# POSTER: Run-Time Detection of Malware in the Internet of Things (IoT)

Domhnall Carlin*, Philip O'Kane, and Sakir Sezer

Centre for Secure Information Technologies, Queen's University, Belfast, N. Ireland.
*d.carlin@qub.ac.uk

The evolution of a global network of internet-connected consumer devices, which had previously been the preserve of computers, has rapidly outpaced security considerations. This Internet of Things (IoT) allows tiny Linux-based devices, such as home heating controls and door bells, to become highly functional, both for the end user and the hacker. This offers a potential amplifying magnitude that, if compromised, could make unparalleled scope for high-volume cyberattacks. The website of prominent cybersecurity figure Brian Krebs was attacked with 620 Gbps of traffic in September 2016, blamed on a botnet composed of IoT devices. An attack using of up to 1.1 Tbps targeted the webhost and cloud service provider OVH of France. In 2017 Radware's IoT honeypot recorded 1,895 Permanent Denial of Service (PDoS) attempts, which has been called BrickerBot. Fast-forward to the 25th June 2019, and a new BrickerBot-type piece of malware named Silex began a similarly destructive campaign.

It's clear that there is very real potential for zombie armies of lightbulbs, coffee pots and fridges to be used as pawns in severe cyberattacks on anything from websites to critical infrastructure. There is an urgent need for novel research on increasing the security posture of such devices, and in providing solutions capable of being implemented in light-weight contexts.

**Operational codes (opcodes)** are machine language instructions that perform CPU operations. Dynamic opcode analysis has been successfully used for malware classification in the literature e.g.,[2], with high accuracy and low training and testing times. Opcode analysis has also been used to detect IoT malware with high accuracy[3, 1], albeit statically.

**Rationale**: The work conducted on IoT opcode analysis has all been static in nature. While obfuscation techniques may be limited in IoT, this is a technically trivial problem, and would represent the next step in IoT malware evolution. Classification should be between malicious and benign patterns to investigate the potential for detecting unauthorized code. Most of the work identified has employed deep learning architectures, which are computationally expensive and potentially beyond the capabilities of IoT and embedded devices. Similarly, several publications have used feature reduction techniques to remap feature vectors into dimensionally-reduced feature spaces.The present research seeks to overcome shortcomings in the current body of work by exploring dynamic opcode analysis on IoT platforms.

**Methodology**: A clean Debian 9 image was installed on Qemu v4.0.0 using full ARMv7 32bit emulation, with Linux kernel 4.9.0.9. Dynamic opcode tracing functionality was provided by using the TracerGrind plugin for Valgrind. The

count of each instruction was stored per sample as a comma-separated value (CSV) file, which provides the input dataset for machine learning algorithms. 78 instructions from the ARM instruction set became the feature vector. The exact same process was followed for the benign data. 401 malicious files were obtained from VirusShare.com. 209 benign files were taken from Debain. A second dataset was obtained (271 benign and 189 malicious)[3] allowing comparison with the present work.

**Results**: Experiment 1 compared the malicious (VirusShare) dataset to the benign (Debian). All malware was detected, and benignware was detected with 99.5% True Positive. Of the 610 samples compared, 608(99.6721%) were correctly classified. Experiment 2 compared Benign Vs Malicious from [3]: The results showed very high levels of accuracy, with 99.1304% correctly classified instances. In Experiment 3 the model from Experiment 1 was evaluated against the dataset of [3] as an unseen evaluation test set. The results show very high levels of classification ability. Accuracy was 99.8361% with a single misclassified instance. All data was pushed together as a 2-class problem in Experiment 4 to investigate the benefits of additional data on classification performance. Of the 1070 classification attempts, there were 1065 correctly classified instances (99.5327 %) and 5 incorrectly classified instances (0.4673%).

**Summary**: The present work has demonstrated that dynamic opcode analysis provides extremely positive discrimination ability in a two class problem in an IoT context. Our model shows much more favourable results than the static deep learning model in [3] (accuracy 99.1304% vs 98.18%) and their static Random Forest model (99.1304% vs 92.37%). This demonstrates the power of dynamic analysis, particularly with Random Forest. The present work used no feature selection or extraction, with the bare opcodes from the ARM Instruction Set as the only feature vector. As a result, the model build times for the experiments were extremely fast: between 0.06-0.33s.

This research demonstrates the first dynamic opcode analysis approach to malware detection on IoT platforms. While static analysis can be fast in the initial analysis, it is also highly prone to obfuscation techniques, which dynamic analysis may bypass. In this work, we have generated a highly accurate, fast, lightweight model, with generalisability to new unseen samples. This shows superiority to past research, when directly compared to the present work, indicating the value of the approach and results presented.

## References

1. Azmoodeh, A., Dehghantanha, A., Choo, K.K.R.: Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning. IEEE transactions on sustainable computing (2018)
2. Carlin, D., O'Kane, P., Sezer, S.: A cost analysis of machine learning using dynamic runtime opcodes for malware detection. Computers Security **85**, 138–155 (5 2019)
3. HaddadPajouh, H., Dehghantanha, A., Khayami, R., Choo, K.K.R.: A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting. Future Generation Computer Systems **85**, 88–96 (2018)