Stony Brook University

Detecting Smart Home Activity through Network Traffic Signatures

A Thesis presented

by

Sayali Anil Alatkar

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Master of Science

in

Computer Science

Stony Brook University

May 2020

Stony Brook University

The Graduate School

Sayali Anil Alatkar

We, the thesis committee for the above candidate for the
Master of Science degree, hereby recommend
acceptance of this thesis

Samir R. Das
Professor and Chair, Department of Computer Science

Amir Rahmati
Assistant Professor, Department of Computer Science

Michalis Polychronakis
Associate Professor, Department of Computer Science

Vasudevan Nagendra
Head of Security, Plume Design Inc.

This thesis is accepted by the Graduate School

Eric Wertheimer
Dean of the Graduate School

Abstract of the Thesis

Detecting Smart Home Activity through Network Traffic Signatures

by

Sayali Anil Alatkar

Master of Science

in

Computer Science

Stony Brook University

2020

The popularity of the Internet of Things (IoT), smart home devices specifically, has been tremendous in the past couple of years. Devices like smart speakers, smart fridge, etc. have integrated seamlessly with the lives of homeowners. The main reasons being their accessibility, cost-effectiveness, and energy-efficiency. These devices by nature, continuously communicate with each other and their servers, to automate the day-to-day activities for their owners and in the process generate huge amounts of network traffic. But at the same time, this underlying automation can often occlude the operation and communication between devices from their owners.

This thesis is a step towards understanding the causation of smart-home activities, thereby increasing the visibility and transparency in a smart home. In this thesis, we use the network traffic traces to generate packet-level traffic signatures for device activities (change speaker volume, play music, turn ON lights), which can help identify the triggering devices for an observed activity in a smart home, without causing significant computational overhead or storage constraints. By signature matching, we show how the triggering device can be identified by the users. We present our results on our IoT testbed (WINGS Lab in Stony Brook University) and publicly available datasets. We demonstrate that our approach can identify activities and corresponding sources with good accuracy.

Contents

Acknowledgments

I would like to express my deep sense of gratitude to Prof. Samir Das for giving me this opportunity. I am highly indebted to him for helping me improve and present this work by giving useful suggestions. I am very grateful to Vasudevan Nagendra for introducing to me the concepts of Internet of Things, Security and Privacy, and, Causal Inference. I would like to thank him for his constant support, motivation and valuable insights that helped me complete this thesis. I am also grateful to have got the chance to work with other students, like Shubham, Nikita, Omik and Astitva.

Sayali Anil Alatkar
May 2020
Stony Brook University

Chapter 1

Introduction

Smart home devices have been widely adopted by homeowners. Business Insider reported up to 8 billion IoT devices in 2019 and has projected up to 41 billion IoT devices by 2027 [21]. The advent of IoT has made the lives of users not only comfortable but also efficient. Users can now control electronic devices remotely through smart-apps. To accomplish their goals, smart devices continuously communicate with respective cloud servers and/or other smart devices. Despite the ease of operating these devices, the smart home ecosystem offers little knowledge of the cause or source of an event taking place. For example, if a smart lock was disabled during the day, it could correspond to a normal activity performed by the user, an attack of malicious intent, or device failure. By realizing the true cause, the user or network administrator can better handle situations.

In this thesis, we take a step towards detecting the cause of an event within a smart home by effectively analyzing the network-level traffic traces to derive signatures out of them. The basic principle of such causal analysis is to find the root cause for why something happens. Applying this mindset to IoT infrastructures means identifying the source of triggers or actions we observe in the network. The complexity of the problem increases with the addition of new devices and functionalities. Studies [20] predict that the number of IoT devices will exceed 25 billion by 2021, and in 2022 a single "smart home" is likely to have over five hundred connected devices, while a large scale "smart campus" or "smart city" might have hundreds of thousands or even millions of devices attached to its IoT infrastructure including street lamps, weather sensors, and traffic signals [18],[6].

The multitude of smart devices is accompanied by inherent vulnerabili-

ties that persist and can be attributed to the resource-constrained nature of the devices. In such a situation, compute-intensive security solutions may not be feasible. Hence, it is important to address these vulnerabilities by understanding their root-cause. We focus on a lightweight solution that can be extended to all smart devices. Our approach uses the extensive network traffic produced by these devices to generate packet-level signatures for device-specific activities. Each smart device activity is represented by a novel signature, that uniquely captures its traffic pattern. Further, by signature matching, we can identify the activity as well as the device that led to the activity. We expect that our study: 1) provides visibility into the smart home system; 2) detects any abnormal activities performed by the device; 3) detects any device failures. Below we provide a brief introduction to the IoT infrastructure.

1.1 IoT Architecture

The IoT infrastructure consists of four stages. First of all, it consists of the devices (sensors and actuators) that are connected to the internet and send the information they sense from the environment to IoT gateways. The next stage consisting of the IoT data acquisition systems and gateways collect this huge unprocessed data, convert it into digital streams, filter and pre-process it so that it is ready for analysis. This is followed by edge devices that further process and analyze the data, where they also apply visualization and machine learning algorithms. Finally, the data is sent to data centers which could be cloud servers or locally installed. The cloud servers perform powerful analytics on the data and provide decisions that help humans interact, control, and monitor the devices. The cloud servers communicate back with the devices to inform of the decisions.

1.2 Motivation

A large number of smart devices have flooded the market, each one providing new and better functionalities than the previous one. It doesn't come as a surprise that a majority of these devices have seamlessly integrated with our day-to-day lives. Despite the ease of operation, a smart home ecosystem is not as transparent. The inherent automation obstructs our understanding of

how these devices operate or communicate.

With new devices, we have a new set of vulnerabilities that are exploited by attackers to not only disrupt the functionality but also the safety of its users. One of the most recent attacks involving the hijack of the Ring home security system [19] shows how much a user can be compromised. Such security concerns have led to several studies that focus on exposing existing vulnerabilities and patching them.

Most recent works have focused on improving the security of IoT devices by analyzing the network traffic and identifying malicious activities [9], [1]. In this thesis, we take a fundamental approach where we characterize the behaviors (responses to different triggers) of each smart device by creating activity-specific packet-level signatures. Such that each signature is associated with an activity. By being able to identify device-activity, we can determine the triggering device source. Root cause identification can be used to alert the user or network administrator to detect unexpected behaviors.

Therefore, we aim to increase transparency within the smart home ecosystem by exposing the associated triggers of an action performed by smart devices.

1.3 Organization of this Thesis

- Chapter 2 discusses related work in the field of IoT traffic analysis and device activity inference.
- Chapter 3 discusses briefly our work on IoT policies, their conflict detection, and resolution; followed by presenting our problem formulation and a case study for traffic analysis.
- Chapter 4 presents our methodology.
- Chapter 5 provides details on our testbed and data collection setup.
- Chapter 6 presents our evaluations on both public and our datasets.

