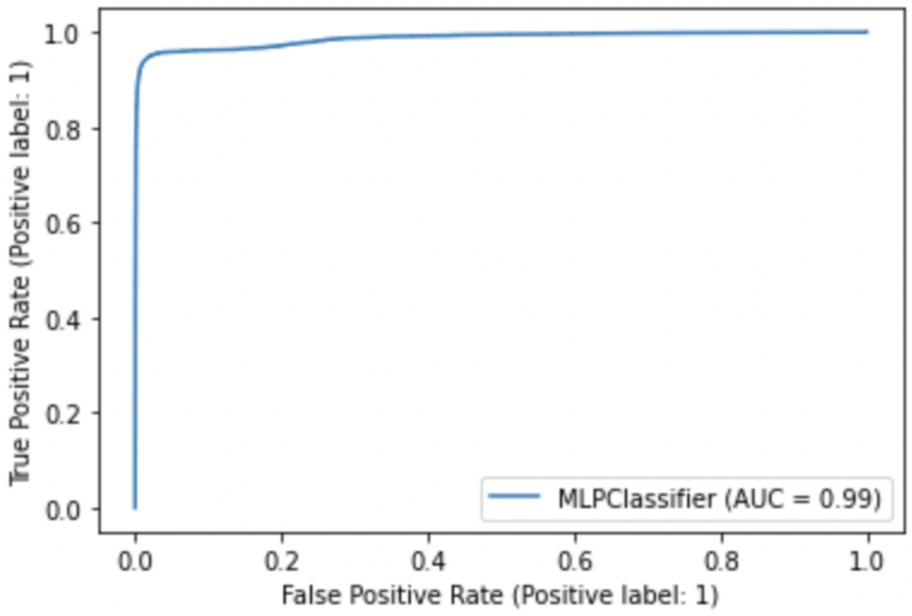

Fig. 5. An example ROC Curve using the neural network and 30 minute randomized-netwox data, as shown on the right side of Figure 7.

5.1 Logistic Regression

Perhaps the most lightweight, yet effective, machine learning model suitable for our task is logistic regression. In addition to being lightweight, it does not require much data for training, which also makes it ideal for our problem space. The results show that the LR model could easily detect the `netwox`-related malware, but struggled more to detect the APT.

5.2 Shallow Neural Network

Another lightweight ML model is a shallow neural network. In this work, we use a three layer NN that is provided off-the-shelf from scikit-learn called MLPClassifier [19]. This model is a three-layer neural network that optimizes the log-loss function. It uses L2 regularization ($\alpha = 1e{-3}$), a logistic sigmoid activation function, and the Adam optimizer with learning rate 1e$-$3. Each of the parameters are user-specified and were chosen through experimentation with the data.

As with the LR model, the NN is easily able to detect the `netwox`-related malware, but again struggled more to detect the APT.

It is interesting to note how the AUC values for both the LR and NN models follow the same trajectory and have essentially the same values for the `netwox`-related malware, with the only major difference being that the NN had marginally better results for the APT malware. Similarly, both models show a

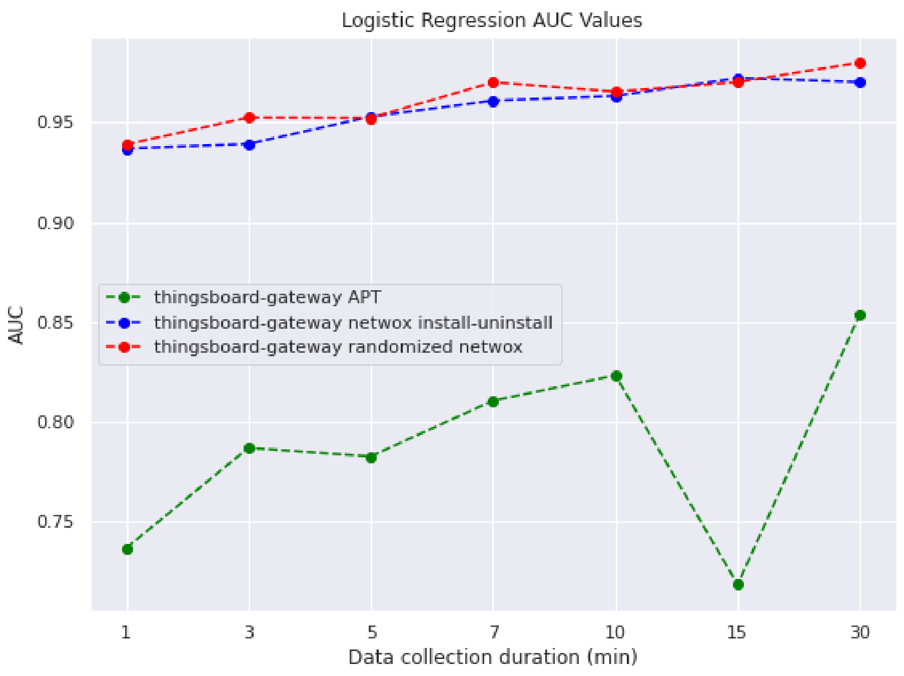

Fig. 6. Area Under the Curves for detecting the three flavors of malware on the `VarIoT-gateway` using a Logistic Regression model.

slight decrease in AUC using the 15 minute training data. This suggests that the slight AUC decrease has more to do with limitations from that particular training dataset rather than a model deficiency.

