HomeSnitch: Behavior Transparency and Control for Smart Home IoT Devices

TJ OConnor
North Carolina State University
tjoconno@ncsu.edu

Reham Mohamed
Technische Universität Darmstadt
reham.mohamed@trust.tu-darmstadt.de

Markus Miettinen
Technische Universität Darmstadt
markus.miettinen@trust.tu-darmstadt.de

William Enck
North Carolina State University
whenck@ncsu.edu

Bradley Reaves
North Carolina State University
bgreaves@ncsu.edu

Ahmad-Reza Sadeghi
Technische Universität Darmstadt
ahmad.sadeghi@trust.tu-darmstadt.de

ABSTRACT

The widespread adoption of smart home IoT devices has led to a broad and heterogeneous market with flawed security designs and privacy concerns. While the quality of IoT device software is unlikely to be fixed soon, there is great potential for a network-based solution that helps protect and inform consumers. Unfortunately, the encrypted and proprietary protocols used by devices limit the value of traditional network-based monitoring techniques. In this paper, we present HomeSnitch, a building block for enhancing smart home transparency and control by classifying IoT device communication by semantic *behavior* (e.g., heartbeat, firmware check, motion detection). HomeSnitch ignores payload content (which is often encrypted) and instead identifies behaviors using features of connection-oriented application data unit exchanges, which represent application-layer dialog between clients and servers. We evaluate HomeSnitch against an independent labeled corpus of IoT device network flows and correctly detect over 99% of behaviors. We further deployed HomeSnitch in a home environment and empirically evaluated its ability to correctly classify known behaviors as well as discover new behaviors. Through these efforts, we demonstrate the utility of network-level services to classify behaviors of and enforce control on smart home devices.

CCS CONCEPTS

• **Security and privacy → Access control**; • **Networks → Programmable networks**; • **Computing methodologies → Bagging**.

ACM Reference Format:
TJ OConnor, Reham Mohamed, Markus Miettinen, William Enck, Bradley Reaves, and Ahmad-Reza Sadeghi. 2019. HomeSnitch: Behavior Transparency and Control for Smart Home IoT Devices. In *12th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '19), May 15–17, 2019, Miami, FL, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3317549.3323409

1 INTRODUCTION

The pervasiveness of the Internet-Of-Things (IoT) devices is estimated to reach 50 billion by 2020 [31]. A substantial population of these IoT devices is emerging in the residential environment. These smart home devices offer convenience by monitoring wireless sensors such as temperature, carbon monoxide, and security cameras. Further, these devices enable automating actions including sounding alarms, making coffee, or controlling lighting. As the functionality of these devices mature, they are increasingly interacting autonomously, invisible to the end-user. Turning off a connected alarm can signal a coffee pot to begin brewing and a motion-sensor can automate turning on a light. However, risk is introduced with each newly connected device due to their often-insecure software.

The smart home market is incredibly broad and heterogeneous, lacking sufficient standards and protocols to enforce security and privacy. Yet, these same devices generate, process, and exchange vast amounts of privacy-sensitive and safety-critical information in our homes [30]. Manufacturers can surreptitiously gather data from the end user in violation of their expressed policy desires. As an example, Germany banned the wireless *Cayla* doll, which contained a microphone and uploaded recorded conversations with children [19, 20]. Further, an attacker may use a smart home IoT device to gain unauthorized access to other devices in the home. The Central Intelligence Agency's Weeping Angel implant on connected Samsung Televisions targeted home WiFi networks, accessed user-names and passwords, and covertly recorded audio [12].

Smart home security is an emerging topic without a clearly defined threat model. A compromised or misbehaving smart home device may attack other network hosts, but it may also eavesdrop on the physical home environment. It is often not necessary to stop the first, or even second, instance of covert audio recording. Rather, it is important to identify misbehaving devices and take some action (e.g., access control, or simply turning off the device). For example, in the case of the Cayla doll, there is limited harm in eavesdropping on a few of a child's conversations, but risk increases the longer the eavesdropping occurs. Similarly, when a device suddenly changes its behavior, as in the case of Weeping Angel recording audio, the owner's goal is to prevent prolonged exposure.