Chapter 2

Related Work

A recent work by Trimananda et al. [22] identifies the traffic activity of smart devices by generating packet-level signatures by extracting request-reply packet pairs in WiFi and Zigbee devices with a 97% recall. But their approach is only limited to devices that communicate over TCP and several IoT devices communicate over UDP (no request-response packets), which limits their application. Our approach focuses on traffic volume characteristics, and thus can be extended to different devices and IoT protocols.

Early works have focussed on Nest Thermostat and wired Nest Protect [5], where they are able to infer the thermostat's transition modes, i.e., *Home* and *Auto Away* by analyzing the network traffic. On the other hand, HomeMatic HAS [8] focuses on Zigbee/Z-Wave devices and compares the SmartApps' activities inferred from the encrypted network traffic with their expected behaviors prescribed in their source code or UI interfaces.

Sivanathan et al. [17], [15], [14] proposed clustering to identify IoT devices using features from the network traffic generated by these devices in different environments (smart campuses and smart cities). But their approach is unable to distinguish between the device-activities. Specifically, their work [14] reports an accuracy of up to 94% for IoT device identification, while, [16] proposes a lightweight solution for device identification through TCP ports, but this again does not extend to device activity inference.

Apthorpe at al. [2], [4], [3] have shown how passive monitoring of the network traffic generated by smart devices can be used to user activities in a smart home. However, these works cannot identify device activities. Other works either use an ML-based approach that builds a Random Forest classifier [9] to identify application behaviors like downloading the firmware,

receiving a configuration change, and sending video to a remote user to learn the traffic behavior of device activities or use a Hidden Markov Model [1] to infer user activities by identifying the device actions with up to 90% accuracy.

Signature extraction has been used extensively for malware/anomaly detection. Perdisci et al. [10] use structural similarities among malicious HTTP traffic traces generated by executing HTTP-based malware to build signatures while Zhang et al. [24] detect deviations in SmartApps by building a DFA for each SmartApp and comparing the observed behavior to inferred behavior with the expected DFA models.

Other works that analyze network traffic behavior use traffic activity graphs (TAGs) [13] which capture the interactions among hosts engaging in certain types of communications and their collective behavior.

Chapter 3

Problem Formulation

We begin this section by briefly discussing about our work on IoT policy conflict detection and resolution. Then, we present the problem statement for this thesis. Later, we discuss about the key insights we obtained by manually analyzing the network traffic from a smart speaker - Google home mini. The traffic characteristics observed in Google home mini led us to the approach. In this thesis, we use the words *activity* and *interaction* interchangeably, and both refer to the functionalities/capabilities of smart devices (for example, turn on/off, change brightness/color, move in front of movement sensor, change temperature, etc).

3.1 IoT Policy Conflict Detection and Resolution

The goal of this work was to develop an IoT policy conflict detection and resolution engine for IoT policies in a smart environment (smart homes, smart campuses, smart cities). To do so, we translated the IoT policies to Z3-based SMT programs [11] which can automatically detect the violations between the policies and resolve them using a precedence mechanism. IoT policies are automation rules determined by a network administrator, to control device behavior according to specific requirements from smart devices within a network. For example, the policy: "For Nest Cameras in building 1 allow Web traffic only between 9 AM and 6 PM", controls the action of all Nest Cameras in building 1.

Policies are designed keeping in mind the security, privacy, and usage

requirements of users. Often, policies conflict with each other thereby leaving the devices in a confused state. Usually, for a campus as big as Stony Brook University, there could be at least 10,000 policies. Manually identifying the conflicting ones can be a daunting task for any network administrator. Therefore, to automate such a situation, we built an engine that can identify conflicts and resolve them using precedence based on policy context (security policies, privacy policies, user-specific policies, etc.).

This work is divided into 2 parts: Conflict detection and Conflict resolution.

3.1.1 Conflict Detection

For this part, we used an SMT solver called Z3 [23]. Z3 is a high-performance theorem prover developed by Microsoft Research with several applications like software/hardware verification and testing, constraint solving, analysis of hybrid systems, security, biology (in silicon analysis), and geometrical problems. We used Z3 as a constraint solver to identify overlapping/conflicting policies. We converted each IoT policy into a constraint that was given as input to the solver. Z3 constraints contain Binary or Integer values concatenated with operators (<, <=, >, >=, == and !=) for comparison. Since IoT policies are defined as Strings, it is important to transform them into Z3 specific constraints first. If the solver flagged the constraints as overlapping, then the corresponding policies are conflicting. We marked all such policies as "conflicting" and proceeded towards their resolution.

3.1.2 Conflict Resolution

For policy conflict resolution, we used a precedence mechanism. This means that if two policies are conflicting, the one with higher precedence overrules the other. The precedence of a policy is determined on the basis of its context. Therefore, higher precedence means a higher severity of the policy context. For example, a policy: "Allow traffic from all Nest Cameras in building 1 after 5 PM" has a *security* context, whereas another policy: "Disable traffic from all Nest Cameras on Floor 5 in Building 1 after 4 PM" was designed by a user to prevent unnecessary traffic since floor 5 is shut down after 4 PM. Both these policies are at conflict for the Nest devices on Floor 5 after 4 PM. Also, both have different contexts, i.e., security and user-specific. To resolve conflicts among them, we gave precedence based on context severity,

such that until 5 PM, the second policy is in effect, but after 5 PM the first policy comes into effect disabling the second policy.

3.2 Problem Statement

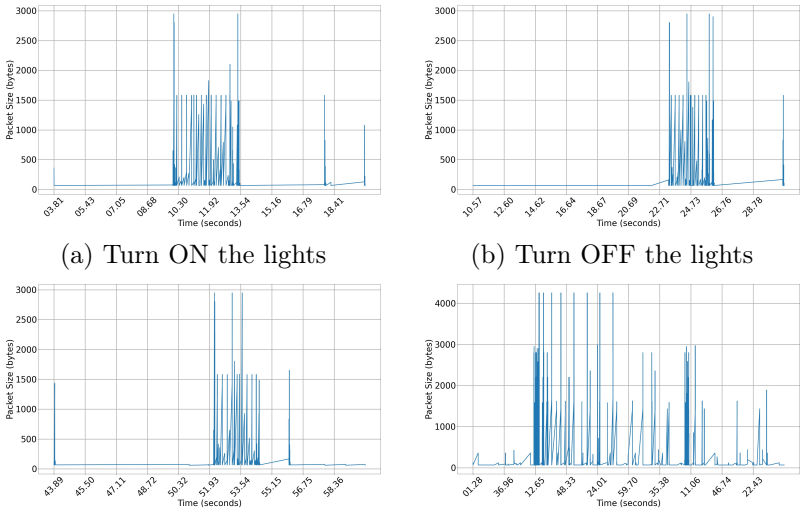
The goal of this thesis is to provide transparency in the trigger-action behavior of IoT devices in a smart home ecosystem. As a step towards this goal, we focus on device activity detection using minimal information obtained from the network traffic traces.