6 Conclusions

In this work, we built a simple, yet fully-functional, IoT testbed which is comprised of an Air Quality Sensor, a gateway, and an instance of a ThingsBoard server. Kernel-level system calls were logged on the gateway, which is configured specifically to connect to a `VarIoT` server. This gateway connects a common IoT device and a `VarIoT` server which relays information to users, which is shown in Figure 2. These raw system calls were collected during periods of strictly benign behavior as well as during periods of malware execution on the gateway.

The raw system calls were then transformed into usable features using a lightweight NLP technique. Specifically, we transformed them into a feature set of unigrams, which is a lightweight representation of the data and thus ideal for resource-constrained IoT devices. Lastly, two lightweight and efficacious machine learning classifiers were built and were successful in classifying malware, especially the `netwox`-related malware.

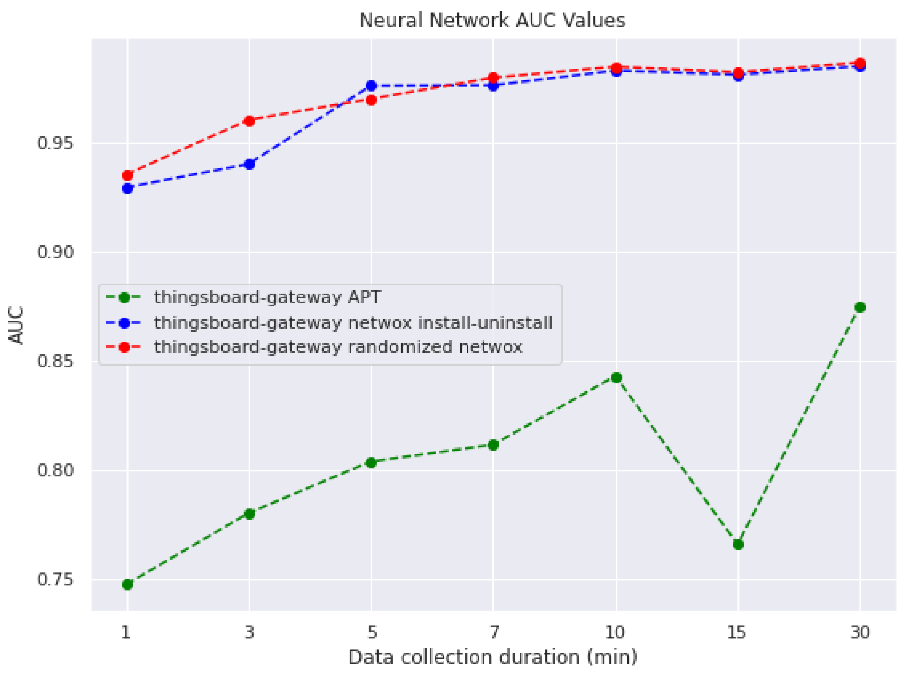

Fig. 7. Area Under the Curves for detecting the three flavors of malware on the `VarIoT-gateway` using a Neural Network model.

The ability of the classifiers to detect the installation/uninstallation malware is quite promising and useful because with only 1 minute of training data, both models were able to detect malware with greater than 90% Area Under the Curve. This finding is very useful from a user standpoint, because if the malware can be stopped early before it executes, the user has a chance to prevent malware from damaging their system. Likewise, both models were able to detect the randomized `netwox` just as well, which suggests that even if the data contains less obvious DoS behavior than downloading packages, the models will still perform well. However, the models were significantly less successful in classifying the stealthier APT malware. Because of the randomization of the APT's behavior as well as its much smaller system call footprint, it is harder to detect using only the lightweight NLP data representations used in this work.

6.1 Future Work

We would like to extend this work in two different directions. The first is by using more advanced NLP techniques, such as Recurrent Neural Networks, Gated Recurrent Units (GRU) [8], and Long Short-Term Memory (LSTM) [11]. We believe these techniques will work better for stealthier malware such as the APT and will result in more effective behavioral malware detection.

Secondly, we would like to explore this topic further using other IoT devices. In addition to other WiFi-enabled devices, we would like to work with devices that communicate using other protocols. These can include Bluetooth-enabled devices, which are ubiquitous, as well as devices that support protocols like LoRa, ZigBee, SigFox, UHF RFID and mmWave radar, which are both used by low-power and long-range IoT devices [1].

Since all of these devices are already natively supported by the `VarIoT` platform, we can replicate the work in this research and compare the effectiveness of the techniques shown here using devices that connect using a wide variety of protocols. This will not only further validate the work presented, but will also show the feasibility of using these methods for IoT devices that are less common and more often overlooked in security research.

