Edge2Guard: Botnet Attacks Detecting Offline Models for Resource-Constrained IoT Devices

Bharath Sudharsan*, Dineshkumar Sundaram†, Pankesh Patel*, John G. Breslin*, Muhammad Intizar Ali‡
*Confirm SFI Research Centre for Smart Manufacturing, Data Science Institute, NUI Galway, Ireland
{bharath.sudharsan, pankesh.patel, john.breslin}@insight-centre.org
†AVM Solutions, dinesh.kumar@avmsolutionsuk.com
‡ School of Electronic Engineering, DCU, Ireland, ali.intizar@dcu.ie

***Abstract*— In today's IoT smart environments, dozens of MCU-based connected device types exist such as HVAC controllers, smart meters, smoke detectors, etc. The security conditions for these essential IoT devices remain unsatisfactory since: (i) many of them are built with cost as the driving design tenet, resulting in poor configurations and open design; (ii) their memory and computational resource constraints make it highly challenging to implement practical attack protection mechanisms; and (iii) currently, manufacturers use simplified light protocol versions to save memory for extra features (to boost sales). When such issues and vulnerabilities are exploited, devices can be compromised and converted into bots whereby severe DDoS attacks can be launched by a botmaster. Such tiny devices are safe only when connected to networks with defense mechanisms installed in their networking devices like routers and switches, which might not be present everywhere, e.g. on public/free Wi-Fi networks. To safeguard tiny IoT devices from cyberattacks, we provide resource-friendly standalone attack detection models termed *Edge2Guard* (E2G) that enable MCU-based IoT devices to instantly detect IoT attacks without depending on networks or any external protection mechanisms. During evaluation, our top-performing E2G models detected and classified ten types of Mirai and Bashlite malware with close to 100% detection rates.**

***Index Terms*—Edge Intelligence, Intelligent Microcontrollers, IoT Security, Attacks Detection, IoT Botnet Attacks, DDoS.**

I. Introduction

Advances in semiconductor technology have reduced their dimensions and cost while improving the performance and capacity of small IoT chipsets. Methods from recent research [1]–[4] enable users to fit, deploy and execute advanced ML models [5]–[8] on quite basic chipsets that are highly resource-constrained. On the other hand, algorithms [9] are emerging to enable resource-constrained IoT devices to self-learn from evolving real-world data, producing futuristic IoT devices that can re-train themselves *on-the-fly* to produce better edge analytics results. Such cutting edge technologies and algorithms are not of much use when an IoT application powered by them gets easily compromised during cyberattacks like DDoS (Distributed Denial of Service) attacks. The DDoS packets generated from a single compromised device are sufficient to launch attacks with an enormous magnitude that could easily disrupt any IoT smart environment (see Fig. 1), corrupting a functioning IoT meta-system. Recent reports have estimated that over 50 billion devices will be connected to the Internet by 2025, over half of which may be vulnerable to multiple cyberattacks[1]. According to a Symantec report[2], it takes *only 2 minutes* to attack an IoT device, and in a Kaspersky Lab threat report[3], they were able to collect *121,588 malware samples* from IoT devices in 2018, $\approx$ 4 times more than in 2017.

Despite the existence of such high threats, the security conditions of widely deployed essential IoT devices like HVAC controllers, smart meters, security cameras, etc. remain unsatisfactory. This is because, to produce lower-power and -cost devices (feasibility), MCUs and small CPUs are used as the brains of such IoT devices. Since such tiny chipset devices are highly resource-constrained with only a few MB of memory (Flash, SRAM and EEPROM) and have limited computational power, it is highly challenging to implement practical attack protection mechanisms. Also, since such devices are built with cost as the driving design tenet, they have *poor configurations and open design*. Next, to save memory and simultaneously provide attractive functionalities to customers (to boost sales), manufacturers adopt simplified lightweight versions of protocols in their devices, making them susceptible to various attacks. Such issues make IoT devices the low-hanging fruit for attackers. More seriously, their vulnerabilities can be exploited to create botnets that launch severe DDoS attacks when commanded by an attacker/botmaster.

To safeguard tiny IoT devices, we cannot expect network-based attack detection mechanisms [10] on all external networks. For example, our smartwatch may connect to dubious free Wi-Fi networks in public places like shopping malls, when we arrive at the airport, etc. and get attacked by bots or malicious devices from such insecure networks. Hence, there is a pressing need for a defense mechanism that can execute on memory and power-constrained MCU-based IoT devices, without impairing their lifespan or jeopardizing their functionality.