The software on IoT devices is unlikely to improve significantly in the near future. Therefore, consumers would greatly benefit from a network-based solution. However, contemporary tools provide limited help in interpreting communications exchanges taking place in the smart home network, as they require in-depth understanding about network protocols and the specific network set-up. Meaningful monitoring and access control are challenging for experts, let alone for regular smart home users. The task is further complicated by the fact that IoT devices often use encryption, meaning solutions cannot rely on the contents of network traffic.

In this paper, we present HomeSnitch as a building block for improving the transparency and control over the network communications of smart home devices. The key insight of HomeSnitch is to classify and control *semantic behaviors* using features that represent application-layer dialogue between clients and servers. We implement HomeSnitch on a commodity wireless router and build upon Software Defined Networking (SDN) primitives to control network traffic based on end-user preferences. By operating on application behaviors, rather than IP addresses and packet flows, HomeSnitch facilitates enhanced user agents to provide transparency and control over smart home devices. We evaluate HomeSnitch in two ways. First, we evaluate its classification accuracy using a dataset of labeled packet traces from the YourThings [3] project. For this dataset, our analysis demonstrates that HomeSnitch achieves an accuracy of 99.69% in classifying behaviors. Second, we deployed HomeSnitch in a real home environment with 20 devices and evaluated its ability to correctly classify known behaviors, as well as to help discover new behaviors. Such capabilities demonstrate the practical utility and importance of providing end-users with greater transparency and control of smart home devices.

We make the following contributions in this paper.

- *We present the HomeSnitch framework for identifying distinct semantic behaviors by smart home devices.* HomeSnitch provides a building block for transparency and control of smart home IoT device network communications by reporting activity on the granularity of semantic device behaviors instead of low-level concepts such as distinct packet flows.
- *We propose a supervised learning technique to classify semantic device behaviors using network traffic.* The technique works on both encrypted traffic and proprietary application-layer protocols and achieves greater than 99% accuracy on an independent data-set. More importantly, HomeSnitch can identify novel behaviors with resistance to under-fitting.
- *We perform an empirical evaluation of HomeSnitch in a real home environment.* We demonstrate how HomeSnitch can help identify previously unseen devices and behaviors. We demonstrate our behavior fingerprinting approach enables policy and access control across fine-grained behaviors.

The remainder of this paper proceeds as follows. Section 2 provides motivation and articulates the problem. Section 3 overviews the HomeSnitch architecture. Section 4 describes its design. Section 5 describes the implementation. Section 6 evaluates accuracy and performance. Section 7 presents an empirical study of HomeSnitch in a home area network. Section 8 discusses limitations. Section 9 overviews related work. Section 10 concludes.

2 MOTIVATION AND PROBLEM

IoT smart home devices present significant security and privacy challenges for consumers. Many smart home devices have poor authentication mechanisms, un-patchable software vulnerabilities, and limited access control configurability. Even when devices are not being exploited, they may present privacy threats due to unexpected behavior implemented by device manufacturers. Providing transparency into device behavior is challenging, as smart home devices frequently use proprietary application-layer protocols. Finally, efforts to secure smart home devices (e.g., HTTPS encryption) further complicate network layer access controls.

Motivating Example (Alexa API Abuse): Checkmarx security researchers recently identified a method for abusing the implicit policies of the Amazon Alexa digital assistant [21]. In order to covertly record audio from Alexa enabled devices, the researchers created a voice activated application, known as an Alexa *skill*. The application posed as a simple application for solving math problems but instead recorded audio indefinitely. The researchers abused the API by proving a null value to a prompt, silencing the device and deceiving the user into believing it is no longer listening. This abuse of trust motivates our work and presents a challenging problem for defense. In terms of the smart home IoT device, it must assume the server-side platform is trusted after mutual authentication and execute commands. HomeSnitch addresses this by pushing the monitoring and policy enforcement into the network using SDN and a machine learning classification of application behaviors.

Problem Statement: The goal of HomeSnitch is to provide the primitives for transparency and control of behaviors of otherwise resource constrained smart home IoT devices. To achieve this, HomeSnitch seeks to operate on application *behaviors* such as heartbeat, firmware check, reporting, and uploading audio or video recordings. Abstracting network flows and packet data into classifications offers better intuition about the device activity. Thus, HomeSnitch seeks to enable transparency by reporting *which* device performs *what* behaviors, *when* and *how often*, rather than the simple existence and frequency of network traffic. Furthermore, HomeSnitch seeks to enable enhanced systems that enforce network access control policies based on observations of previous undesirable behaviors, thereby allowing end users to use valuable features of smart home devices that otherwise exhibit privacy invasive behaviors.

