WiP: WDPKR: Wireless Device Profiling Kit and Reconnaissance

Tushar M. Jois
Johns Hopkins University
Baltimore, U.S.A.
jois@cs.jhu.edu

Claudia Moncaliano
Johns Hopkins University
Baltimore, U.S.A.
cmoncal1@jhu.edu

Khir Henderson
Morgan State University
Baltimore, U.S.A.
khhen2@morgan.edu

Aviel D. Rubin
Johns Hopkins University
Baltimore, U.S.A.
rubin@cs.jhu.edu

ABSTRACT

Internet-of-Things (IoT) devices are the building blocks to a stream of automated smart environments including residential homes, neighborhoods, schools, and office buildings. Due to their rapid growth, quick production cycles, and large market space, IoT devices are susceptible to undiscovered vulnerabilities and privacy concerns. These connected devices are diverse, feature-rich, and not standardized which makes their coexistence in a Smart Home environment difficult for researchers to study. We propose WDPKR, pronounced "woodpecker", which stands for Wireless Device Profiling Kit and Reconnaissance. WDPKR is a data collection and analysis solution designed for IoT device profiling in smart home environments to strengthen the analysis of 802.11 networks. In this work in progress paper, we present our design of WDPKR and the experimental results from testing our prototype. We also discuss a smart home testbed and our next steps for WDPKR. We believe that our design is feasible and useful for profiling IoT devices. We conclude that WDPKR's discovery, active traffic decryption, and location mapping features are strong and worth furthering, and highlight where WDPKR can continue to grow.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

datasets, internet of things, profiling, privacy

1 INTRODUCTION

Internet-of-Things (IoT) devices bridge the divide between the digital world and the physical one, linking information about a physical environment to computerized control. Consumers add devices to their home networks, creating smart homes of devices that communicate with each other to provide services such as monitoring or task automation. However, these IoT devices represent a treasure trove of data into their users' lives, and as such, are a target for malicious actors. Compounding the problem is a lack of standardization in APIs between major IoT vendors: a HomeKit-aware device, for example, may not recognize a device that operates only using Google Home. This means that devices can potentially have an inconsistent view of the network, making security breaches harder to detect.

One solution to this problem is the notion of a profile, a contract to adhere to a set of possible actions. This profile is enforced at the network-level. Any unexpected actions are a violation of the profile is thus considered a compromise of the device. Previous work in the profiling space requires the use of IoT device code in order to statically generate a profile for a device for a network [5]. However, this requires *a priori* knowledge of the code for a device, which may not be available: smart home platform manufacturers may be hesitant to release their code, or a legacy device may not support any system at all out-of-the-box.

To address these shortcomings, we present WDPKR, a proposed system that creates *dynamic* smart home profiles out of network data. We believe that this is possible because (1) IoT devices have a limited set of functionalities, which translates to a limited set of network-visible features, and (2) these network features do not change significantly over time. Thus, a "dynamic" profile for a device represents a consistency of its features over time. In a multi-vendor smart home, we can only consider a device's network features; vendor-specific information may not align across different device platforms. Additionally, WDPKR is designed to be a software platform operating on existing wireless hardware. The data processing facilities of consumer-grade networking hardware is limited, and thus WDPKR must be as performant as possible – the use of advanced neural networks would be infeasible.

In order to develop dynamic profiling, we first assume that the functionality of an (uncompromised) IoT device has a fixed set of operations it performs. Profiling these operations depends on an accurate set of fingerprints that capture a device's runtime behavior. These fingerprints require discovery and measurement of the smart home environment to understand the characteristics of the devices operating. As such, WDPKR requires all three, building up discovery tools to perform fingerprinting and utilizing fingerprints to create profiles dynamically. The research question of WDPKR, therefore, surrounds the ability to capture the state of devices solely from looking at dynamic traffic and that this state is unique enough to tie to a specific device.

This paper. This project is a work-in-progress. We have a tentative design, which we outline in Section 2. We then discuss our WDPKR prototype, which represents our current progress in assessing dynamic profiling, in Section 3. We conclude with a review of related work and a discussion of our next research steps. While there is still much work to be done using rigorous methods to create

dynamic profiles for IoT devices consistently, we believe that our methods are rigorous enough to accomplish our goal.