In this paper, we provide *Edge2Guard* (E2G) models to alleviate the above cyber-security issues. When E2G models are deployed on IoT devices, they continuously monitor network traffic data to detect malware attacks in real-time. We achieved almost 100% accurate detection rates using E2G.

[1]https://threatpost.com/half-iot-devices-vulnerable-severe-attacks/153609/
[2]https://docs.broadcom.com/doc/istr-22-2017-en
[3]https://securelist.com/new-trends-in-the-world-of-iot-threats/87991/

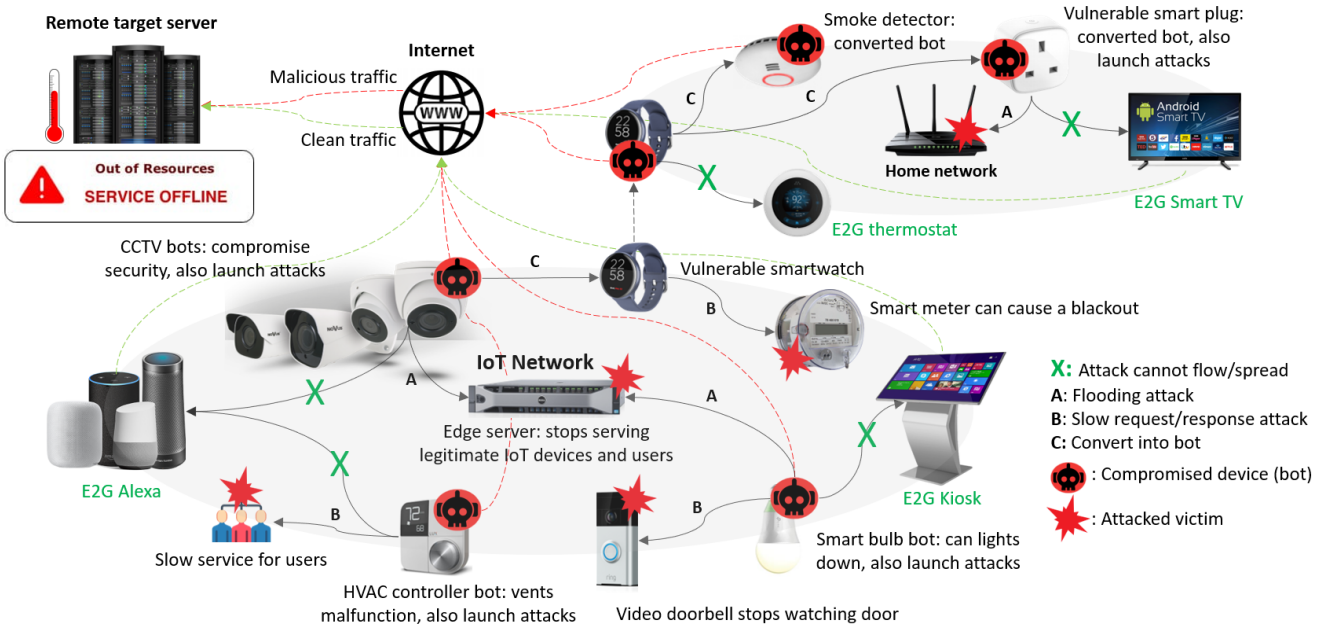

Fig. 1. Attack flow in a typical IoT smart environment; vulnerable devices are converted into bots, the rest are attacked: E2G models can execute offline on resource-constrained IoT devices to detect malware like Bashlite/Gafgyt and Mirai that turns connected devices into remotely controlled bots.

Unlike other approaches, our models detect attacks without depending on networks (standalone) or any external protection mechanisms (offline), and they can be executed on a wide range of MCU-based IoT devices without imposing computational pressure and also without disturbing device routine (resource-friendly design).

II. Background and Related Work

In Section II-A, we briefly describe how bots can affect an IoT smart environment by initiating various DDoS attacks with its operational flow/sequence in Section II-B. In sections II-C & II-D, we brief the standard and ML-based studies that aim to detect botnets, IoT-related anomalies, malware attacks in order to maintain a risk-free smart environment.

A. DDoS Attacks

Using any type of attack from the DDoS family, the botmaster attempts to obstruct legitimate IoT devices from accessing their network services. As shown in Fig. 1, a hacker can create a network of bots (a botnet) by initially compromising one IoT device (e.g. a CCTV). Then, the CCTV bot, disrupting its standard routine (thereby compromising security), converts vulnerable devices in the network into bots and also launches a DDoS attack on other nearby devices and remote servers when commanded.