Despite their encrypted communication [**1**], passive traffic monitoring can reveal enough information about the IoT devices and their activities [2], [3], [4]. We focus on *attributes* like packet magnitude, inter-packet arrival times, DNS servers, and communication protocols that can be easily captured through passive monitoring. We found that these attributes can be used to infer the activity performed by the smart devices(lights ON/OFF/dim/change color), and consequently used them to generate packet-level traffic signatures specific to each device activity. Through these signatures, we want to identify the device activity and infer the triggering device. This inference provides visibility into the smart home ecosystem, thereby allowing the users and network administrators to identify the source of activity (motion detected/smart app).

3.3 Case Study: Traffic Analysis of Google Home Mini

We illustrate through the example of a smart speaker: Google home mini, the fundamental insights obtained by manually inspecting its traffic generated over WiFi. Figure 3.1 shows the burstiness of traffic generated as a response to four commands: 1) Lights ON 2) Lights OFF, 3) Tell the current time, and 4) Play music.

Traffic generated by IoT devices can be categorized as autonomous and user-generated. Autonomous traffic includes network configuration (DNS, NTP) [15] as well as routine communication between devices and back-end servers (keep-alive messages). Autonomous traffic repeats consistently and is limited to constant traffic size. Whereas, user-generated traffic is asynchronous and has a higher magnitude and frequency. Also, different IoT



(a) Turn ON the lights

(b) Turn OFF the lights

(c) Tell the current time

(d) Play music

Figure 3.1: Traffic exhibited by a google home mini for 4 different commands. The burstiness and packet sizes for all commands is distinct.

devices communicate with unique servers through diverse protocols.

By analyzing the traffic produced through the time command, we show that the following four features can uniquely represent the activity of a device.

3.3.1 Magnitude

We observed a significant increase in traffic magnitude (packet size). Unlike regular traffic which is limited to 500 bytes, the traffic for this activity went up to and above 2,500 bytes (Figure 3.2).

3.3.2 Inter-Packet Arrival Time

Figure 3.3 shows how the inter-packet arrival time (IPA) suddenly decreases for the duration of the interaction. Compared to an average communication time of 1.5 seconds, the IPA went as low as 0.08 seconds.

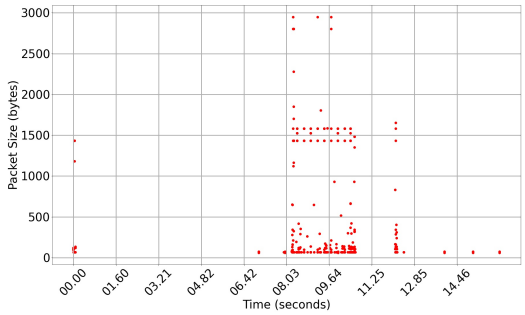


Figure 3.2: High magnitude traffic generated for the time command.

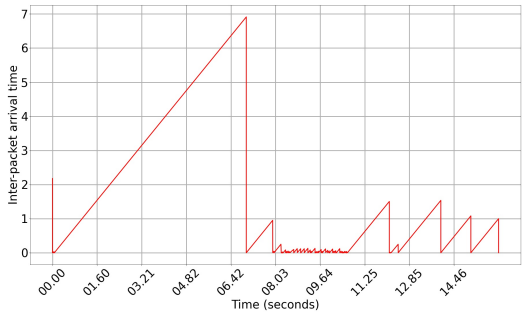


Figure 3.3: The reduced inter-packet arrival time for the time command.

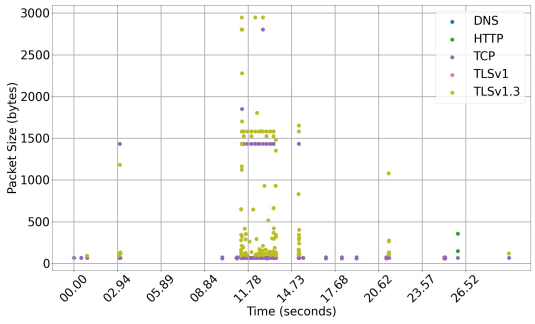
3.3.3 DNS Servers

The google home mini is continuously communicating with its cloud servers and other devices. But we observed that it communicates with the same server each time for a given activity. Over two separate instances, it talked to the servers: *google.com* and *time.google.com*. We use this as another feature to identify traffic specific to a device activity.

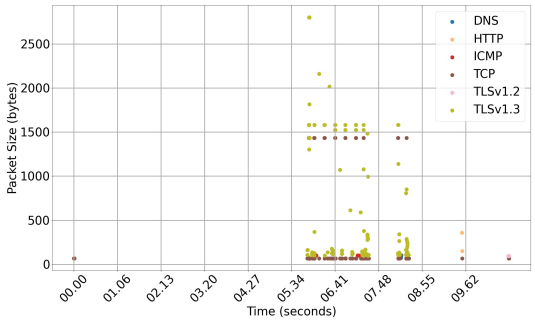
We also observed that for command "play music", the device talked to *youtube.com* and *googlevideo.com*.

3.3.4 Network Protocols

Figure 3.4 demonstrates the protocols used by the smart device in two different instances. The protocols: DNS, HTTP, TCP, TLSv1.3 are used during both instances. Thus, we use the protocols used by the device during an activity to uniquely represent the activity.



(a)



(b)

Figure 3.4: Network protocols used by google home mini during interaction (current time) at two separate instances.

Chapter 4

Setup and Dataset Details

In this section, we present our IoT testbed and data collection steps.

4.1 IoT Testbed

We set up an IoT testbed with a few commercially available IoT devices, that operate on WiFi, to collect traffic traces. Figure 4.1 shows our setup. The WAN interface of the ethernet hub is connected to the public Internet via the university network. We developed our approach by focusing on one test-case which, we show through results, can be extended to different scenarios. We tried to simulate a smart-lights system which can be controlled either through voice commands given to Google home mini or through an app, within the same network.

4.2 Data Collection

We captured the traffic for while continuously interacting with the devices. Before beginning with the experiments, we waited for at least two minutes for all the devices to power-on. To capture the traffic, we used the command-line tool, tcpdump, on the NUC flashed with Ubuntu 16.04 placed between the access point and the gateway. *Tcpdump* passive monitors, collects, and stores the traffic as PCAP files on the disk.

Once the devices were ready, we began the packet capture and started communicating with them. During our experiments, we actively interacted with the devices and recorded the type and timestamp for each interaction.