2 WDPKR DESIGN

We now discuss the proposed design of WDPKR, aiming to provide dynamic profiling for wireless devices in a smart home setting. We stress that the capabilities discussed in this section are those being considered for development. We implement an initial prototype of some of the capabilities described below and discuss their effectiveness in Section 3. We expect the methods to adapt as the research progresses. We first discuss device discovery, which helps us develop device fingerprinting, leading to our ultimate goal of device profiling.

2.1 Discovery

Smart home device discovery, broadly speaking, uses scanning capabilities to understand the structure of the IoT devices on a wireless network. Some vendors employ specific techniques to detect devices on a network, but these methods do not generalize beyond the platform for which they are designed. Additionally, legacy devices that do not communicate with any platform may not have accessible orchestration commands. As such, WDPKR relies on general techniques to perform device discovery.

Basic network scanning. WDPKR's first discovery tool (and its simplest) is a scan of the entire local area network. Such a scan combs through the entire IP subnet used by the smart home gateway, looking for devices that respond to port scanning. Devices that respond are added to a list of discovered devices, along with any associated open ports and identifying information returned from the scan. This type of scan is prevalent and is the baseline technique used in general for network asset discovery [1].

Wireless monitoring. WDPKR augments traditional scanning techniques with continuous monitoring of the smart home's wireless environment. WDPKR uses available wireless hardware in monitor mode, ingesting all available 802.11 frames in its vicinity. This allows it to identify traffic that might not be picked up on the local network explicitly, such as devices operating on different networks nearby.

Wireless monitoring can also help locate devices in physical space. Smart home IoT devices are most often fixed in their location once installed. For example, smart light bulbs typically stay in their physical sockets for their entire lifespan. Thus, WDPKR leverages signal strength (among other RF characteristics) in order to determine the distance to a device [6].

The increased use of mesh networking in smart homes improves the quality of data from wireless monitoring. Rather than having a single WDPKR instance at the gateway, each mesh node can run WDPKR. Each mesh node can independently collect information over the local network and over the air, and come to a shared understanding of the state of the network. A satellite node can detect devices that might be far away physically from the central node, and multiple nodes working in concert can pool distance information together to pinpoint the exact location of a device.

Uncooperative devices. Some devices may choose not to be visible on the smart home network. WDPKR still attempts to track these devices, as they may be surreptitiously stealing data or otherwise attempting to be malicious while undetected. Continuous wireless scans try to detect any device operating in range of WDPKR, regardless if they are associated with the home network or not. This will help identify devices attempting to spoof their location (as to appear to be a part of the network) or otherwise lie about their functionality.

2.2 Fingerprinting

We use the discovery methods discussed above to begin fingerprinting the smart home network. Much of the research in fingerprinting in the IoT space regards device classification (Section 4). In order to create a dynamic profile, though, knowing which devices exist in the environment is not enough. We must instead look at the behavioral patterns of a device, as it is these behaviors on which the final profile will be based. We consider a model in which WDPKR measures fingerprint data continuously, creating successive snapshots of a device's state over time. This continuous fingerprinting allows WDPKR capture a more holistic view of a device, allowing us to be more confident in a device's results.

Discrete flows. We plan to implement fingerprinting in WDPKR by looking at the patterns of network flows on a device. The tracking network flows between endpoints is used by stateful firewalls, which uses flows to identify valid communication. WDPKR plans on using this technique to determine the set of possible flows an uncompromised device as a fingerprint. During an initial "learning period", we plan to record flow data, such as source, destination, bandwidth utilization, and inter-packet timing and then perform fundamental statistical analysis to discern flows and their relevant properties. We believe that only basic methods are necessary since IoT devices have a limited set of operations that they support. We also eschew using a more intensive method, such as deep packet inspection, to keep the resource requirements to a minimum and minimize overhead.

Active probing. In order to passive monitoring of network traffic, WDPKR also performs probing of known devices. We plan on implementing basic port scanning to identify services listening on an IoT device's ports. In instances where the service is known (*e.g.*, `httpd`), WDPKR will collect additional information, such as library version or protocols supported, to add to its fingerprint for the device. We assume that the services supported by IoT devices are updated infrequently, as IoT device software functionality is tied to its physical functionality. Thus, active probing allows WDPKR to snapshot the configuration of a device at a given point in time.