Acknowledgments. The work was funded in part by Spiros Mancoridis' Auerbach Berger Chair in Cybersecurity.

References

1. Ali, A.I., Partal, S.Z., Kepke, S., Partal, H.P.: ZigBee and LoRa based wireless sensors for smart environment and iot applications. In: 2019 1st Global Power, Energy and Communication Conference (GPECOM). pp. 19–23 (2019). https://doi.org/10.1109/GPECOM.2019.8778505
2. An, N., Duff, A., Noorani, M., Weber, S., Mancoridis, S.: Malware anomaly detection on virtual assistants. pp. 124–131 (10 2018). https://doi.org/10.1109/MALWARE.2018.8659366
3. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., et al.: Understanding the mirai botnet. In: 26th USENIX security symposium (USENIX Security 17). pp. 1093–1110 (2017)
4. Aslan, A., Samet, R.: A comprehensive review on malware detection approaches. IEEE Access **8**, 6249–6271 (2020). https://doi.org/10.1109/ACCESS.2019.2963724
5. Bilge, L., Dumitraş, T.: Before we knew it: An empirical study of zero-day attacks in the real world. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security. p. 833–844. CCS '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2382196.2382284, https://doi.org/10.1145/2382196.2382284
6. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition **30**(7), 1145–1159 (1997). https://doi.org/https://doi.org/10.1016/S0031-3203(96)00142-2, https://www.sciencedirect.com/science/article/pii/S0031320396001422
7. Carter, J., Mancoridis, S., Galinkin, E.: Fast, lightweight iot anomaly detection using feature pruning and pca. In: Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing. p. 133–138. SAC '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3477314.3508377, https://doi.org/10.1145/3477314.3508377
8. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)

9. Du, W.K.: Tool 78: Reset every TCP packet, https://web.ecs.syr.edu/w̃edu/Teaching/cis758/netw522/netwox-doc_html/tools/78.html
10. Hasan, M., Islam, M.M., Zarif, M.I.I., Hashem, M.: Attack and anomaly detection in iot sensors in iot sites using machine learning approaches. Internet of Things **7**, 100059 (2019). https://doi.org/https://doi.org/10.1016/j.iot.2019.100059, https://www.sciencedirect.com/science/article/pii/S2542660519300241
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (nov 1997). https://doi.org/10.1162/neco.1997.9.8.1735, https://doi.org/10.1162/neco.1997.9.8.1735
12. Jain, A., Patel, H., Nagalapatti, L., Gupta, N., Mehta, S., Guttula, S., Mujumdar, S., Afzal, S., Sharma Mittal, R., Munigala, V.: Overview and importance of data quality for machine learning tasks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining. p. 3561–3562. KDD '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3394486.3406477, https://doi.org/10.1145/3394486.3406477
13. Kang, D.K., Fuller, D., Honavar, V.: Learning classifiers for misuse and anomaly detection using a bag of system calls representation. In: Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop. pp. 118–125 (2005). https://doi.org/10.1109/IAW.2005.1495942
14. Kolias, C., Kambourakis, G., Stavrou, A., Voas, J.: DDoS in the IoT: Mirai and other botnets. Computer **50**, 80–84 (01 2017). https://doi.org/10.1109/MC.2017.201
15. Lemay, A., Calvet, J., Menet, F., Fernandez, J.M.: Survey of publicly available reports on advanced persistent threat actors. Computers Security **72**, 26–59 (2018). https://doi.org/https://doi.org/10.1016/j.cose.2017.08.005, https://www.sciencedirect.com/science/article/pii/S0167404817301608
16. Li, S., Zhang, Q., Wu, X., Han, W., Tian, Z., Yu, S.: Attribution classification method of apt malware in iot using machine learning techniques. Sec. and Commun. Netw. **2021** (jan 2021). https://doi.org/10.1155/2021/9396141, https://doi.org/10.1155/2021/9396141
17. Liu, A., Martin, C., Hetherington, T., Matzner, S.: A comparison of system call feature representations for insider threat detection. In: Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop. pp. 340–347 (2005). https://doi.org/10.1109/IAW.2005.1495972
18. Mittal, A., Shrivastava, K., Manoria, M.: A review of DDOS attack and its countermeasures in TCP based networks. IJCSES **2**, 177–187 (11 2011). https://doi.org/10.5121/ijcses.2011.2413
19. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
20. Ramos, J.: Using TF-IDF to determine word relevance in document queries (01 2003)
21. Surya, S.R., Magrica, G.A.: A survey on wireless networks attacks. In: 2017 2nd International Conference on Computing and Communications Technologies (ICCCT). pp. 240–247 (2017). https://doi.org/10.1109/ICCCT2.2017.7972278
22. ThingsBoard - Open source IoT Platform: Thingsboard - open source iot platform, https://thingsboard.io

23. Wallach, H.M.: Topic modeling: Beyond bag-of-words. In: Proceedings of the 23rd International Conference on Machine Learning. p. 977–984. ICML '06, Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1143844.1143967, https://doi.org/10.1145/1143844.1143967