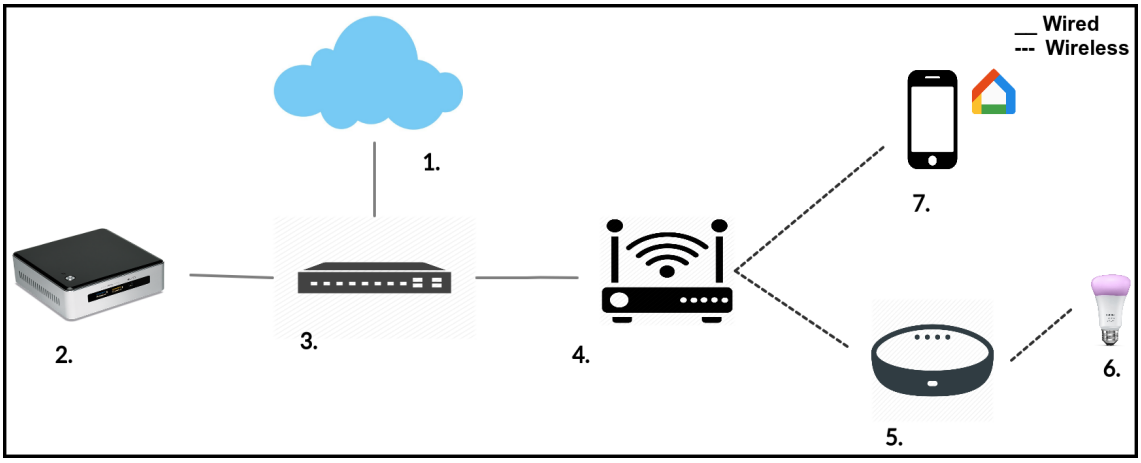


Figure 4.1: WINGS Lab IoT Testbed [1-Internet(cloud backend), 2-NUC, 3-Ethernet Hub, 4-Wireless AP, 5-Google Home Mini, 6-Philips Hue, 7-Smartphone(google home app)]

After each interaction, we waited for 5 more seconds stopped the packet capture. We conducted the same experiments 4 times for each activity to collect enough traffic samples.

The types of interactions include 1) voice command, where we triggered the Google mini using a voice command, to communicate with the smart lights and turn them on/off; 2) cloud communication, by using a companion app on the smartphone, in the same network, that communicates with the device through its cloud end-point.

We interacted with the Google mini it through three commands 1) Lights ON, 2) Lights OFF, and, 3) Tell current time.

Therefore, our collected data includes PCAP files of network traces accompanied by a text file containing device names, timestamps, and interaction-types.

Chapter 5

Methodology

In this section, we explain our signature generation and detection techniques.

5.1 Preprocessing

Before signature generation or detection, we preprocess the PCAP files by dividing them into intervals.

5.1.1 Trace Feature Extraction

Given a PCAP file, we split it into *equal* time intervals of 5 seconds each i.e. {[0-5),[5-10),[10-15),..}. For each interval, we extract a set of features, such that each interval is represented by an $8D$ data point with following features: 1) *Maximum packet size*, 2) *Mean packet size*, 3) *Standard deviation in packet size*, 4) *total protocols*, 5) *total packets*, 6) *mean inter-packet arrival time*, 7) *DNS Servers*, 8) *Unique Protocols*. Therefore, each interval is a new data point with 8 dimensions.

Additionally, for signature generation, we add two more features: 1)*Interval start-time*, 2)*Interval end-time*.

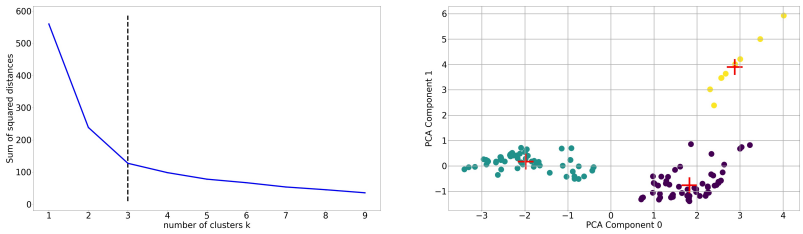
5.2 Signature Generation

5.2.1 Attributes Clustering

We use the 8D data points for signature generation. Before further processing, we standardize our data samples using z-scores. These data points, although specific to an activity, can contain the additional periodic IoT traffic. To separate the data points most relevant to the activity, we use an unsupervised learning algorithm: K-Means, to cluster and determine them. Using multidimensional clustering over the first 6 attributes, we single-out the cluster that correctly represents the activity traffic. We used the elbow method heuristic to determine the optimal value for k.

Every device activity requires a separate K-Means clustering.

Figures 5.1 and 5.2 show the clusters obtained for a smart speaker - Google home mini for two different commands (volume change and voice command). Since it is difficult to visualize more than three attributes, we used Principal Component Analysis (PCA) to project the data points to a two-dimensional space.

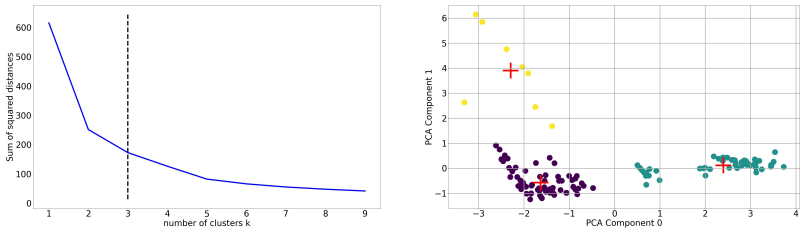


(a) The Elbow Method showing the optimal k

(b) K-Means Clustering

Figure 5.1: Traffic clustering for volume change command.

Among the k clusters, we selected the one with the largest data points. We found that the data points in such a cluster also have higher packet size values and lower IPA values. Such a cluster best described the nature of traffic spike observed when a smart device has been interacted with. Therefore, for each activity, we have an associated cluster that we use to generate signatures.



(a) The Elbow Method showing the optimal k

(b) K-Means Clustering

Figure 5.2: Traffic clustering for voice command.

5.2.2 Generation

For a device D let A_1,A_2,A_3 be the activities it performs. Let A_i be the i^{th} activity for the device where $A_i = [d_i^1, d_i^2, d_i^3, ...]$, and d^j is a data point(5-second interval) from the largest cluster identified for A_i. Also, as mentioned earlier, we have ten features for each data point (see Table 5.1).

t_s	Interval start-time
t_e	Interval end-time
pkt_{max}	Maximum packet size
pkt_{mean}	Mean packet size
pkt_{std}	Standard deviation packet size
pkt_{total}	Total packets
$proto_{total}$	Total unique protocols
$proto_{unique}$	Unique protocols
dns_{unique}	Unique dns servers
ipa_{mean}	Mean inter-packet arrival time

Table 5.1: Features used for traffic signature creation

An entire communication between the smart device and its cloud server or other devices may not always be captured within a single interval, with part of it occurring before or after the interval's start or end times. To make sure that we are not losing important traffic details about the protocols and DNS servers, we expand every data point by considering traffic 1 second before the start time and 1 second after the end-time of the data sample. For convenience, we represent a data point simply as d_i. Therefore, for the data point d_i, we expand its interval by looking into additional traffic between $(t_s^i - 1, t_e^i - 1)$ and $(t_s^i + 1, t_e^i + 1)$ and append any additional protocols and

DNS servers to $proto_{unique}$ and dns_{unique} respectively. We do this for all the points (intervals) in the cluster.