Location ranging. WDPKR builds upon the discovery functionality in order to fix a device's location in space. Our current WDPKR prototype uses the free-space path loss equation metric in order to calculate distance. Our testing (Section 3) shows that while it is imprecise and can be subject to interference, it provides a rough estimate of where a device should be. Several location measures, taken throughout the day, can be merged to provide a range of potential locations for a device. The idea is that, in a smart home environment, there will be times when things are "still", *i.e.*, there

is no movement, and WDPKR can take an accurate location measurement. We additionally plan on exploring other metrics that can be used in distance calculations.

Confidence. Because our fingerprinting is dynamic, there will be a gap between the actual state of configuration of the device and the state that can be ascertained through WDPKR's methods. For example, the location ranging discussed above may be imprecise due to interference. However, we expect that the continuous location measurements will center on or near the true device location with our ongoing fingerprinting strategy. Periods of interference should, we think, create outliers that can be eliminated. The fingerprint can then be interpreted not as a specific configuration or state but as a gradient of possible configurations, some more likely than others. The variance around measurements then becomes inversely related to the confidence of that measurements. We plan on crafting our experiments around this fingerprinting gradient approach.

2.3 Profiling

Once we have built the discovery and fingerprint portions of WDPKR, the profiling component is simply a composition of the two. As discussed in Section 2.2, WDPKR aims to collect multiple fingerprint types continuously. Profiling acts as an enforcement layer over an existing fingerprint set for a discovered device. The profiling module intends to compare this known fingerprint against the current state of the network. WDPKR assumes that a compromised device has different network characteristics than normal, as the malicious actor will be trying to exfiltrate data or pivot into other devices on the smart home network. This can take the form of new communication flows to attack-controlled domains, a change in the configuration of services on the device, location spoofing, or more. If there are any differences between known patterns and current network activity, WDPKR knows a profile violation has occurred.

Aggregating the fingerprints is non-trivial. Our initial solution involves using a per-device fingerprint information database and using runtime data collection to match against database entries. WDPKR will employ fuzzy hashing [9] to compare known fingerprint state to the current environment. The "fuzziness" allows slight variations in fingerprint information to hash to the same value, but not major ones. This method does assume that the fingerprints collected by WDPKR do not change over time. We anticipate that such a change is infrequent and easier to verify from the user's side. For example, the fingerprint may change if a manufacturer enables new user-facing functionality on a device. In these cases, WDPKR can compute a new fingerprint for the device.

Range authentication. Our profiling approach allows us to perform some interesting enforcement behaviors. For example, WDPKR attempts to locate a device in space based on signal characteristics, among others. WDPKR can instruct the wireless router to adjust its power when communicating to only use the minimum signal strength necessary to communicate with a device. This adds physical-layer protection on top of the other security measures employed on the smart home network, a form of "range authentication" that prevents devices outside of the smart home from spoofing as being on the inside. This can be combined with techniques from distance bounding [4, 10] to add additional security.

3 INITIAL EXPERIMENTS

3.1 WDPKR Prototype

WDPKR is a command-line application written in Python using object-oriented software design techniques. It leverages sniffing and spoofing functionality from Scapy [3]. The WDPKR prototype aims to make 802.11 data collection simple and effective, perform active traffic decryption, locate devices and lays the groundwork for device fingerprinting and profiling.

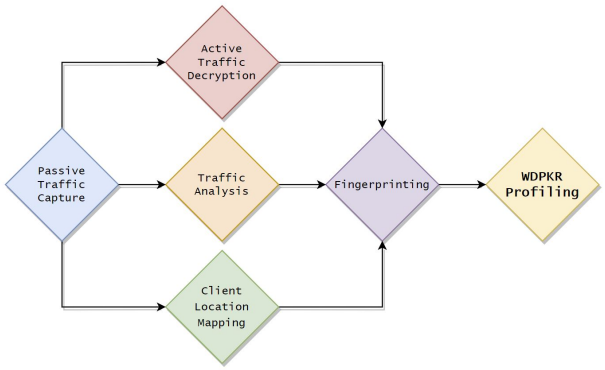

Figure 1: The current structure of our WDPKR prototype.