The DDoS attacks launched by tiny IoT devices like smart speakers [11], smart plugs, smart meters, etc. are more challenging to detect since the attackers forge malicious packets leveraging the device's firmware design flaws or bugs in their communication protocols. Also, as shown in Fig. 1, besides being affected by DDoS attacks, compromised IoT devices (bots) are used by the botmaster as a local assistant tool during DDoS attacks. Hence the current *dumb security levels* of such devices allows them to be controlled, resulting in establishing a distributed botnet. For example, within 20 hours of the Mirai malware being released, 65k IoT devices were compromised and turned into bots, and were used to launch massive DDoS attacks targeting remote servers. In the following, we outline the two major categories of IoT DDoS attacks.

Flooding attacks. A massive numbers of disguised network packets are injected to exhaust the victim's resources and network bandwidth. A common type of attack is where the attacker employs a large number of repetitive communication requests, eventually filling the victim's buffer, restricting victims from accepting new messages and requests from legitimate devices. ICMP flooding, UDP flooding, DNS flooding are common examples of this attack. In the following, we present commonly found attacks:

– ***Protocol exploitation attacks*.** The attackers exploit specific features or implementation bugs to exhaust the victim's network resources. For example, the three-way handshake feature in TCP can be used to disrupt legitimate IoT devices by ACK and PUSH flooding. Here, initially, SYN messages are sent *en masse* to the edge server, and the bot which sent it refuses to react to the server response messages, causing servers to persistently wait for non-existent ACK messages. This leads to exhausting of the limited buffer queues of the server, leaving no room for processing new connection requests.

– ***Reflection flooding attacks*.** Using the bots, forged request packets with a manipulated source address are sent to the edge server. Since the spoofed packets are indistinguishable from the legitimate ones of IoT devices (victims), massive response packets are directed to the victim. **Amplification flooding attack** is a type of reflection attack where bots fraudulently cause a server to generate a tremendous volume of response

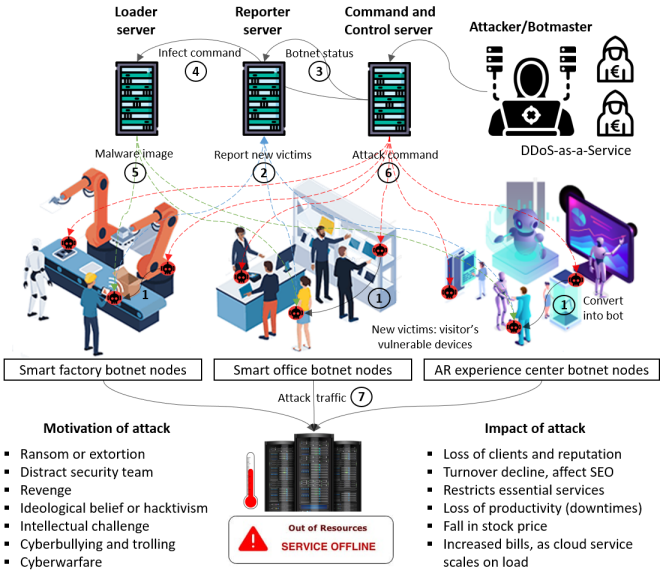

Fig. 2. Overview of malware communication and its basic components.

packets to a victim with limited requests. Attackers usually combine those two types of attacks to turn small queries into much larger payloads, then use it to bring down the IoT devices.

– ***Slow request/response attacks.*** The attacker's bot in the IoT network holds the communication channel by spoofing high-workload requests or responses, resulting in high server resource consumption, preventing legitimate requests from going through. Differentiating this attack from the usual traffic is difficult, and its mitigation process is cumbersome as a single bot (any tiny IoT edge device like a smart doorbell) in the network is sufficient to perform this attack.

B. Botnet and the IoT Zombie Armies

Malicious actors use our daily IoT devices as their powerful and mysterious playground since such devices are vast in numbers and diverse in locations. Their primary goal is to gather botnets to serve their nefarious purposes, ranging from spam and advertisement fraud to DDoS. In this section, we present the IoT botnet landscape and the reasons for its success.