For device D, given the associated PCAP traces D_{traces} for activity A_i, and, n data points for i_{th} activity i.e., $d = [d^1, d^2, d^3, ..., d^n]$, we calculate the signature for this activity using algorithm 1.

Algorithm 1 Generate activity-specific traffic signatures

procedure ADD-DATA-POINTS(D_{traces}, d)
  d_new = []
  for $i \leftarrow 1$ to n **do**
    protocols $\leftarrow \emptyset$
    dnsServers $\leftarrow \emptyset$
    dCurr $\leftarrow \emptyset$
    Iterate over packets 1 second before the data-point interval
    $x_0 \leftarrow t_s^i - 1$
    $x_1 \leftarrow t_e^i - 1$
    $j \leftarrow 0$
    while $D_{traces}^j[time] \geq x_0$ & $D_{traces}^j[time] \leq x_1$ **do**
      protocols $\leftarrow$ protocols $\cup$ $D_{traces}^j[protocols]$
      dnsServers $\leftarrow$ dnsServers $\cup$ $D_{traces}^j[dns]$
      $j \leftarrow j + 1$
    Iterate over packets 1 second after the data-point interval
    $x_0 \leftarrow t_s^i + 1$
    $x_1 \leftarrow t_e^i + 1$
    $j \leftarrow 0$
    while $D_{traces}^j[time] \geq x_0$ & $D_{traces}^j[time] \leq x_1$ **do**
      protocols $\leftarrow$ protocols $\cup$ $D_{traces}^j[protocols]$
      dnsServers $\leftarrow$ dnsServers $\cup$ $D_{traces}^j[dns]$
      $j \leftarrow j + 1$
    dCurr[maxPktSize,meanPktSize] = [pkt_{max}^i, pkt_{mean}^i]
    dCurr[stdPktSize, totalPkt, meanIpa] = [pkt_{std}^i, pkt_{total}^i, ipa_{mean}^i]
    dCurr[proto] = $proto_{unique}$ $\cup$ protocols
    dCurr[dnsServers] = dns_{unique} $\cup$ dnsServers
    d_new[i] = dCurr
  signature $\leftarrow$ GENERATE-SIGNATURES(d_new)
  return signature

From the new dataset d_{new}, we define the signature by properties like maximum packet size, mean packet size, std packet, mean inter-packet arrival time by taking the mean of each attribute. Along with the mean value, we also store the standard deviation of each attribute. This standard is used during signature detection stage.

We take the intersection of the unique protocols and dns servers from all the datapoints and add them to the signature.

```
30: procedure GENERATE-SIGNATURES(d_new)
31:     maxPkt ← 0
32:     meanPkt ← 0        stdPkt ← 0
33:     totalPkt ← 0       meanIpa ← 0
34:     protocols ← ∅      dnsServers ← ∅
35:     for i ← 1 to n do
36:         maxPkt ← maxPkt + d_new[i][maxPktSize]
37:         meanPkt ← meanPkt + d_new[i][meanPktSize]
38:         stdPkt ← stdPkt + d_new[i][stdPktSize]
39:         totalPkt ← totalPkt + d_new[i][totalPkt]
40:         meanIpa ← meanIpa + d_new[i][meanIpa]
41:         protocols ← protocols ∩ d_new[i][protocols]
42:         dnsServers ← dnsServers ∩ d_new[i][dnsServers]
43:     signature ← ∅
44:     signature[maxPkt, meanPkt] ← [overline(maxPkt), overline(meanPkt)]
45:     signature[stdPkt, totalPkt, meanIpa] ← [overline(stdPkt), overline(totalPkt), overline(totalPkt)]
46:     signature[protocols] ← protocols
47:     signature[dnsServers] ← dnsServers
48:     return signature
```

Signature File: Each signature is created and stored as a dictionary. Collectively, a signature file of a device stores all the relevant signature dictionaries of a device. During signature detection, we retrieve the stored signatures and match them with the newly observed network traffic traces.

5.3 Signature Detection

Given a set of traffic traces from a smart home, we want to be able to identify the underlying activity. To do this, we perform signature detection over this traffic.

For signature detection, we first preprocess the PCAP network traces by dividing them into 5-second intervals, such that each interval is a new data point with 8 dimensions.

Then, we match these data points against the already-generated signatures to detect the device activity. As explained above, our signature contains 7 features. The first 5 features are quantitative, where we use a *relaxed matching* strategy. This is because devices do not always exhibit the exact traffic

patterns in terms of packet lengths or inter-packet arrival times, and there are always slight variations. In *relaxed matching*, we check if the quantitative attributes of the new data points are within a single standard deviation of the corresponding attribute value of the signature. Specifically, we check if the observed traffic has value between $(attr - delta)$ and $(attr + delta)$.

The remaining two attributes are categorical, where we implemented an *exact matching* strategy. The two attributes, protocols and DNS servers, are sets consisting of relevant values. In *exact matching*, we make sure that values for these attributes within the data points match exactly with signature.

Chapter 6

Evaluations

In this section, we provide our evaluations on two datasets: our own IoT testbed in the WINGS Lab at Stony Brook University, and, on a publicly dataset - the Mon(IoT)r dataset [12]. We divide our evaluations into two categories, where we first present the unique signatures on our dataset, followed by the detection accuracy on the Mon(IoT)r dataset.

We have conducted all our evaluations on a 64-bit Dell Insipron-5559, memory 15.6 GiB and Intel® Core™ i7-6500U CPU @ 2.50GHz 4 with Ubuntu 16.04 LTS.

6.1 WINGS Lab Testbed

6.1.1 Unique Signatures

In order to perform evaluations on our testbed, we collected upto 4 traffic traces for each command that we interacted with the Google Home Mini. We used the setup in Chapter 6 to capture the traces, where we manually interacted with the device through 3 commands: 1) Turn ON lights, 2) Turn OFF lights, 3) Tell current time. Tables 6.1 and 6.2 show the unique signatures and computational characteristics for the commands.