Passive traffic capture. WDPKR is capable of both live and file-based passive traffic capture. It uses Scapy for both functions, and users are able to specify the amount of time to capture packets or the number of packets they wish to capture. The live capture returns the packets captured to the user, and the file capture puts the packets in a PCAP file for easy reuse and further analysis. A user can also specify the interface on which to capture. By choosing a WLAN interface in monitor mode, WDPKR is able to capture 802.11 frames. Additionally, WDPKR provides a command-line interface and configuration file to help users easily take advantage of each of WDPKR's modules and features. See Figure 1 for a diagram of WDPKR's modules, Figure 2 for WDPKR's help menu, and Figure 3 for the configuration file. Figure 4 shows a sample PCAP file captured using WDPKR over 802.11 networks.

```
usage: wdpkr.py [-h] --config CONFIG [--ssid SSID] [-p PCAP] [-o OPEN]
                    [-n NAME]
                    {capture,attack,analyze,map}

WDPKR: Wireless Data Processing Kit & Reconnaissance

positional arguments:
  {capture,attack,analyze,map}
                        perform passive capture, active decryption,
                        analysis, or mapping

optional arguments:
  -h, --help            show this help message and exit
  --config CONFIG       wdpkr.ini file location
  --ssid SSID           ssid of network to attack for active decryption
  -p PCAP,              open a pcap file for analysis or mapping
  -o OPEN,              json file to load a previous WDPKR session
  -n NAME,              name of this session
```

Figure 2: WDPKR command-line help menu.

```
[default]
interface = wlan0

[gps]
# Configure latitude and longitude for location ranging.
latitude =  39.312285
longitude = -76.620257
altitude = 35.0  # in meters

[analysis]
# Select which modules to use and skip.
# Skip takes priority over Use.
use_modules = all
skip_modules = none

[psk]
# Pre-shared Keys for active traffic decryption.
WiFi SSID = 'password'

[known_devices]
# A lookup table for associating MAC addresses to known device names.
FFFFFFFFFFFF = broadcast
```

Figure 3: User configuration file.

Traffic analysis. WDPKR's analysis module is two-pronged: rudimentary and classifying. The rudimentary analyzer provides the user with a JSON representation of the wireless environment's state given pre-captured frames through a PCAP file. The Analyzer class processes features from packets sent over the 802.11 protocol using a packet parser. The JSON representation includes a collection of basic features for each device scanned on the network, including Vendor Name, Device Type, # of frames sent and received, and location data (see Client Location Mapping). Figure 5 contains sample output from our Analysis Module detecting an access point device and display its client list and AP Management Info extracted from the capture.

Active traffic decryption. This WDPKR module aims to actively decrypt encrypted packets captured across the network. By doing so, WDPKR gains real-time information about the devices it is able to *attack*. This module's first implementation of this uses the user-known WiFi password (WPA2-PSK). WDPKR 's Decryption class sends a deauthentication frame to simulate a disconnection between a device and the WiFi signal. It then captures the resultant 4-way handshake to get the Pairwise Temporal Key (PTK).

Client location mapping. This module calculates the distance between a device and the wireless client or access point (AP) using the free-space path loss equation (1).

$$\frac{P_r}{P_t} = D_t D_r \left(\frac{\lambda}{4\pi d}\right)^2 \tag{1}$$

Which states the ratio of the power received P_r, to the power transmitted, P_t, is equal to the distance of the transmitting and receiving antennas times the free-space path loss factor. The distance formula then becomes (2).

$$d = 10^{(27.55-20log10(f)-|\lambda|)/20} \tag{2}$$

The Client Location class uses Received Signal Strength Indicator (RSSI) metrics from captured packets and GPS coordinates provided by the user to calculate dbm in MHz and estimate the location range of the devices.

3.2 Experimental Setup

Testbed. The test data was collected in a laboratory setup at Morgan State University to replicate a typical home network: network devices connected wired or wirelessly to a standard consumer router (TP-Link AC1750). The network consists of 60 devices commonly used in home network settings. The testbed contains 26 common home appliances (refrigerators, fans, smart plugs, etc.), 4 smart assistants, 6 general-purpose devices, 3 smart hubs, 9 IP cameras, 7 media streaming devices, and two standalone sensors. The devices range from a variety of manufacturers to capture the broadest scope of the IoT market. These range from large regulated companies such as Apple and Samsung to small companies such as Govee and Gosund, and niche manufacturers with no easily found company name. The testbed uses automation and scheduling to reproduce routines typical in a small family home. This further increases the robustness and legitimacy of the test data. Other wireless networks used nearby add noise and extraneous data that one would find in the wild. The final goal is robust, secure legitimate data that allows for testing in various real-world scenarios.