Threat Model and Assumptions: HomeSnitch is designed for a residential smart home network setup consisting of sensors, actuators, and smart objects, both wired and wirelessly connected to a consumer-grade router. We assume smart home devices contain default credentials, lack sufficient security protocols, enable over-privilege, and contain un-patched vulnerabilities. We assume an adversary can compromise smart home devices either by directly connecting to them either through *NAT hole punching* [33], lateral pivoting from other compromised devices, gaining physical control of the device [9], or abusing cloud-based control. HomeSnitch protects against both malicious adversaries and benign behavior by manufacturers that violate the explicit desires of the end-user.

The adversary's goal is to exploit smart home devices to cause physical effects (e.g., turn off an alarm), ex-filtrate data (e.g., enable remote video surveillance), steal credentials, or compromise other devices in the network. The attacker may launch an attack through malware residing on the device, a remote network attack, or abuse server-side privileges on the specific service-provider for the device. Further, we assume smart home devices lack timely patching and updates to correct vulnerabilities. Defending attacks without the cooperation of the device is a sufficiently hard problem since

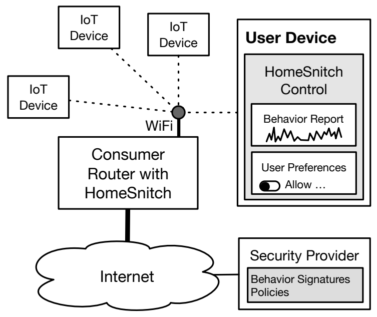


Figure 1: HomeSnitch provides a building block for transparency and control of IoT device behaviors.

traditional security approaches involve scanning for vulnerabilities, enabling fine-grained permissions on the host, and patching vulnerabilities. A smart home network of devices cannot employ any of these approaches in the defense. HomeSnitch prevents against attacks via IP connection-oriented protocols. We consider devices that communicate via other protocols (e.g. ZigBee, Bluetooth, NFC) as important and orthogonal problems for future work. As discussed in Section 8, our work does not address the case of mimicry attacks.

3 OVERVIEW

As motivated in the previous section, HomeSnitch seeks to enable practical transparency and control of the network communication of IoT smart home devices. Figure 1 depicts our vision of how HomeSnitch would operate in practice. The HomeSnitch software is originally configured with a set of behavior models and default network access control policies. Over time, these models and policies are updated via a feed from a security service provider (e.g., BitDefender [7] or Norton [23]). When IoT smart home devices communicate with servers on the Internet, or with one another on the local LAN, HomeSnitch classifies the network flows as known behaviors for devices. Unknown network flows are matched with a set of the closest device behaviors and optionally reported to the security service provider to enhance training models. End users gain transparency through the HomeSnitch control interface (e.g., a web page or mobile app). This interface reports the network activity of IoT smart home devices based on their high-level behaviors. The interface also allows the end user to specify preferences of what behaviors they would like to allow (e.g., similar to how smartphone platforms enable greater privacy control).

This paper addresses two key research challenges.

- *Behavior Classification:* There is a large semantic gap between the bytes sent in packets and the behaviors that are understandable to users. Classification is further inhibited by encryption and proprietary application-layer protocols.
- *Network Mediation:* Network traffic must be mediated both between devices and the Internet, as well as between devices within the home network. Smart home devices are designed for simple residential networks and functionality often fails when enhanced technologies such as VLANs are introduced. Therefore, the LAN must retain a flat address space.

HomeSnitch overcomes these research challenges through the use of SDN and supervised machine learning. More specifically, it extends the firmware of an operating system for commodity routers (OpenWrt) with OpenFlow capabilities. By using OpenFlow, HomeSnitch is able to mediate all network communication, both between devices and between devices and the internet, without changing the IP address space structure. Using OpenFlow also allows the design to easily scale to environments with multiple access points, which are becoming more common as newer WiFi technologies such as IEEE 802.11ac operate faster over shorter distances. HomeSnitch then uses OpenFlow to send network flow information to an SDN controller application. In our prototype, the SDN controller application runs on a separate Raspberry Pi controller board connected via Ethernet; however, as consumer devices with greater processing capabilities (e.g., the Turris Omnia [36]) become available, all of the HomeSnitch functionality can be moved to a single device.