6.2 Public Dataset: Mon(IoT)r dataset

We apply our technique to a state-of-the-art publicly available dataset, the Mon(IoT)r dataset [12], which contains traffic traces for 55 devices. It is

Signature Attributes	ON	OFF	Time
pkt_{max}	2592.0	1986.88	2000.16
pkt_{mean}	451.28	311.66	370.8
pkt_{std}	623.23	461.10	356.46
pkt_{total}	204	46	117.16
ipa_{mean}	0.0427	0.1485	0.1170
$proto_{unique}$	DNS,TCP,TLSv1,TLSv1.3	TCP,TLSv1.3	TCP,TLSv1.3
dns_{unique}	-	-	-

Table 6.1: Signatures extracted for 3 commands given to Google Home Mini

	ON	OFF	Time
Computation Time (seconds)	10.7	9.4	9.9
Size (bytes)	275	235	258

Table 6.2: Computation time and size for 3 commands given to Google Home Mini

a labeled dataset, containing PCAP files for each device, divided according to event/interactions conducted. The interaction experiments conducted by the authors include 1) Local-Device, which consists of physically interacting with the device, or using voice commands (without using a voice assistant from a separate device); 2) Phone-Device, by using a companion app on a phone connected to the same network as the IoT device, thus allowing direct communication between the phone and the IoT device; 3) Alex-Device, by using voice commands to trigger the Echo Spot's Alexa voice assistant, which subsequently interacted with the device according to the voice command; 4) Cloud-Device, by using a companion app on a phone connected to a different network than the IoT device, to force the IoT device to use cloud infrastructure to communicate. We tested our approach on only a subset of the 55 devices due to a limitation of the dataset and our approach. The dataset does not contain enough samples for Local-Device interactions for certain devices, and thus cannot produce reliable signatures using our approach.

We show two types of evaluations: *matching* and *detection*.

In *matching*, we demonstrate the ability of our technique to extract signatures and affirm to them, through matching accuracy, by dividing the traffic traces into train and test samples.

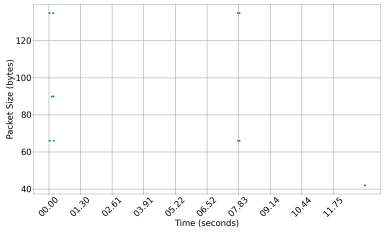
In *detection*, we show the precision and recall scores of our technique on mixed traffic (combining traces of multiple devices) to inform of its perfor-

mance in a real-time scenario.

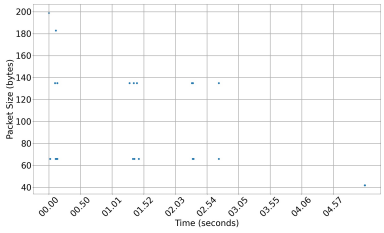
6.2.1 Matching

To generate signatures for each device activity, we merged all the PCAP files and extracted 5-seconds from them as described in Chapter 4. After extracting the intervals, we divided the dataset into training (60%) and testing(40%) sets. We used the training set to generate signatures and performed signature detection on the testing set. Tables 6.3, 6.4, 6.5, 6.6 summarize the results and accuracy.

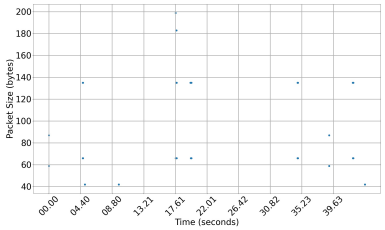
We observed that for device events with matching accuracy less than 100%, the network traffic was more scattered, and so their clusters could not accurately capture the expected details. Figures 6.1 and 6.2 show the difference in traffic generated by two devices from Table 6.3: Blink Security Hub and Lefun Cam.



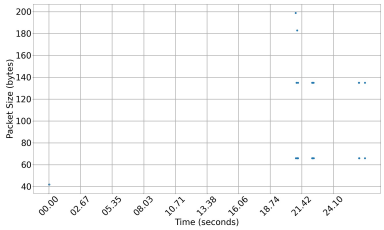
(a) Event:Stop; Interaction type:Alexa-Device



(b) Event:Watch; Interaction type:Alexa-Device

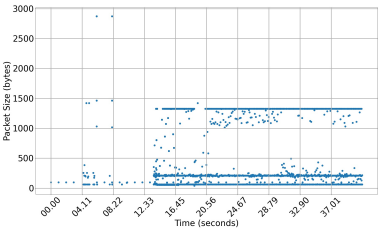


(c) Event:Watch; Interaction type:Phone-Device

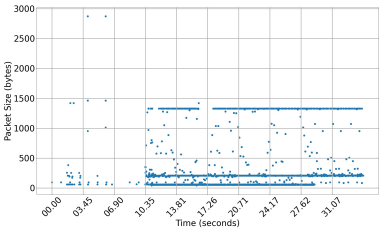


(d) Event:Photo; Interaction type:Phone-Device

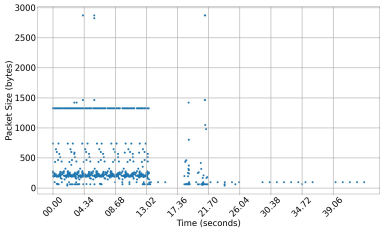
Figure 6.1: Traffic distribution for Blink Security Hub



(a) Event:Recording; Interaction type:Phone-Device



(b) Event:Watch; Interaction type:Phone-Device



(c) Event:Photo; Interaction type:Phone-Device

Figure 6.2: Traffic distribution for Lefun Cam

6.2.2 Detection

To demonstrate detection accuracy, we merged the PCAP files of different devices in two different combinations. First, we chose all the devices with the capability to directly(smart lights) or indirectly(smart hubs) turn lights on (see Table 6.7). We did this to show if our technique can identify the right source within potential sources. Second, we selected 5 commonly used devices: Amazon Echo Spot, Roku TV, Xiaomi Strip, Luohe Spy Cam, and Smartthings Hub and arbitrary interaction events for each (see Table 6.8).

Device	Event	Interaction type	Matching Accuracy
Flux Blub	ON	Alexa-Device	100%
	OFF	Alexa-Device	87.5%
	Dim	Alexa-Device	87.5%
	Color	Alexa-Device	100%
Amcrest Camera	Photo	Phone-Device	100%
	Recording	Phone-Device	100%
	Watch	Phone-Device	100%
Fire TV	Change Menu	Phone-Device	87.5%
Samsung Fridge	View inside	Local Device	100%
	Voice command	Local Device	68.75%
	Volume	Local Device	100%
	Set	Phone-Device	100%
	view inside	Phone-Device	100%
Xiaomi Hub	OFF	Phone-Device	75%
	ON	Phone-Device	75%
Blink Security Hub	Stop	Alexa-Device	75%
	Watch	Alexa-Device	50%
	Photo	Phone-Device	56.25%
	Watch	Phone-Device	68.75%
Amazon Cloudcam	Stop	Alexa-Device	56.25%
	Watch	Alexa-Device	50%
	Watch	Phone-Device	50%
Amazon Echo Spot	Audio OFF	Phone-Device	100%
	Audio ON	Phone-Device	100%
	Voice	Local Device	81.25%
	Volume	Local Device	62.5%
Insteon Hub	OFF	Alexa-Device	81.25%
	ON	Alexa-Device	100%
	OFF	Phone-Device	100%
	ON	Phone-Device	85.7%
Invoke with Cortana	Volume	Local Device	62.5%
	Voice	Local Device	75%
Lefun Cam	Photo	Phone-Device	100%
	Recording	Phone-Device	100%
	Watch	Phone-Device	100%
LG TV	Change Menu	Phone-Device	100%
TP-Link Plug	OFF	Alexa-Device	100%
	ON	Alexa-Device	93.75%
	OFF	Phone-Device	100%
	ON	Phone-Device	93.75%
Ring Doorbell	Stop	Alexa-Device	25%
	Watch	Alexa-Device	68.75%
	Watch	Phone-Device	81.25%