A map of the testbed by signal strength is given in Figure 6. The IoT devices are clustered around three locations. Location 1, L1, is the testbed's primary functional portion, housing most of the IoT testbed. Location 2, L2, encapsulates the media devices. Location 3 contains the hub devices. This location data is saved in order to attest for similar signal strength readings since devices sharing locations share similar location readings.

Methodology. Network data is captured remotely via a local Raspberry Pi 4 with 4GB of RAM running Ubuntu 20.10 Desktop. Wireless network data is captured via Panda Wireless PAU06 300Mbps Wireless N USB Adapter placed in monitor mode. Our current WDPKR prototype captures all traffic wirelessly and saves packets iteratively to reduce ram and CPU usage. WDPKR can scan in high traffic areas without the use of excessive processing power.

3.3 Preliminary Results

Our WDPKR prototype collected over 260 MB of over-the-air data over a 6 hour time period, generating a robust IoT dataset. By running this packet data through the Analysis module, WDPKR discovered and identified every device in the testbed. The JSON representation of the rudimentary analysis includes data on the access point and client device types. Figure 7 is a screenshot of how a single device is represented in the output file post-analysis of a 15-minute capture. Its MAC address can decode the device. In this case, we have a Nooie Cam 360, recognized as a client device that has sent 5266 frames and received 8551 frames. The only device on its "Prob_list" is the testbed's TCP Link Router. This represents WDPKR's discovery phase, followed by a rudimentary analysis including organized and labeled data about the capture that can guide the project towards device fingerprints and profiling.

We tested WDPKR's client location mapping module against the testbed as well. As seen in Figure 7, it successfully calculated the average dBm values of each device, in this case, the Nooie Cam 360, and the distance between each device and the access point. We also estimate the distance between each device and the access point using the free space loss formula. Preliminary results show a potential correlation between dBm and distance values for devices