The SDN controller application performs behavior classification based on network flow information. Specifically, HomeSnitch extracts application data unit (ADU) exchanges to acquire an abstract representation of the client and server communication. ADU features are then used by a supervised machine learning algorithm to classify the flow as a known behavior triple: (<vendor>, <model>, <activity>), e.g., (Ring, Doorbell, Motion-Upload). Note that human readable names such as "Motion-Upload" must be assigned to a learned behavior. When a network flow has a poor match to any known behavior triple (as defined by a configurable threshold), HomeSnitch provides a set of closest matches. These matches can provide valuable insight into the correct classification (e.g., all closest matches are for the same vendor or the same activity). In practice, we imagine a security service provider that maintains and distributes a trained model of these triples.

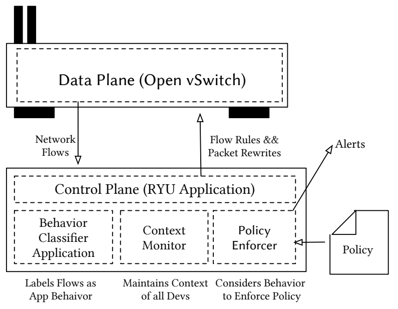

Data Plane (Open vSwitch)
Network Flows
Flow Rules && Packet Rewrites
Alerts
Control Plane (RYU Application)
Behavior Classifier Application
Context Monitor
Policy Enforcer
Policy
Labels Flows as App Behaivor
Maintains Context of all Devs
Considers Behavior to Enforce Policy

Figure 2: HomeSnitch leverages a fine-grained behavior classification approach to provide context for effective policy enforcement and access control.

4 DESIGN

This section describes the design of HomeSnitch, as depicted in Figure 2. Our design currently separates the classification logic in the controller application onto an external embedded device; however, as routers continue to become more computationally powerful, it is reasonable to expect the controller to be moved onto the router. We begin the discussion of the HomeSnitch design by describing the behavior classification from network flow information. This classification system represents the core of our technical contribution. We then proceed to describe the context manager, the policy language, and policy enforcement using SDN primitives.

Note that for simplicity, we assume all devices on the network are IoT devices. However, in practice networks will contain desktops, laptops, and smartphones. We assume the existence of techniques to determine if a device is an IoT device, which can be accomplished by looking at the OUI or using more advanced classification systems such as IoT Sentinal [18].

4.1 Classification of Application Behaviors

We treat smart home device behavior identification as a multi-class classification problem. More specifically, we seek to map a network flow to the specific application behavior and device that produced it. Examples of device behavior include downloading firmware, receiving a configuration change, and sending video to a remote user. Deep packet inspection, protocol analysis, and certificate inspection provide limited insight into behaviors, as network flows are often encrypted to a cloud-based server such as Google Cloud, Amazon Web Services, or Microsoft Azure.

HomeSnitch defines an application behavior as a triple. We originally sought to define a behavior simply by an activity that transcends vendors and devices. However, in practice, we found vendor and model implementations defined the structure of application data exchanges. This even varies across specific product offerings as we discovered the original Ring Doorbell had a much more simplistic set of behaviors compared against the recent Ring Pro model. Therefore, we decided to tie the activity to a specific vendor and device. We found that this has the benefit of helping identify unknown behaviors. For example, a new behavior may have a relatively close match for another device by the same vendor. The remainder of this section describes our feature selection, classification model, and operational considerations.

Application Data Unit (ADU) Extraction: To overcome the limitation of analyzing encrypted traffic, HomeSnitch constructs application data unit (ADU) exchanges that provide an abstract representation model of the exchange between a client and a server. Several approaches exist to build application-level models from packet headers in an offline manner [14, 26, 27]. HomeSnitch uses *adudump* [35], which has the capability for continuous measurement of application-level data from network flows. Additionally, *adudump* only uses TCP/IP packet header traces (i.e., no application layer headers or payloads) to construct a structural model of the application dialog between each server and the client. This overcomes the limitations of increasingly encrypted payloads in smart home devices. As an added benefit, this header information is readily available to our SDN controller applications. In our analysis of several different devices and behaviors, we found that the application data units can be used to accurately distinguish between behaviors.