Table 6.3: Signature matching accuracy: Trained and tested on the Mon(IoT)r dataset

Device	Event	Interaction type	Matching Accuracy
Blink Cam	Stop	Alexa-Device	42.85%
	Watch	Alexa-Device	64.28%
	Photo	Phone-Device	87.5%
	Watch	Alexa-Device	100%
Smarter iKettle	ON	Alexa-Device	31.25%
	OFF	Alexa-Device	37.5%
	Set	Alexa-Device	75%
	Color	Phone-Device	62.5%
	Dim	Phone-Device	93.75%
Luohe Cam	Photo	Phone-Device	100%
	Recording	Phone-Device	100%
	Watch	Phone-Device	87.5%
Philips Bulb	ON	Phone-Device	62.5%
	OFF	Phone-Device	62.5%
	Color	Phone-Device	68.75%
	Dim	Phone-Device	75%
Lightify	ON	Alexa-Device	56.25%
	OFF	Alexa-Device	84.6%
	Color	Alexa-Device	62.5%
	Dim	Alexa-Device	75%
	ON	Phone-Device	87.5%
	OFF	Phone-Device	87.5%
	Color	Phone-Device	93.75%
	Dim	Phone-Device	62.5%
Xiaomi Strip	ON	Alexa-Device	68.75%
	OFF	Alexa-Device	50%
	Color	Alexa-Device	75%
	Dim	Alexa-Device	93.75%
	ON	Phone-Device	93.75%
	OFF	Phone-Device	0%
	Color	Phone-Device	100%
	Dim	Phone-Device	25%
Wansview Cam	Photo	Phone-Device	100%
	Recording	Phone-Device	100%
	Watch	Phone-Device	100%
Yi Cam	Photo	Phone-Device	100%
	Recording	Phone-Device	100%
	Watch	Phone-Device	100%
ZModo Doorbell	Photo	Phone-Device	100%
	Recording	Phone-Device	100%
	Watch	Phone-Device	100%

Table 6.4: Signature matching accuracy: Trained and tested on the Mon(IoT)r dataset

Device	Event	Interaction type	Matching Accuracy
Amazon Echo Plus	ON	Alexa-Device	25%
	OFF	Alexa-Device	75%
	Color	Alexa-Device	37.5%
	Dim	Alexa-Device	62.5%
	ON	Phone-Device	100%
	OFF	Phone-Device	0%
	Color	Phone-Device	100%
	Dim	Phone-Device	87.5%
	Audio ON	Phone-Device	50%
	Audio OFF	Phone-Device	100%
	Voice command	Local Device	18.75%
	Volume	Local Device	56.25%
Sengled Hub	ON	Alexa-Device	50%
	OFF	Alexa-Device	68.75%
	Color	Alexa-Device	0%
	Dim	Alexa-Device	25%
	ON	Phone-Device	56.25%
	OFF	Phone-Device	81.25%
	Color	Phone-Device	100%
	Dim	Phone-Device	68.75%
Smartthings Hub	ON	Alexa-Device	67.85%
	OFF	Alexa-Device	78.57%
	Color	Alexa-Device	75%
	Dim	Alexa-Device	67.85%
	Lock	Alexa-Device	75%
	Unlock	Alexa-Device	87.5%
	ON	Phone-Device	35.71%
	OFF	Phone-Device	85.71%
	Color	Phone-Device	85.71%
	Dim	Phone-Device	100%
	Lock	Phone-Device	100%
	Unlock	Phone-Device	92.85%
TP-Link Bulb	ON	Alexa-Device	50%
	OFF	Alexa-Device	50%
	Color	Alexa-Device	0%
	Dim	Alexa-Device	75%
	ON	Phone-Device	75%
	OFF	Phone-Device	37.5%
	Color	Phone-Device	100%
	Dim	Phone-Device	100%

Table 6.5: Signature matching accuracy: Trained and tested on the Mon(IoT)r dataset

Device	Event	Interaction type	Matching Accuracy
Roku TV	Remote	Phone-Device	100%
Samsung TV	Browse Menu	Phone-Device	92.85%
Magichome Strip	ON	Alexa-Device	93.75%
	OFF	Alexa-Device	100%
	Color	Alexa-Device	100%
	Dim	Alexa-Device	93.75%
	ON	Phone-Device	75%
	OFF	Phone-Device	100%
	Color	Phone-Device	100%
	Dim	Phone-Device	100%
Microseven Cam	Watch	Phone-Device	87.5%
Nest T-stat	Set	Alexa-Device	50%
	ON	Phone-Device	68.5%
	OFF	Phone-Device	75%
	Set	Phone-Device	50%
Philips Hue	ON	Alexa-Device	6.25%
	OFF	Alexa-Device	43.75%
	Color	Alexa-Device	18.75%
	Dim	Alexa-Device	56.25%
	ON	Phone-Device	100%
	OFF	Phone-Device	100%
	Color	Phone-Device	92.85%
	Dim	Phone-Device	92.3%
TP-Link	ON	Alexa-Device	93.75%
	OFF	Alexa-Device	100%
	ON	Phone-Device	93.75%
	OFF	Phone-Device	50%
WeMo Plug	ON	Alexa-Device	80%
	OFF	Alexa-Device	78.5%
	ON	Phone-Device	100%
	OFF	Phone-Device	50%
Wink 2	ON	Alexa-Device	85%
	OFF	Alexa-Device	78.5%
	ON	Phone-Device	100%
	OFF	Phone-Device	50%

Table 6.6: Signature matching accuracy: Trained and tested on the Mon(IoT)r dataset

Device	**Matching Accuracy**	**Precision**	**Recall**
Insteon Hub	100%	100%	100%
Lightify	56.25%	100%	41.1%
Sengled	37.5%	100%	36.5%
Smartthings Hub	67.8%	89.7%	67.8%
Wink 2	43.75%	100%	48.9%
Magichome Strip	93.75%	100%	92.5%
TP-Link Bulb	56.25%	100%	63.2%

Table 6.7: Signature detection accuracy: devices triggered by voice command "Lights ON" given to Alexa.