Multitudinous IoT-powered DDoS armies are rapidly assembled by prevalent malware families like Mirai and Bashlite. The majority of such DDoS-malware comprises of two parts: the bot part coded in C, and its infrastructure. Using Fig. 2, we explain the botnet and its infrastructure. In any such smart environment, if there is a bot deployed, as shown in the circled ①, it starts exploring the IP space (at a max rate of 128 connections/sec) to find and convert vulnerable devices into bots. After successfully compromising a new device, it gains shell access, then in the next step circled ②, the new victim's characteristics such as IP address, port, login credentials, etc., are transmitted to the *Reporter server*. Then, in the step circled ③, the botmaster connects to the report server using a Tor circuit to investigate the features of the newly acquired devices. After this, the botmaster infects it by sending the device IP from the report server to the *Loader server*, which is the step circled ④. In the next step circled ⑤, depending on the victim's hardware architecture, via TFTP or wget, the appropriate binary is downloaded and executed. In the step circled ⑥, new and existing bots listen to instructions from the *Command and Control server* (C&C), and every 60 secs, heartbeat packets are transmitted between C&C and bots (servers directly controlled by the botmaster). When commanded, attack traffic is generated by an enormous number of bots (compromised IoT devices functioning in smart factories, experience centers, etc.), through which target servers are attacked in the final step circled ⑦.

C. DDoS Attack Defense Techniques

To protect IoT devices, in the following, we outline standard techniques that detect attacks before they occur.

IP traceback. Performing packet filtration closer to the attack source (to avoid impacting other networks) is essential to reveal the actual bot and path followed by attack packets. Widely used IP traceback techniques can be used to assist in packet filtration. A link testing technique was presented in [12] that performs a recursive search, where the link testing path starts from the router closest to the victim device and ends at the source router of the identified attacker. In [13], a hash-based IP traceback approach by using practical packet logging was presented, which can be used to defend against botnet DDoS and reflection-based attacks.

Entropy variations. To detect slow request or response attacks, users can find the difference in the entropy values (flow-header features and behavioral features) of attack packets at the source or destination IP address and their corresponding ports [14]. In [15], short term entropy was employed for early attack detection and long term entropy for attack classification. This approach can be deployed to identify *protocol exploitation flooding attacks*.

Intrusion detection and prevention systems (IDS/IPS). In an IoT architecture, IDS can be deployed on any layer, such as in the cloud layer [16] for gathering alerts from edge sensors, and can then correlate and analyze the alerts. In the firewall layer [17] on SDN switches, when malicious activity is detected, IPS can prevent intrusions via packet drops, resetting connections, or blocking the attacker's traffic. In attack detection studies, various methods [18, 19] focus on detecting the early steps of propagation and communication with the command and control server. Other articles look at host-based [19, 20] and network-based approaches [18, 21]–[23]. In [24], the hierarchical taxonomy of botnet detection approaches was presented.

D. ML Techniques to Detect Botnets

A wide variety of ML algorithms are available to detect attacks in IoT environments. Article [25] presents an OC-SVM detection mechanism for application-layer DDoS attacks. Honeypots [26] detect botnet DDoS attacks by capturing device malware installation attempts using unsupervised methods. Another ANN-based method [27] accurately discovered several application-layer DDoS attacks. Methods to detect botnet

activities within consumer IoT networks was presented in [28]. N-BaIoT [10], an autoencoder, was proposed to detect anomalous network traffic from compromised IoT devices (bots). The study [29] presents a CNN-based DDoS attack detection and warning system.

The techniques in section II-C are executed at various IoT architecture levels, i.e., the IP traceback and entropy variations are at the network layer, and IDS are on the cloud and in switches. Similarly, the above and other ML approaches [30] build models based on *attack packets* to predict or classify DDoS attacks, which are then deployed on different networking devices such as routers, honeypots, switches and firewalls. From the surveys by [24, 31], many on-device methods were left uncited, and offline (network independent) methods for resource-constrained IoT devices were not mentioned at all. In conclusion, our E2G models can execute offline on MCU-based IoT devices to detect attacks in any IoT environment with high accuracy, without imposing computational pressure, and also without disturbing device routines.

III. Proposed Edge2Guard Design

To provide a standalone attack-detecting mechanism for resource-constrained MCU-based IoT devices, we follow the ML-based approach, where we select a dataset, then train to produce resource-friendly attacks detecting E2G models, followed by extensive evaluation[4]. In the rest of this section, we provide the steps involved in designing E2G models, and explain how to deploy and execute them on tiny IoT devices.

A. Dataset and IoT Devices

We select and use the N-BaIoT [10] dataset as it contains data patterns (raw data in the *pcap* packet capture format using port mirroring) for normal and attack traffic from nine commercial IoT devices. The attack traffic was collected by infecting devices with authentic botnets from the Mirai and Bashlite families, whose traits and attack impacts were explained in Section II-B. For each data packet, the 115 features provided in the dataset that we use for training are the traffic statistics (packet context) extracted over several temporal windows (behavioral snapshots of when packets arrive). These 115 features are derived from five sets of 23 features extracted from 100ms, 500ms, 1.5sec, 10sec, and 1min time windows that are most recent. In the dataset, for each device, under each attack class, 5 subclasses represent the type and class of traffic. Hence, it contains 80 files with attack traffic data, 9 files (one for each device) with benign (regular) traffic data, and the overall dataset contains around 7 million instances.

