Poster: Analyzing, Comparing, and Detecting Emerging Malware: A Graph-based Approach

Hisham Alasmary◇, Aminollah Khormali◇, Afsah Anwar◇, Jeman Park◇,
Jinchun Choi◇‡, DaeHun Nyang‡ and Aziz Mohaisen◇

◇University of Central Florida   ‡Inha University

{hisham, aminkhormali, afsahanwar, parkjeman, jc.choi}@knights.ucf.edu
nyang@inha.ac.kr, amohaisen@gmail.com

Abstract—The growth in the number of Android and Internet of Things (IoT) devices has witnessed a parallel increase in the number of malicious software (malware), calling for new analysis approaches. We represent binaries using their graph properties of the Control Flow Graph (CFG) structure and conduct an in-depth analysis of malicious graphs extracted from the Android and IoT malware to understand their differences. Using 2,874 and 2,891 malware binaries corresponding to IoT and Android samples, we analyze both general characteristics and graph algorithmic properties. Using the CFG as an abstract structure, we then emphasize various interesting findings, such as the prevalence of unreachable code in Android malware, noted by the multiple components in their CFGs, and larger number of nodes in the Android malware, compared to the IoT malware, highlighting a higher order of complexity. We implement a Machine Learning based classifiers to detect IoT malware from benign ones, and achieved an accuracy of 97.9% using Random Forests (RF).

Index Terms—Malware; Android; IoT; Graph Analysis; IoT Detection.

I. INTRODUCTION

Background. As IoT finds new applications, IoT software security becomes of a paramount importance. IoT malware stands as one of the most significant threats to the security and stability of the Internet, and understanding IoT malware through analysis and detection is an essential problem to mitigate their security threats [1], [2]. The limited existing literature on IoT malware, and despite malware analysis, classification, and detection being a focal point of analysts and researchers [3]–[5], points at the difficulty, compared to other malware types. To understand IoT malware, we perform software analysis between IoT and Android samples, using graph properties obtained from CFG structures, and build a detection system of IoT malware utilizing those properties.

Overview. Starting with a new dataset of IoT malware samples, we pursue a graph-theoretic approach to malware analysis. Each malware sample can be abstracted into a Control Flow Graph (CFG) to extract representative static features of the application. As such, graph-related features from the CFG can be used as a representation of the software, and classification techniques can be built to tell whether the software is malicious or benign. Using the CFG graph constructs, We perform a comparative study of those graph-theoretic features in both types of software to highlight the CFG shift in IoT malware to Android application malware to uncover various similarities and differences. We similarly analyze the CFGs of 261 IoT benign samples and use that to build IoT classifiers from 23 different features extracted from the CFGs.

II. DATASET

Our IoT malware set is 2,874 samples, randomly selected from IoTPOT [6]. We also obtained a dataset of 2,891 Android malware samples from [7] for contrast. Finally, we manually created a dataset of benign samples from source files gathered from *OpenWrt.org* [8], and kernel files. To this end, we disassembled the IoT binaries, in the form of Executable and Linkable Format (ELF) files, as well as the Android Application Packages (APKs) using *Radare2* [9] to extract the CFG from the disassembly codes. Moreover, we used an off-the-shelf tool, NetworkX [10], for further graph analysis.

III. EVALUATION METRICS AND RESULTS

Evaluation Metrics. For our initial analysis of the various malware (Android and IoT) and benign samples, we use various standard algorithmic graph properties, including the number of nodes, the number of edges, the closeness, the number of components, etc. For the lack of space, we omit the definitions of those properties, and refer the interested reader to [5] for more details. In the following, we use a normalized version, from 0 to 1, of the closeness centrality.