Device	**Event**	**Interaction type**	**Matching Accuracy**	**Precision**	**Recall**
Smartthings Hub	Unlock	Alexa-Device	85.7%	100%	87.5%
Luohe Spy Cam	Watch	Phone-Device	87.5%	100%	36.2%
Xiaomi Strip	ON	Phone-Device	87.5%	100%	86.17%
Roku TV	Remote	Phone-Device	100%	67.5%	100%
Amazon Echo Spot	Audio ON	Phone-Device	100%	46.25%	58.11%

Table 6.8: Signature detection accuracy: Smart Home simulated.

Chapter 7

Conclusion

Through this work, we have taken a step towards causal inference of IoT device activity in a smart home by creating activity-specific traffic signatures. We identified key characteristics of the IoT traffic that distinguish between routine (firmware updates, NTP queries) and interactive activities of the devices. Using these characteristics, we define a unique signature for each activity of a device. By comparing the signatures to the observed traffic, we are able to identify the device-activity and thus causality of the observed event.

We created our own dataset for the technique, and our method only requires prior preprocessing of the raw traces. In fact, the signatures can be created and stored beforehand. We showed that their computation and storage does not cause significant overhead. We also showed that our approach works with good accuracy on a public dataset. Through this work, we aim to provide visibility into the smart home ecosystem which can further be used to detect any system failures or abnormal activities in the smart house by identifying the cause of observed events.

Limitations: We discuss the limitations of our technique.

First, as the devices undergo firmware updates, the traffic pattern could change with it. As a result, these signatures would be required to calculate again. Second, for devices with little or scattered network traffic, the matching accuracy for our technique is not very high. We want to improve this in our future scope. Third, traffic shaping techniques [2] can make it harder to identify the activity as unexpected traffic is added to the network.

7.1 Future Work

As a future scope, we want to make our signature more exclusive and robust by incorporating more sources of information like network traffic from other IoT protocols (Zigbee, Z-Wave, BLE), device power consumption information, EM emanation records and IoT app automation-rules that are unique to devices. By incorporating more information and features, the signatures can be made more robust. Also, we want to test our approach extensively on datasets like the UNSW IoT traces [7].

Bibliography

[1] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and A. Selcuk Uluagac. "Peek-a-Boo: I see your smart home activities, even encrypted!" In: *ArXiv* abs/1808.02741 (2018).

[2] Noah Apthorpe, Danny Yuxing Huang, Dillon Reisman, Arvind Narayanan, and Nick Feamster. "Keeping the Smart Home Private with Smart(er) IoT Traffic Shaping". In: *Proceedings on Privacy Enhancing Technologies* 2019 (2019), pp. 128 –148.

[3] Noah Apthorpe, Dillon Reisman, and Nick Feamster. "A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic". In: *ArXiv* abs/1705.06805 (2017).

[4] Noah Apthorpe, Dillon Reisman, Srikanth Sundaresan, Arvind Narayanan, and Nick Feamster. "Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic". In: *ArXiv* abs/1708.05044 (2017).

[5] Bogdan Copos, Karl N. Levitt, Matt Bishop, and Jeff Rowe. "Is Anybody Home? Inferring Activity From Smart Home Network Traffic". In: *2016 IEEE Security and Privacy Workshops (SPW)* (2016), pp. 245–251.

[6] *Internet2 Smart Campus and Cities*. May 2017.

[7] *IOT TRAFFIC TRACES*. `https://iotanalytics.unsw.edu.au/iottraces`.

[8] Frederik Möllers, Sebastian Seitz, Andreas Hellmann, and Christoph Sorge. "Short paper: extrapolation and prediction of user behaviour from wireless home automation communication". In: July 2014. DOI: `10.1145/2627393.2627407`.

[9] TJ OConnor, Reham Mohamed, Markus Miettinen, William Enck, Bradley Reaves, and Ahmad-Reza Sadeghi. "HomeSnitch: Behavior Transparency and Control for Smart Home IoT Devices". In: *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*. Miami, Florida, 2019, 128–138. DOI: 10.1145/3317549.3323409.

[10] Roberto Perdisci, Wenke Lee, and Nick Feamster. "Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces". In: *NSDI*. 2010.

[11] *Programming Z3*. https://theory.stanford.edu/~nikolaj/programmingz3.html.

[12] Jingjing Ren, Daniel J. Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. "Information Exposure for Consumer IoT Devices: A Multidimensional, Network-Informed Measurement Approach". In: *Proc. of the Internet Measurement Conference (IMC)*. 2019.

[13] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization". In: *ICISSP*. 2018.

[14] A. Sivanathan, H. H. Gharakheili, and V. Sivaraman. "Inferring IoT Device Types from Network Behavior Using Unsupervised Clustering". In: *2019 IEEE 44th Conference on Local Computer Networks (LCN)*. 2019, pp. 230–233.

[15] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics". In: *IEEE Transactions on Mobile Computing* 18 (2019), pp. 1745–1759.

[16] Arunan Sivanathan, Hassan Habibi Gharakheili, and Vijay Sivaraman. "Can We Classify an IoT Device using TCP Port Scan?" In: Dec. 2018, pp. 1–4. DOI: 10.1109/ICIAFS.2018.8913346.

[17] Arunan Sivanathan, Daniel Sherratt, Hassan Habibi Gharakheili, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. "Characterizing and classifying IoT traffic in smart cities and

campuses". In: *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2017), pp. 559–564.

[18] *Smart Cities will be consuming 1.2 billion IoT devices per year by 2025.* May 2016.

[19] *Somebody's Watching: Hackers Breach Ring Home Security Cameras.* `https://www.nytimes.com/2019/12/15/us/Hacked-ring-home-security-cameras.html`.

[20] *The Future Smart Home: 500 Smart Objects Will Enable New Business Opportunities.* `http://www.gartner.com/document/2793317`.

[21] *THE INTERNET OF THINGS 2020: Here's what over 400 IoT decision-makers say about the future of enterprise connectivity and how IoT companies can use it to grow revenue.* `https://www.businessinsider.com/internet-of-things-report`.

[22] Rahmadi Trimananda, Janus Varmarken, Athina Markopoulou, and Brian Demsky. *PingPong: Packet-Level Signatures for Smart Home Device Events.* 2019. arXiv: `1907.11797 [cs.NI]`.

[23] *Z3 API in Python.* `https://ericpony.github.io/z3py-tutorial/guide-examples.htm`.

[24] Wei Zhang, Yan Meng, Yugeng Liu, Xiaokuan Zhang, Yinqian Zhang, and Haojin Zhu. "HoMonit: Monitoring Smart Home Apps from Encrypted Traffic". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (2018).