	Time	Source	Destination	Protocol	Length	Signal strength (dBm)	Info
1	0.000000	8e:49:62:c0:30:23	Broadcast	802.11	275	-35dBm	Beacon frame, SN=3921, FN=0, Flags=........, BI=100, SSID=Wildcard (Broadcast)
2	0.003520	Roku_c0:30:23 (8c:4…	Tp-LinkT_77:51:23 (98:da:c4:77:…	802.11	34	-33dBm	Request-to-send, Flags=........
3	0.004023	Roku_c0:30:23 (8c:4…	Tp-LinkT_77:51:23 (98:da:c4:77:…	802.11	34	-33dBm	Request-to-send, Flags=........
4	0.004331		Roku_c0:30:23 (8c:49:62:c0:30:2…	802.11	28	-5dBm	Clear-to-send, Flags=........
5	0.004895	Roku_c0:30:23 (8c:4…	Tp-LinkT_77:51:23 (98:da:c4:77:…	802.11	34	-33dBm	Request-to-send, Flags=........
6	0.005216		Roku_c0:30:23 (8c:49:62:c0:30:2…	802.11	28	-5dBm	Clear-to-send, Flags=........
7	0.006090	Roku_c0:30:23	Tp-LinkT_77:51:23	802.11	96	-33dBm	QoS Data, SN=3664, FN=0, Flags=.p.....T
8	0.006399		Roku_c0:30:23 (8c:49:62:c0:30:2…	802.11	28	-5dBm	Acknowledgement, Flags=........
9	0.010138	ArubaaHe_cc:92:e1	Broadcast	802.11	206	-73dBm	Beacon frame, SN=3703, FN=0, Flags=........, BI=100, SSID=MSU-Guest
10	0.018939	AmazonTe_5f:27:48	Tp-LinkT_77:51:23	802.11	44	-27dBm	QoS Null function (No data), SN=1, FN=0, Flags=...P...T
11	0.018989		AmazonTe_5f:27:48 (40:b4:cd:5f:…	802.11	28	-5dBm	Acknowledgement, Flags=........
12	0.026790	ArubaaHe_cc:81:20	Broadcast	802.11	226	-65dBm	Beacon frame, SN=518, FN=0, Flags=........, BI=100, SSID=eduroam
13	0.028501	ArubaaHe_cc:81:21	Broadcast	802.11	206	-65dBm	Beacon frame, SN=519, FN=0, Flags=........, BI=100, SSID=MSU-Guest
14	0.030462	ArubaaHe_cc:81:22	Broadcast	802.11	229	-65dBm	Beacon frame, SN=520, FN=0, Flags=........, BI=100, SSID=MSU-Secure
15	0.039463	Espressi_4a:7f:a1	Tp-LinkT_77:51:23	802.11	42	-37dBm	Null function (No data), SN=281, FN=0, Flags=...P...T
16	0.039514		Espressi_4a:7f:a1 (84:0d:8e:4a:…	802.11	28	-5dBm	Acknowledgement, Flags=........
17	0.039641	Espressi_10:b8:c1	Tp-LinkT_77:51:23	802.11	42	-33dBm	Null function (No data), SN=1585, FN=0, Flags=...P...T
18	0.039829		Espressi_4a:7f:a1 (84:0d:8e:4a:…	802.11	28	-5dBm	Acknowledgement, Flags=........
19	0.039930	Espressi_d1:cd:32	Tp-LinkT_77:51:23	802.11	42	-27dBm	Null function (No data), SN=2718, FN=0, Flags=...P...T
20	0.039967		Espressi_d1:cd:32 (10:52:1c:d1:…	802.11	28	-5dBm	Acknowledgement, Flags=........
21	0.040981	Espressi_7e:78:76	Tp-LinkT_77:51:23	802.11	42	-37dBm	Null function (No data), SN=3847, FN=0, Flags=...PR..T
22	0.041097		Espressi_7e:78:76 (84:f3:eb:7e:…	802.11	28	-11dBm	Acknowledgement, Flags=........
23	0.041375	Espressi_10:b8:c1	Tp-LinkT_77:51:23	802.11	42	-33dBm	Null function (No data), SN=1585, FN=0, Flags=...PR..T
24	0.041524		Espressi_10:b8:c1 (a0:20:a6:10:…	802.11	28	-5dBm	Acknowledgement, Flags=........
25	0.048437	Tp-LinkT_77:51:23	Broadcast	802.11	219	-5dBm	Beacon frame, SN=1231, FN=0, Flags=........, BI=100, SSID=CREAMHOMEv2.4
26	0.204353	8e:49:62:c0:30:23	Broadcast	802.11	275	-37dBm	Beacon frame, SN=3923, FN=0, Flags=........, BI=100, SSID=Wildcard (Broadcast)

Figure 4: PCAP collection.


```
    "08:3a:2f:11:33:a4": {
        "Vendor": "Unknown",
        "Device_type": "Access Point",
        "Frame_sent": 922,
        "Frame_received": 45,
        "Distance": "0.837 meters",
        "Avg_DBM": "-38.55097613882863",
        "Client_list": [
            "9c:a3:a9:3b:22:20",
            "9c:a3:a9:2d:ce:0c",
            "9c:a3:aa:69:c0:64",
            "9c:a3:a9:3b:25:f5",
            "9c:a3:a9:3e:10:95"
        ],
        "AP_management_info": {
            "Frequency": "2412MHz",
            "Authentication": "00:0f:ac (Ieee 802.11) PSK",
            "Channel": "14",
            "Cipher": "00:0f:ac (Ieee 802.11) AES (CCM)",
            "SSID": "NVR083a2f1133a4"
        }
    },
```


Figure 5: Analysis module, access point identified.