B. Exploratory Data Analysis

We used a PCA dimensionality reduction method to mathematically reduce the 115 features into two features, which we plot and visualize in Fig. 3 in order to explore the patterns and find out trends between the malicious and benign traffic data. From Fig. 3, we can notice from the plots of the Provision PT-737E and PT-838 camera models that they both have similar

[4]Code available at https://github.com/bharathsudharsan/Edge2Guard

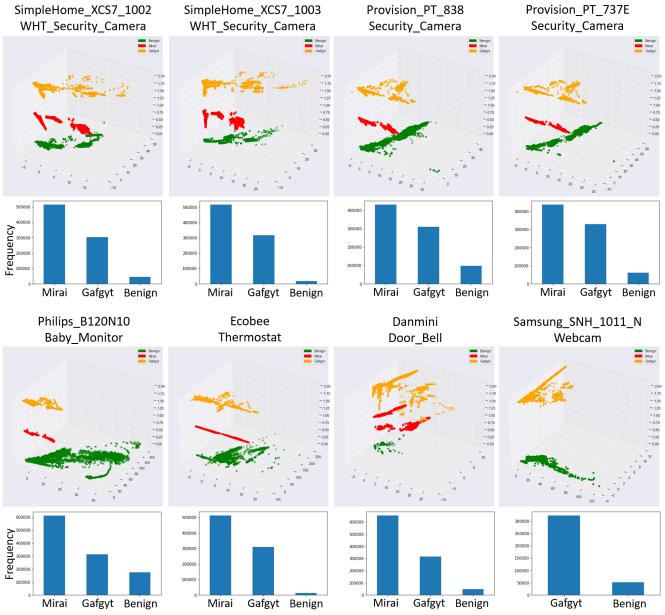

Fig. 3. 3D scatter plots for EDA: PCA was used to reduce and plot the 115 features of main classes Mirai, Bashlite, and benign (regular) traffic for various IoT devices. The bar graphs show the class count.

traffic patterns. Also the plots of SimpleHome 1002 and 1003 camera models have similar patterns. From this, we can infer that both Mirai and Bashlite malware behave the same way for devices from the same brand. Therefore, one E2G model is sufficient for same brands of device. Next, we can notice from the plot for a baby monitor that it contains benign data (regular traffic plotted in green) that is spread across the plot, indicating that frequent activities are performed on the baby monitor compared to other IoT devices. We also noticed that two devices contain traffic data for only one family of attacks, which was the reason for absence of 10 attack traffic data files (80 files were available instead of 90). From the bar graphs in Fig. 3, it is clear that the dataset is highly imbalanced (unequal distribution of classes) with a 1:13 ratio of normal:attack traffic. Hence we need to pre-process the dataset before training in order to obtain unbiased E2G models. A more detailed report that includes data statistics, variable types, 2D visualizations of features, data profile of each malware type, etc. are available in our repository.

C. Data Pre-processing

We pre-process the data to group it into four categories, using which, for each device, we train multiple supervised learning classifiers that can detect attacks by accurately differentiating them from regular traffic. The group names starting with *all data* consists of the entire imbalanced dataset with 65,00,000 instances (data rows) for attack traffic, 500,000 instances for normal traffic. The E2G models trained using *all data with 3 classes* should detect and classify the main class name, i.e., whether the input data is Mirai or Bashlite or benign, and the model trained on *all data with 11 classes* should classify exactly which subclass the input data belongs to (each subclass is a member of the Mirai or Bashlite families). For group names

TABLE I
COMPARING PERFORMANCE OF VARIOUS ATTACK DETECTION E2G MODELS FOR DANMINI DOORBELL.

Model	Acc	Recall	Prec	F1	Kappa	MCC
Random Forest	1.0000	0.9999	1.0000	1.0000	0.9999	0.9999
Decision Tree	0.9998	0.9997	0.9998	0.9998	0.9997	0.9997
K Neighbors	0.9980	0.9935	0.9980	0.9980	0.9960	0.9960
Ridge Regr	0.9969	0.9958	0.9969	0.9969	0.9936	0.9936
iForest	0.9700	0.96	1.00	0.98	0.6546	0.6977
OC-SVM	0.9300	0.93	1.00	0.96	0.0453	0.1058
Ada Boost	0.9245	0.9202	0.9340	0.9216	0.8392	0.8522
QDA	0.6834	0.8271	0.8491	0.6724	0.4799	0.5712
Naive Bayes	0.6585	0.3543	0.7312	0.5410	0.0693	0.1829
Linear SVM	0.4204	0.3930	0.4682	0.3959	0.0762	0.1060
LOF	0.1400	0.85	0.09	0.17	0.0182	0.0912
Logistic Regr	0.0486	0.3333	0.0024	0.0045	0.0000	0.0000

