DroidScour: an Android Application for IoT Device Data Collection

Brenda S. Schussler
Concordia University & UFRGS
Montreal, Canada
Porto Alegre, Brasil

Abdul Jamali
Concordia University
Montreal, Canada

Weverton Cordeiro
UFRGS
Porto Alegre, Brasil

Carol Fung
Concordia University
Montreal, Canada
carol.fung@concordia.ca

***Abstract*—The significant increase in the use of Internet of Things (IoT) devices in our society has introduced new challenges related to security and privacy. In this context, identification of IoT devices is necessary for the implementation of protection rules and control measures. Although many solutions have been proposed to identify and classify IoT devices, most rely on outdated or limited datasets, which compromises the effectiveness of these models. Therefore, an up-to-date and well-labeled IoT dataset is necessary for the public research community. This paper presents DroidScour, an Android app developed in Kotlin to collect network traffic, identify, and label IoT devices connected to a smartphone that acts as a hotspot. The IoT data collected by the smartphone is automatically uploaded to the Firebase services online, supporting training and future research. In this way, DroidScour allows the generation of high-quality real datasets of IoT devices, providing a data collection tool for researchers.**

***Index Terms*—Internet of Things (IoT), Network Traffic, Android Application.**

I. Introduction

The evolution of embedded systems has accelerated the development of Internet of Things (IoT) devices, transforming how everyday objects collect data and act in real time. The number of IoT devices is expected to more than double worldwide, from 19.8 billion in 2025 to 40.6 billion in 2034 [1]. Integrated into millions of homes, IoT devices automate tasks, manage appliances, and support security, health, and entertainment. However, limited built-in security exposes users to privacy risks and large-scale cyberattacks [2], making device identification and monitoring essential.

Recent efforts have used machine learning and datasets to identify and protect IoT devices. Meidan et al. [3] used supervised learning on traffic features from 17 IoT devices. Yin et al. [4] proposed a deep learning model trained on the UNSW [5] and YourThings [6] datasets, with 28 and 45 devices, respectively. Several other public IoT datasets were used in the literature for IoT identification, such as N-BaIoT [7] and IoTSentinel [8], with network traffic tracking of 9 and 27 IoT devices, respectively. In summary, existing data sets often lack sufficient labeling or device diversity, limiting the general applicability of models.

Creating a large and comprehensive IoT dataset requires an efficient tool that can collect traffic from IoT devices and label them with relevant information. To address this challenge, we designed and developed DroidScour, an Android application that can identify and label IoT devices, by collecting traffic data from these devices connected to the smartphone via hotspot. This hotspot-based approach provides practical deployment and control, but it naturally limits data collection to IoT devices capable of connecting via Wi-Fi, excluding Bluetooth or Zigbee devices. Nevertheless, this controlled environment enables the generation of well-labeled datasets, which is the concrete focus of this work. In this way, it becomes possible to create a large set of labeled data and train more effective identification and defense models, contributing to the IoT research community.

In this paper, we present DroidScour, a tool designed to run on Android smartphones, which allows network scanning to find IoT devices connected to the phone. The user can edit and save information about the devices and then collect their traffic data based on capture settings provided by the user. Unlike vendor-provided IoT apps, which are typically closed and device-specific, DroidScour offers a general-purpose and extensible solution for dataset creation, directly supporting research needs. To our knowledge, no other Android-based application integrates hotspot traffic capture, manual labeling, and automatic cloud upload into a single workflow. In this way, DroidScour is a convenient tool that can potentially be used by many researchers to create a large IoT dataset.

II. Architecture Overview

DroidScour is an Android application written in Kotlin[1], designed to run on smartphones with root privileges. DroidScour acts as a Wi-Fi hotspot and captures network traffic generated by IoT devices connected to it. This approach allows us to directly monitor the traffic of IoT devices in a real setting. The system design of DroidScour is presented as follows.

A. Architecture Design

Figures 1a and 1b illustrate the overall architecture and operational workflow of the DroidScour system, respectively.

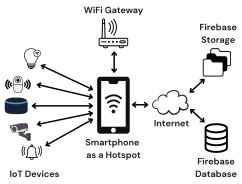
[1]https://developer.android.com/kotlin

This work was financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and grant #88887.954253/2024-00, and Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq) - Grants #314506/2023-3 and #405940/2022-0. The student Brenda Schussler was funded as a research intern in Canada by the Mitacs GlobalLink Award IT44130 during this project.