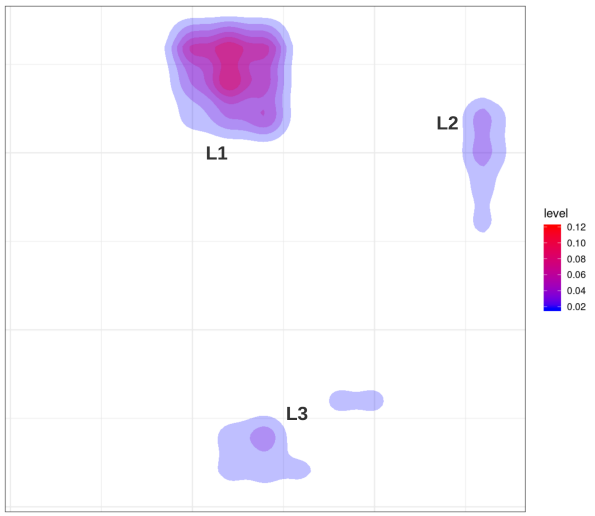

Figure 6: Heatmap of device location in our testbed. The "level" corresponds to the relative number of devices per square foot.

sharing the same locations. Average dBm values remain the same across network scans, which reveal that WDPKR could be used to confirm a device remains in the same location. Further analysis is needed to improve the confidence of the location results.

WDPKR's active decryption module was tested on a smaller home network of three devices and a single router. Using the network owner's known WPA2-PSK, WDPKR performed a deauthentication attack and disconnected all three devices from the home Wifi network. This decryption attack gives further insight into each device and its vulnerabilities. This motivates the addition of more decryption attacks into WDPKR's toolset, which could contribute security characteristics to a device profile.

4 RELATED WORK

Device selection. Past work that incorporated a large number of real-world IoT devices and experimented with them on-hand achieved better results [2, 13]. However, many lack strength in dataset diversity through their selection of devices - focusing on individual families of devices such as Samsung Smart Things, Google or Amazon limited their experiments. Works that diversified their devices and device activities by choosing low and high-power devices, varying brand names, and functionality types achieved more interesting and applicable results [12].

Feature selection. Prior work has demonstrated the ability to discover and investigate devices using a subset of packet features, RF data, or RSSI data [8, 11, 13]. Features selected from packet headers or payload information have been used to identify devices and study their behavior. RF data has been used as the physical

Table 1: Comparison of prior work.

	Packet	RF	RSSI	Dataset	Discovery	Traffic Decryption	Fingerprinting	Location Mapping	Profiling
Stojkoska [11]			X					X	
Jafari [8]		X					X		
DEFT [13]	X						X		
HoMonit [14]	X								X
IoT Inspector [7]	X			X	X		X		
WDPKR (this work)	X	X	X	X	X	X	X	X	X


```
"34:75:63:cc:0c:50": {
    "Vendor": "Unknown",
    "Device_type": "Client",
    "Frame_sent": 5266,
    "Frame_received": 8551,
    "Distance": "0.514 meters",
    "Avg_DBM": "-34.313330801367265",
    "Prob_list": [
        "98:da:c4:77:51:23"
    ]
},
```


Figure 7: Analysis module results of a Nooie Camera 360.

layer alternative for identifying devices on a network [8]. RSSI data has been used for locating devices in a home and detecting anomalies [11]. Device classification studies have employed port-based, payload-based, statistical-based, and behavioral-based methods to discover patterns across packets and identify devices [12]. Much like in device selection, prior works lack robust feature selections. These prior works have each focused on a different subsection of device features. We have not seen work that studies a holistic view of all features that can be extracted from a single device to help identify it.

Fingerprinting. Fingerprinting methods vary across prior work, but many focus on improving classifier accuracy and the ability to detect new devices and fingerprint them quickly. Thangavelu created a dynamic and scalable solution to this with a distributed fingerprinting technique that relied on packet features [13]. Their fingerprinting methodology aims to classify devices into device types, whereas WDPKR approaches fingerprinting to analyze a device's behavior. It generates a fingerprint around the device's behavior before grouping it into a type or class of device. Prior work has used various classification algorithms on traffic, including Random Forest, SVM, KNN, and even deep learning methods for fingerprinting [8]. In [7], crowd-sourced traffic and ARP spoofing techniques are used to discover and identify devices. The work focuses on labeling the data, generating statistics on the collected traffic, and assessing the security and privacy trends they discover across the devices. The developed tool could expand by using their expansive dataset to train fingerprinting and profiling methods.

Profiling. Prior work has used fingerprinting methods to generate profiles for device behavior. In [2] each profile is based on discrete fingerprints where each fingerprint is constructed from data points from session packets. Zhang [14] uses automata to illustrate stateful devices and generate profiles for each device type. In [5], the authors used static analysis to develop a profile of IoT devices by analyzing source code. This profiling method is limited in its inability to work dynamically against software updates and user re-configurations of devices. In general, a stateful profile of a device needs to be adaptable to the changing characteristics of IoT devices in real-time in order to effectively identify devices and detect anomalies.