starting with *under sampled*, we balance the data by under sampling data rows belonging to the majority attack traffic class to obtain a balanced dataset with 360,410 instances for normal traffic and 639,590 for attack traffic. Similar to the above groups, the *under sampled data with 3 classes* and *under sampled data with 11 classes* trained models should classify the family names and family members respectively. When using any of the above four groups, we follow a 70-30 training-testing split and used all the 115 features since malware can attack the device at different time intervals.

D. E2G Models Design and Evaluation

The processed data can be used to build E2G models in three different approaches. In the first approach, we propose training a generic E2G model using the entire dataset that can detect attacks on any IoT device. Second, we can build one model for each category of device, i.e. one E2G model for cameras and another for doorbells. Third, we build separate models for each IoT device using the processed benign and attack traffic data of the corresponding devices. All of the above approaches have their merits and demerits. For example, in the first, the size of the E2G model will be larger than the available MCU memory because model size increases with the count of devices. Also, the dataset for the other existing thousands of devices needs to be collected and used for training, which is not feasible. In the second, accurate attack detection cannot be guaranteed for devices whose traffic data were not exposed during training. So we implement a third approach where the resultant E2G model for each device is small in size due to using device-specific training data.

E2G Supervised Learning Models. Before training various types of models for each IoT device, we initially took the pre-processed data of the Danmini doorbell and trained multiple supervised learning models, and present its performance in terms of accuracy, recall, precision, F1 score, Kappa, and Matthews Correlation Coefficient (MCC) in Table I. This result was produced when evaluating using 30% unseen data. Due to the non-linear nature of the dataset, the Random Forest (RF) and Decision Tree (DT) classifiers performed the best, and Logistic Regression produced the least accurate results. We did not use Deep Autoencoders, CNNs, or ANNs like in previous methods since we aim to train and deploy attack-detecting models on tiny IoT devices and not on networking devices like routers and switches that have better resources.

TABLE II
F1 SCORE OF THE TOP-PERFORMING RF AND DT E2G MODELS WHEN TESTING USING IOT DEVICE DATA.

Device	All data with 3 classes		Under sampled data with 3 classes		All data with 11 classes		Under sampled data with 11 classes	
E2G Model	RF	DT	RF	DT	RF	DT	RF	DT
Danmini Doorbell	1.0	1.0	1.0	1.0	1.0	0.86	1.0	0.57
Ecobee Thermostat	1.0	0.99	1.0	1.0	0.99	0.92	0.98	0.77
Ennio Doorbell	1.0	0.99	1.0	1.0	0.99	0.94	0.98	0.98
Philips B120N10 Baby Monitor	1.0	1.0	1.0	1.0	0.99	0.85	0.98	0.87
Provision 737E Cam	1.0	1.0	1.0	1.0	0.99	0.78	0.98	0.85
Provision 838 Cam	1.0	1.0	1.0	1.0	1.0	0.79	1.0	0.87
Samsung SNH 1011 N Webcam	1.0	1.0	1.0	1.0	0.99	0.89	0.99	0.99
Simple Home XCS7 1002 Cam	1.0	0.99	1.0	1.0	1.0	0.91	1.0	0.64
Simple Home XCS7 1003 Cam	1.0	0.99	1.0	1.0	0.99	0.91	0.97	0.84

E2G One-Class Classification Models. Malware keeps evolving, so the E2G models deployed on devices need to be updated (via OTA) with models that are trained using the attack traffic data for the new malware. There is a high chance of such updates being released *after* the device has been compromised. To avoid this, we approached this problem using one-class learning approaches, as we had used in other recent work [32]. To produce E2G One-Class Classification (OCC) models, we trained only using benign data since, as briefly described earlier, it is not feasible to track hundreds of new malware forms and to collect their attack traffic data by infecting and observing the device. Instead, in the OCC approach, we train models just using the device-specific benign data and consider other hundreds of types of malware attack data like Mirai, Bashlite, etc. as outliers (not shown during training) that need to be detected. We apply One-Class Support Vector Machines (OC-SVM), Isolation Forest (iForest), and a Local Outlier Factor (LOF) on the 115 features and show their evaluation results in Table I (arranged in the order of performance).