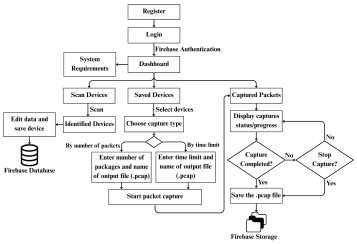
978-3-903176-75-1 ©2025 IFIP

As seen in Figure 1a, DroidScour uses the smartphone as a hotspot router, providing Wi-Fi connectivity to IoT devices. All packets originating from or received by these devices pass through the hotspot's network interface, which allows DroidScour to intercept them. The intercepted packets are stored on the smartphone and uploaded to Firebase when collection is done. This architecture makes DroidScour a practical, scalable and secure solution for collecting IoT traffic in real environments, making it easier to obtain reliable datasets.

To manage multiple devices concurrently, each packet capture runs in a separate thread, allowing parallel monitoring without blocking the main application workflow. In practice, Android hotspots typically support 10–15 devices simultaneously, which is sufficient for small-scale experiments. The scalability of the system is further supported by Firebase, which handles long-term storage and retrieval of captured datasets, ensuring that the tool can manage both local device traffic and cloud database growth effectively.



(a) DroidScour Architecture



(b) DroidScour Flow Diagram

Fig. 1: DroidScour overview (a) system architecture (b) flow diagram

B. Root Access and Termux

The application uses root privileges and Termux commands to scan connected devices and capture network traffic. To identify the devices connected to the hotspot, the application uses the *ip neigh* command via Termux to consult the neighbors table and retrieve the IP and MAC addresses of active clients. This approach provides accurate and real-time network discovery without the need for ping scans or external services. Packet capture is done using the *tcpdump* command, also run inside Termux with root access. When a device is selected for monitoring, the application starts *tcpdump* on the hotspot interface filtering by MAC address. This ensures the traffic to/from the device to be captured and saved in a .pcap file.

C. User Interface

The app offers a main dashboard screen (Figure 2) that allows the user to navigate to other screens, where they can:

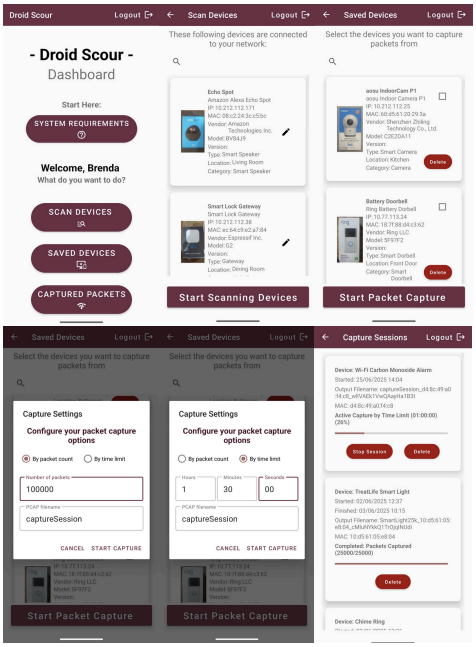


Fig. 2: Screenshots of the main interfaces of the DroidScour app.

scan the devices that are connected to the smartphone via hotspot, edit and save device information, select devices to start a packet capture and monitor the progress of the captures in real time (Figure 1b).

D. Firebase Services

The Firebase service is used to store collected data from multiple collecting smartphones. Authentication is added for secure access control. The Firestore Database stores user information, device data and other relevant references about the capture sessions. It also stores device images and captured packet files (.pcap).

III. Methodology

DroidScour's workflow is designed to enable the systematic and labeled collection of network traffic from IoT devices connected to the smartphone acting as a hotspot. The main stages of the process are described below:

Network Scan. By selecting and running the network scan feature, the app detects all the devices currently connected to the smartphone's hotspot and displays them in a list, identifying them by IP and MAC addresses (Figure 2).

Device Annotation. The scanned devices can, then, have their information: device name, description, vendor, model, version, type, category, location and device image edited and saved by the user. This manual labeling is essential for generating the device data with proper metadata needed to train machine learning models.

Packet Capture. On the saved devices screen (Figure 2), the user has access to a list of all devices for which information has been provided and saved. From the saved devices, with their respective data, the user can choose which devices he wants to capture packets from. Once the devices for capture

have been selected, the user must provide the name of the *.pcap* file in which the captured packets will be saved and the capture mode he wants to start. Capture sessions can be configured to occur by time limit (Figure 2) or by number of packets (Figure 2), information that must also be provided by the user according to the type of capture selected.

Live Monitoring. On the packet capture screen (Figure 2), the app displays information about capture sessions that have already been completed and those still in progress. For those that are still in progress, the progress is displayed in real time, through a progress bar, showing the count of packets captured or the elapsed time (depending on the capture type), allowing the user to monitor the data collection. The app also allows the user to stop an ongoing session and delete capture sessions.