Table 1 compares WDPKR to prior work in terms of feature sets and the WDPKR modules. Columns 1-3 illustrate the split of research across packet-based, RF-based, and RSSI-based methods to understand IoT devices. Column 4 indicates work that has built a representative dataset through experimentation. Columns 5-9 demonstrate prior work has motivated each WDPKR module by demonstrating their value to understanding IoT device security and privacy individually. WDPKR is able to leverage the strength of the modules together to provide a stronger profile for devices. A strong profiling methodology will open WDPKR to anomaly detection as well.

5 CONCLUSION

We present WDPKR, a proposed set of techniques for performing dynamic profiling of smart home IoT devices. By building on discovery and fingerprinting techniques, we believe that WDPKR can achieve reliable, accurate profiles for these devices. We have developed a small prototype, which shows that our ideas have promise. This project, however, is very much a work in progress, and we have much to do. In particular, we wish to slowly build out our WDPKR prototype so that it can perform more of the functionality outlined in Section 2. We also wish to build out our experimental setup to cover other IoT protocols, such as Zigbee, Z-Wave, and Bluetooth. Additionally, we anticipate that we will collect a sizeable amount of quality data in building WDPKR; we will contribute the (labeled) IoT datasets generated in the course of this project back to the community. We hope that, with more development effort and experimentation, we can build WDPKR to effectively provide dynamic profiling for smart home networks using performant scientific and statistical methods.

REFERENCES

[1] Pranshu Bajpai, Aditya K Sood, and Richard J Enbody. 2018. The art of mapping IoT devices in networks. *Network Security* 2018, 4 (2018), 8–15.

[2] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. 2018. Behavioral fingerprinting of iot devices. In *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security.*

41–50.
[3] Philippe Biondi. 2007. Network packet manipulation with Scapy.
[4] Stefan Brands and David Chaum. 1993. Distance-bounding protocols. In *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 344–359.
[5] Z Berkay Celik, Patrick McDaniel, and Gang Tan. 2018. Soteria: Automated iot safety and security analysis. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. 147–158.
[6] Yuanfeng Du, Dongkai Yang, and Chundi Xiu. 2015. A novel method for constructing a WiFi positioning system with efficient manpower. *Sensors* 15, 4 (2015), 8358–8381.
[7] Danny Yuxing Huang, Noah Apthorpe, Frank Li, Gunes Acar, and Nick Feamster. 2020. Iot inspector: Crowdsourcing labeled network traffic from smart home devices at scale. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 2 (2020), 1–21.
[8] Hossein Jafari, Oluwaseyi Omotere, Damilola Adesina, Hsiang-Huang Wu, and Lijun Qian. 2018. IoT Devices Fingerprinting Using Deep Learning. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*. IEEE, 1–9.
[9] Jesse Kornblum. 2006. Fuzzy hashing. *The Digital Forensic Research Worksho (DFRWS)* (2006).
[10] Kasper Bonne Rasmussen and Srdjan Capkun. 2010. Realization of RF Distance Bounding.. In *USENIX Security Symposium*. 389–402.
[11] Biljana Risteska Stojkoska, Ivana Nižetić Kosović, and Tomislav Jagušt. 2017. How much can we trust RSSI for the IoT indoor location-based services?. In *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, 1–6.
[12] Hamid Tahaei, Firdaus Afifi, Adeleh Asemi, Faiz Zaki, and Nor Badrul Anuar. 2020. The rise of traffic classification in IoT networks: A survey. *Journal of Network and Computer Applications* 154 (2020), 102538.
[13] Vijayanand Thangavelu, Dinil Mon Divakaran, Rishi Sairam, Suman Sankar Bhunia, and Mohan Gurusamy. 2018. Deft: A distributed iot fingerprinting technique. *IEEE Internet of Things Journal* 6, 1 (2018), 940–952.
[14] Wei Zhang, Yan Meng, Yugeng Liu, Xiaokuan Zhang, Yinqian Zhang, and Haojin Zhu. 2018. Homonit: Monitoring smart home apps from encrypted traffic. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1074–1088.