A. Analysis

Android malware size differs from IoT malware significantly. Upon analyzing CFG of different samples belonging to each class, we observed that the Android and IoT malware samples have at least 28,691 and 367 nodes, and 33,887 and 577 edges, respectively. Figure 1 and Figure 2 represent the logarithmic scale for the number of nodes and edges, where the dynamic region of the CDF in Figure 1 is between 1 and 60 nodes, while the active region in Figure 2 between around 1 to 85 edges correspond to around [0.2–0.3] (about 10% of samples). This combined finding of the number of edges and nodes in itself is very intriguing: while the number of nodes in IoT malware samples is relatively smaller than that in Android malware, the number of edges is higher. This is striking, as it highlights a simplicity at the code base (smaller number of functions) yet a higher complexity at the flow-level (more
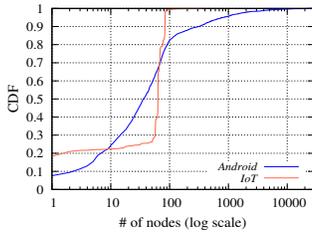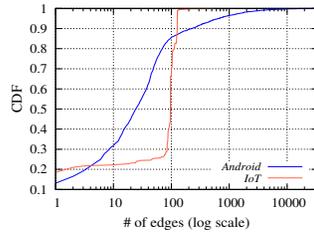
Fig. 1: Log scale for nodes
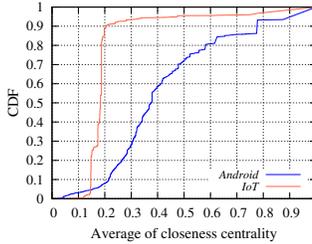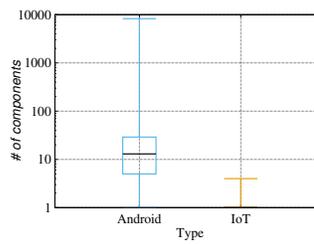

Fig. 2: Log scale for edges

Fig. 3: Closeness centrality in the largest components.

Fig. 4: Number of components: IoT vs. Android.

edges; calls between functions), adding a unique analysis angle to the malware that is only visible through CFG structure.

IoT CFG's are not only dense, but also well enmeshed graphs. Figure 3 and Figure 4 depict the CDF for the average closeness centrality and number of components, respectively, for both datasets. To reach this plot, we notice that around 5% of the IoT and Android have around 0.14 average closeness centrality. On the other hand, The same 80% of IoT samples have a closeness of less than 0.19, highlighting that the closeness alone with the value 0.2 can be used as a distinguishing feature of the two different types of the malware. The relatively higher value also highlight that IoT graphs are well enmeshed.

CFG analysis shed light on software anomalies. 3.23% of the IoT malware (93 IoT samples) have more than two components; i.e., most have one component that have file sizes from 56,500 – 266,200 bytes. On the other hand, 13.83%, or 400 Android samples, have only one component, where their size ranges from around 4,200 – 9,400,000 bytes. However, 2,491 samples (around 86.17%) have more than one component, which show that the Android malware often uses unreachable functions. We observe multiple components in Android CFG, which show the presence of multiple entry-points in the same program. These point the use of decoy functions with the aim to circumvent an analyst when trying to analyze the malware. The gap between datasets can be noticed, showing the new shift trend of the Android malware to the IoT devices.

B. Classification

As a result of the differences between IoT and Android malware across those graph features, it's natural to utilize those features of classification. To this end, we build a classifier for detecting IoT malware against bengin IoT samples. Upon extracting 23 different features from the CFGs for all samples (based on the betweenness centrality, closeness centrality, degree centrality, shortest path, density, # of edges, and # of nodes). Upon initial analysis, we obtained 2,347 IoT samples for classification against 261 benign samples. The results are

TABLE I: Classification results for the whole dataset, biased towards malicious samples. All results are percentages. False Negative Rate (FNR), False Positive Rate (FPR), False Discovery Rate (FDR), False Omission Rate (FOR), F1 score (F1), and Accuracy Rate (AR).

Method	Actual		FNR	FPR	FDR	FOR	F1	AR
LR	16.6	6.7	28.5	4.0	36.3	2.9	67.0	93.8
	9.5	228.0						
SVM	22.3	6.1	20.7	1.6	14.5	2.6	81.8	96.2
	3.8	228.6						
RF	23.6	3.1	11.6	1.1	9.6	1.3	89.5	97.9
	2.5	231.6						
CNN	22.9	3.0	1.3	11.5	1.4	1.4	98.7	97.6
	3.2	231.7						

reported in Table I using standard binary classification performance metrics. The results are obtained using 10-fold cross-validation. As shown, we obtained an accuracy of 97.87% using Random Forest classifier.