Data Storage and Upload After the capture session ends, the generated *.pcap* output file is automatically saved to the device's local storage and then uploaded to Firebase Storage. Additionally, device and capture session information is updated in Firestore, ensuring organized and centralized management of the collected data. Since DroidScour captures traffic generated by end-user devices, privacy is a natural concern. The collection process is passive: the tool does not inject packets or trigger responses, it only monitors traffic that has already been exchanged. Users must explicitly label and consent to the capture. Furthermore, the focus is not on sensitive payloads, but on metadata and traffic patterns, to enable device classification.

A. Requirements and Setup

For the application and its features to work correctly, the smartphone must have:

- Root permissions to execute network scanning and packet capture commands;
- Termux environment installed with the necessary packages, as explained in the tutorial on the system requirements tab;
- Internet connection for authentication, data updates and synchronization with Firebase.

IV. Demonstration Experience

During the demo session at the conference, the audience will have the opportunity to follow the workflow and use of the DroidScour application in a controlled environment with IoT devices connected via hotspot to the Android smartphone.

The presenter's smartphone will be configured as a Wi-Fi access point, with real IoT devices connected to it. Then, participants will be able to observe the process of scanning the network performed by the app, listing the connected devices and their basic information. It will be possible to view and interact with the interface to edit or add detailed information about each of the devices. Then, a selection of devices for capture will be made, with settings by time limit or number of packets, which will allow monitoring of the progress of the captures in real time, with updates of visual indicators of collected packets and elapsed time.

After the capture, the .pcap files and the associated data will be automatically sent to Firebase Storage and Firestore. Then, participants will be able to view this synchronization and storage of the data in real time via the Firebase web dashboard.

V. Final Considerations

DroidScour is a practical and accessible tool for collecting network traffic data and labeling IoT devices connected to a smartphone via a hotspot. As an Android app with a simple and user-friendly interface, it allows users to identify devices connected to their hotspot, edit their information, and capture packets associated with specific devices. This approach allows for the accurate collection of traffic data with the potential to identify devices, vulnerabilities, and anomalous behaviors.

Therefore, this project contributes to the IoT research community by enabling the establishment of a large, labeled IoT traffic reference dataset with real traffic from different devices, which helps to overcome limitations of existing datasets that are often small, outdated, or not very diverse. In this way, other IoT researchers can use our tool by adding their own custom labels to the data as needed. This data can then be used to identify and mitigate potential security risks and vulnerabilities related to IoT devices connected to home networks, as well as for machine learning training purposes.

As future work, we intend to expand the collected dataset with contributions from the community, make the app publicly available to researchers and practitioners in the field, and employ machine learning techniques for the automatic detection of IoT devices using the dataset collected by DroidScour.

References

[1] L. S. Vailshery, "Number of IoT connections worldwide 2022–2034," Statista, May 26, 2025. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[2] R. Kumar, S. Singh, and P. K. Singh, "A secure and efficient computation based multifactor authentication scheme for intelligent IoT-enabled WSNs," Comput. Electr. Eng., vol. 105, Jan. 2023, Art. no. 108495.

[3] Meidan, Yair, et al. "Detection of unauthorized IoT devices using machine learning techniques." arXiv preprint arXiv:1709.04647 (2017).

[4] F. Yin, L. Yang, Y. Wang and J. Dai, "IoT ETEI: End-to-End IoT Device Identification Method," 2021 IEEE Conference on Dependable and Secure Computing (DSC), Aizuwakamatsu, Fukushima, Japan, 2021, pp. 1-8, doi: 10.1109/DSC49826.2021.9346251.

[5] Sivanathan A, Gharakheili H H, Loi F, Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics[J]. IEEE Transactions on Mobile Computing, 2019, 18 (8): 1745–1759.

[6] Omar Alrawi, Chaz Lever, Manos Antonakakis, Fabian Monrose ; SoK: Security Evaluation of Home-Based IoT Deployments, To appear in IEEE S, May 2019.

[7] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici. 2018. N-BaIoT–Network-based detection of IoT botnet attacks using deep autoencoders. IEEE Pervas. Comput. 17, 3 (2018), 12–22.

[8] Miettinen, M., Marchal, S., Hafeez, I., Asokan, N., Sadeghi, A.R. and Tarkoma, S., 2017, June. Iot sentinel: Automated device-type identification for security enforcement in iot. In 2017 IEEE 37th international conference on distributed computing systems (ICDCS) (pp. 2177-2184).