Top-performing E2G Models for IoT Devices. The iForest and OC-SVM OCC models perform reasonably well and also are more feasible to design (they do not require attack traffic data for any of the hundreds of malware family members during training), smaller in size (since only benign data is used), and easy to maintain (they do not require OTA updates). However, we choose RF and DT because, in attack detection use cases, even if a single instance of attack traffic is misclassified as benign, the device will get compromised. Hence, we proceed with the RF and DT E2G models as they accurately classified all 10 types of attacks. Here, as planned, for each IoT device, we train individual RF and DT E2G models for attack detection. All the individual models were evaluated using the same metrics shown in Table I. For brevity purposes, we only present

the F1 scores of the top-performing E2G RF and DT models in Table II. We provide other types of E2G models for each IoT device with detailed multiple metrics-based evaluations, their feature importance, and a confusion matrix in our repository.

E. E2G Models Deployment on MCU-based IoT devices

We saved the top-performing E2G models of each device in separate serialized Python pickle (.pkl) files (containing learned information like model weights) and provide it in our repository. Here, in four steps, we outline how to use our *RCE-NN* pipeline [1] to deploy and execute saved E2G attack detection models on the MCUs of IoT devices. The **Model conversion** is Step 1, where the E2G model is converted into a `FlatBuffer` file containing its direct data structures such as the information arrays of graphs, subgraphs, lists of tensors, operators, etc. Next, in Step 2 which is **Model translation**, since MCUs lack native file-system support, we cannot load and run models directly on such devices, so here we convert the model's FlatBuffer file into a *c-byte array* (the model in a char array) using the *xxd* UNIX command. Step 3 is **Model integration**, where the *c-byte array* of the E2G model is fused with the main IoT program, for which the devices' executable binaries are built using a technique from *RCE-NN* pipeline. Finally, in the **Model Deployment** (Step 4), the output from the last step should be used to flash the binaries of the attack-detecting E2G model onto the MCUs of IoT devices.

F. Comparing E2G Results with Other Methods

When comparing the performance of our E2G with papers that cite and use the N-BaIoT dataset, our RF and DT E2G models outperformed the models in articles from [25]–[29] by showing close to 100% detection rates. Next, when compared with the standard host-based [18]–[20] and network-deployed [21]–[23] approaches (from Section II-C), users can benefit more when they use our E2G models because of their standalone, offline attack detection capabilities that protect devices even when connected to *dubious* networks by mistake. Other benefits are that we have open-sourced the implementation and E2G models so that they can be readily used to reproduce the results from Tables I & II. With modifications, our code can be used to train models that can accurately detect various types of malware family members while being friendly enough (computationally light and not disturbing a device's routine) to be deployed on a wide range of tiny IoT devices.

IV. Conclusion

We presented E2G models that can be comfortably executed within the limited resource of tiny IoT devices and can classify malware attack traffic in real time, with the highest detection performance in comparison to existing approaches. In future work, we aim to deploy our E2G models on devices and test them using real-world data. We also plan to implement and add a resource-friendly prevention and alert mechanism that enables E2G models to perform post-attack detection actions.

Acknowledgement

This publication has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) under Grant Number SFI/16/RC/3918 (Confirm) and also by a research grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289_P2 (Insight), with both grants co-funded by the European Regional Development Fund.

References

[1] B. Sudharsan, J. G. Breslin, and M. I. Ali, "Rce-nn: A five-stage pipeline to execute neural networks (cnns) on resource-constrained iot edge devices," in *10th International Conference on the Internet of Things*.

[2] B. Sudharsan, P. Patel, J. G. Breslin, and M. I. Ali, "Ultra-fast machine learning classifier execution on iot devices without sram consumption," in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2021.

[3] B. Sudharsan, P. Patel, J. G. Breslin, and M. I. Ali, "Sram optimized porting and execution of machine learning classifiers on mcu-based iot devices: demo abstract," in *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems (ICCPS)*, 2021.

[4] B. Sudharsan and P. Patel, "Machine learning meets internet of things: From theory to practice," *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, 2021.

[5] B. Sudharsan, J. G. Breslin, and M. I. Ali, "Adaptive strategy to improve the quality of communication for iot edge devices," in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, 2020.

[6] B. Sudharsan, S. P. Kumar, and R. Dhakshinamurthy, "Ai vision: Smart speaker design and implementation with object detection custom skill and advanced voice interaction capability," in *2019 11th International Conference on Advanced Computing (ICoAC)*, 2019, pp. 97–102.

[7] B. Sudharsan, P. Patel, A. Wahid, M. Yahya, J. G. Breslin, and M. I. Ali, "Demo abstract: Porting and execution of anomalies detection models on embedded systems in iot," *Proceedings of the ACM/IEEE Conference on Internet of Things Design and Implementation (IoTDI)*, 2021.