IV. CONCLUSION AND FUTURE WORK

We conduct an in-depth graph-based analysis of three different datasets to highlight the similarity and differences of IoT and Android malware, as well as benign IoT software towards detection of new IoT malware. Toward this goal, we extract the CFGs as an abstract representation to characterize IoT malware across different graph features, and highlight the shift in the graph representation from the IoT to the Android malware by tracing size (nodes, edges, and components). We observe decoy functions for circumvention, which correspond to multiple components in the CFG. Using those features, we built a classifier that achieved 97.9% of accuracy with 1.1% FPR and 11.6% FNR in IoT malware detection.

REFERENCES

[1] A. Gerber. (2018) Connecting all the things in the Internet of Things. Available at [Online]: https://ibm.co/2qMx97a.

[2] L. Harrison. (2015) The Internet of Things (IoT) vision. Available at [Online]: https://bit.ly/2SrowO1.

[3] A. Mohaisen, O. Alrawi, and M. Mohaisen, "AMAL: high-fidelity, behavior-based automated malware analysis and classification," *Computers & Security*, vol. 52, pp. 251–266, 2015.

[4] S. Shang, N. Zheng, J. Xu, M. Xu, and H. Zhang, "Detecting malware variants via function-call graph similarity," in *Proceedings of the 5th International Conference on Malicious and Unwanted Software, MALWARE*, 2010, pp. 113–120.

[5] H. Alasmary, A. Anwar, J. Park, J. Choi, D. Nyang, and A. Mohaisen, "Graph-based comparison of IoT and android malware," in *Proceeding of the 7th International Conference on Computational Data and Social Networks, CSoNet*, 2018, pp. 259–272.

[6] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoTPOT: A novel honeypot for revealing current IoT threats," *Journal of Information Processing*, vol. 24, pp. 522–533, 2016.

[7] F. Shen, J. D. Vecchio, A. Mohaisen, S. Y. Ko, and L. Ziarek, "Android malware detection using complex-flows," in *Proceedings of the 37th IEEE International Conference on Distributed Computing Systems, ICDCS*, 2017, pp. 2430–2437.

[8] Developers. (2018) Openwrt project. Available at [Online]: https://openwrt.org.

[9] Developers. (2019) Radare2. Available at [Online]: https://https://rada.re/r/.

[10] A. Hagberg, D. Schult, P. Swart, D. Conway, L. Séguin-Charbonneau, C. Ellison, B. Edwards, and J. Torrents, "Networkx. high productivity software for complex networks," *Webová strá nka https://networkx. lanl. gov/wiki*, 2013.

Analyzing and Detection Emerging Internet of Things Malware: A Graph-based Approach

Hisham Alasmary
hisham@knights.ucf.edu

Aminollah Khormali
aminkhormali@knights.ucf.edu

Afsah Anwar
afsahanwar@knights.ucf.edu

Jeman Park
parkjeman@knights.ucf.edu

Jinchun Choi
jc.choi@knights.ucf.edu

DaeHun Nyang
nyang@inha.ac.kr

Aziz Mohaisen
mohaisen@ucf.edu

INTRODUCTION

Background Internet of Things is one of the most recent technologies promising many applications.

Problem As IoT devices get connected to the Internet, their malicious behaviors becomes a substantial threat to critical pieces of infrastructure on the Internet, as well as to end users. Software is a common factor between different IoT devices running on different architecture, and malicious behaviors and intents of adversaries are embodied in malware. Analyzing and understanding how IoT malware differ from other closely related malware, as well as benign IoT software are important.

Idea Use Control Flow Graph (CFG) based features for analyzing, contrasting, and detecting IoT malware.

Previous approaches A lot of work on Android malware analysis, and an increasing number of studies on IoT malware analysis. No work yet on IoT malware analysis using CFG properties.

Our metrics Standard algorithmic graph properties, including # nodes, # edges, closeness, # components, etc. from CFGs. Use those properties to build a detector of IoT malware.

Machine Learning-based Detector

Dataset
- 2,347 malicious and 261 benign IoT samples, totaling to 2,608.
- 23 different CFG-based features in seven different groups, including betweenness centrality, closeness centrality, degree centrality, shortest path, density, # of edges, and # of nodes.

Key idea Detect IoT malware using CFG-based feature extracted from malware binaries. Evaluate detector using standard metrics
- False negative, false positive, false discovery, false omission, accuracy, and f-1 score.

Detection (using binary classifier)
- Applying different machine learning (LR, SVM, RF) and CNN deep learning methods on the whole dataset of the IoT malware.
  - Using 10-fold cross validation for all methods.
- RF achieves the best result with an accuracy of 97.9% with FPR (1.1%) and FNR (11.6%).
- Different classification results are reported

Key Results and Findings

- **Contrasting IoT to Android Malware:** Clear differences between IoT and Android CFG properties across betweenness, degree centrality, density, dimeter, radius, and shortest path counts! While smaller in size, IoT malware graphs are denser

Average of betweenness centrality

The average degree centrality

Density

Diameter
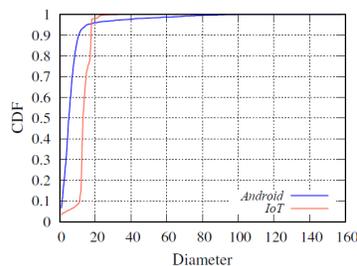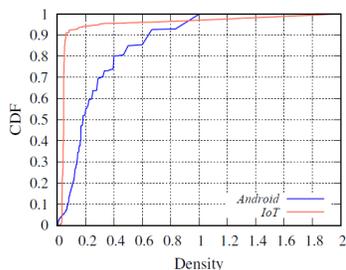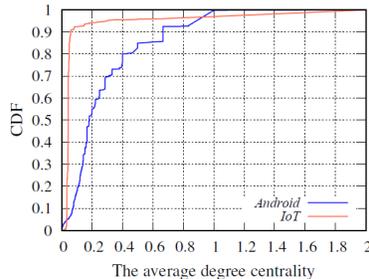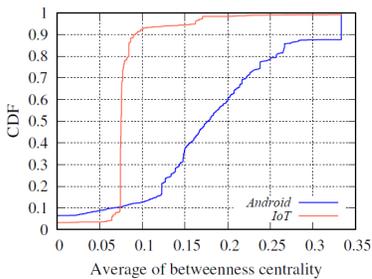
Results of IoT Malware Detection

Method	Actual		FNR	FPR	FDR	FOR	F1	AR
LR	16.6	6.7	28.5	4.0	36.3	2.9	67.0	93.8
	9.5	228.0						
SVM	22.3	6.1	20.7	1.6	14.5	2.6	81.8	96.2
	3.8	228.6						
RF	23.6	3.1	11.6	1.1	9.6	1.3	89.5	97.9
	2.5	231.6						
CNN	22.9	3.0	1.3	11.5	1.4	1.4	98.7	97.6
	3.2	231.7						

Radius

The average of shortest path
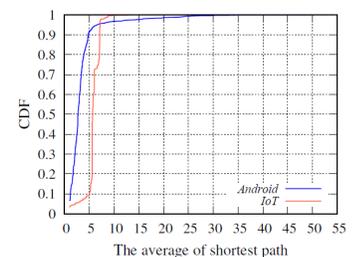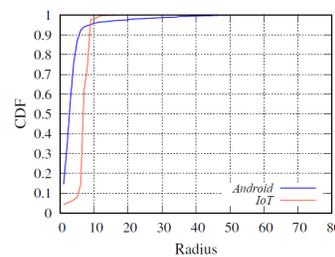
Conclusions

- Various interesting findings: multiple components in Android CFGs possibly due to decoy functions in the malware samples.
- IoT malware have less nodes and edges with various file sizes, which can be a sign of evasion techniques.
- A machine learning-based technique for IoT detection that achieves an accuracy of 97.9% using RF method.