[8] B. Sudharsan, S. Malik, P. Corcoran, P. Patel, J. G. Breslin, and M. I. Ali, "Owsnet: Towards real-time offensive words spotting network for consumer iot devices," in *IEEE 7th World Forum on Internet of Things*, 2021.

[9] B. Sudharsan, J. G. Breslin, and M. I. Ali, "Edge2train: A framework to train machine learning models (svms) on resource-constrained iot edge devices," in *10th International Conference on the Internet of Things*.

[10] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, 2018.

[11] B. Sudharsan, P. Corcoran, and M. I. Ali, "Smart speaker design and implementation with biometric authentication and advanced voice interaction capability," in *Artificial Intelligence and Cognitive Science*.

[12] H. Burch and B. Cheswick, "Tracing anonymous packets to their approximate source." in *LISA*, 2000, pp. 319–327.

[13] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer, "Hash-based ip traceback," *ACM SIGCOMM Computer Communication Review*, 2001.

[14] D. Boro and D. K. Bhattacharyya, "Dyprosd: a dynamic protocol specific defense for high-rate ddos flooding attacks," *Microsystem Technologies*.

[15] S. Oshima, T. Nakashima, and T. Sueyoshi, "Early dos/ddos detection method using short-term statistics," in *2010 International Conference on Complex, Intelligent and Software Intensive Systems*.

[16] S. Roschke, F. Cheng, and C. Meinel, "Intrusion detection in the cloud," in *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*. IEEE, 2009, pp. 729–734.

[17] P. Rengaraju, V. R. Ramanan, and C.-H. Lung, "Detection and prevention of dos attacks in software-defined cloud networks," in *2017 IEEE Conference on Dependable and Secure Computing*.

[18] M. Özçelik, N. Chalabianloo, and G. Gür, "Software-defined edge defense against iot-based ddos," in *2017 IEEE CIT*.

[19] D. H. Summerville, K. M. Zach, and Y. Chen, "Ultra-lightweight deep packet anomaly detection for internet of things devices," in *2015 IEEE 34th IPCCC*, 2015, pp. 1–8.

[20] H. Sedjelmaci, S. M. Senouci, and M. Al-Bahri, "A lightweight anomaly detection technique for low-resource iot devices: A game-theoretic methodology," in *2016 IEEE ICC*.
[21] H. Bostani and M. Sheikhan, "Hybrid of anomaly-based and specification-based ids for internet of things using unsupervised opf based on mapreduce approach," *Computer Communications*, 2016.
[22] I. Butun, B. Kantarci, and M. Erol-Kantarci, "Anomaly detection and privacy preservation in cloud-centric internet of things," in *2015 ICCW*.
[23] D. Midi, A. Rullo, and E. Bertino, "Kalis - a system for knowledge-driven adaptable intrusion detection for the internet of things."
[24] S. García, A. Zunino, and M. Campo, "Survey on network-based botnet detection methods," *Security and Communication Networks*.
[25] C. She, W. Wen, Z. Lin, and K. Zheng, "Application-layer ddos detection based on a one-class support vector machine," *International Journal of Network Security Its Applications*, vol. 9, pp. 13–24, 01 2017.
[26] R. Vishwakarma and A. K. Jain, "A honeypot with machine learning based detection framework for defending iot based botnet ddos attacks," in *2019 Conference on Trends in Electronics and Informatics (ICOEI)*.
[27] M. Asad, M. Asim, T. Javed, M. O. Beg, H. Mujtaba, and S. Abbas, "Deepdetect: Detection of distributed denial of service attacks using deep learning," *Comput. J.*, vol. 63, pp. 983–994, 2020.
[28] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*, 2018, pp. 29–35.
[29] C. Jinyin, Y.-t. Yang, K.-k. Hu, H.-b. Zheng, and Z. Wang, "Dad-mcnn: Ddos attack detection via multi-channel cnn," 02 2019, pp. 484–488.
[30] P. Xiao, W. Qu, H. Qi, and Z. Li, "Detecting ddos attacks against data center with correlation analysis," *Computer Communications*.
[31] E. Bertino and N. Islam, "Botnets and internet of things security," *Computer*, vol. 50, no. 02, pp. 76–79, feb 2017.
[32] B. Sudharsan, D. Sundaram, J. G. Breslin, and M. I. Ali, "Avoid touching your face: A hand-to-face 3d motion dataset (covid-away) and trained models for smartwatches," in *10th International Conference on the Internet of Things Companion*, ser. IoT '20 Companion, 2020.