Feature Selection: HomeSnitch uses supervised machine learning to classify network flows into application behaviors. We selected thirteen features from transport layer headers that describe the client/server dialogues of smart home IoT devices (listed in Table 1). Each feature is a descriptive statistic calculated from a single application data unit. ADUs are bidirectional representations of sequential exchanges within a single connection, terminated upon proper connection tear-down or by flow timeout [35]. A key insight that benefits HomeSnitch is that smart home IoT traffic is predominately sequential with the server and client taking turns.

Table 1: HomeSnitch features describe the application data exchanges. This approach overcomes the limitations of solutions that rely on physical, network, or transport layer headers unique to smart home deployments.

Feature	Category	Importance
average bytes from IoT device per sequence	throughput	0.213104
average bytes from server per sequence	throughput	0.072519
aggregate sever bytes sent over ADU	throughput	0.105775
aggregate IoT device bytes sent over ADU	throughput	0.117552
min bytes from IoT device from a sequence	burstiness	0.038917
min bytes from server from a sequence	burstiness	0.038344
max bytes from server from a sequence	burstiness	0.079063
max bytes from IoT device from a sequence	burstiness	0.135909
stdev of bytes between server sequences	burstiness	0.054491
stdev of bytes between IoT device sequences	burstiness	0.050798
server sequences per ADU	synchronicity	0.013566
IoT device sequences per ADU	synchronicity	0.016211
total time of connection	duration	0.063750

This property is due to the limited complexity of devices with most processing done server side. The lack of traffic concurrency allows HomeSnitch to observe the behavior of the application by extracting statistics from each sequential turn.

Our model excludes IP addresses, port numbers, and DNS information as they present several limitations and are increasingly less reliable features [8, 16, 38]. Instead, we focus on content agnostic features that solely describe the behavior of a smart home application. We recognize a real-world appliance may benefit from additional information (i.e., destination address and port). However, our threat model considers an attacker that is focused on exhibiting a behavior and could use flow information to mask and hide a behavior. We conducted several experiments with real devices to identify features that offer better discriminative capabilities of smart home applications. Specifically, we observed the importance of features describing the burstiness, synchronicity, and throughput of application dialogues. Table 1 lists the set of these features, consisting of average, aggregate, minimum and maximum measurements for these dialogues. For purposes of definition, *sequence* indicates a single directional application exchange and *ADU* indicates the entire application data unit dialogue, bounded by the creation and termination of the entire network flow. The notion of burstiness describes the proximity of arrival instances within each other and the variance between each arrival [4]. To measure this within an ADU, we examine the variance in terms of both payload size and inter-arrival times. To measure synchronicity, we observed measurements that describe how a client and server take turns sending data within the context of an ADU. Throughput is a routinely used but important measure of the aggregate data sent over the duration of the ADU.

As part of our experiments in Section 6, we calculated the feature importance (a measurement of the predictive importance for each feature variable in the random forest) with respect to the YourThings [3] corpus of data (also shown in Table 1). Average bytes, max bytes, and aggregate bytes from the IoT device were the most dominant features and accounted for 46.65% of the decision estimate used by our classifier. Since smart home devices are typically designed for specific purposes and communicate in a limited fashion, these features best describe the throughput and burstiness unique to application data units. This insight matches the analysis of the YourThings data-set, where 64.72% of device flows lasted for less than a second. Overall, we carefully selected a balanced set of features to support the multi-class classification problem associated with device behaviors. We discovered burstiness features describe the on-demand actions of motion sensors. In contrast, throughput features better describe data-intensive sensors. Synchronicity and duration offer strong features for data logging and analytic reports.

Classification Model Selection: We considered multiple algorithms (described in Section 6) but found Random Forest to be the best for intuitive reasons. Random Forest leverages both ensemble and averaging methods to build several estimators independently and then average their predictions [29]. By leveraging a bagging-based ensemble model, HomeSnitch is resistant to under-fitting with a reduced variance. Our results in Section 6.1 demonstrate that Random Forest offers a very accurate model that precisely classifies smart home device behaviors and detects new behaviors.

In Section 6, we show that both Random Forest Classifiers (RFC), Gradient Boost, and K-Nearest Neighbor (kNN) algorithms score well in accuracy using k-fold cross validation of a labeled data-set. However, HomeSnitch needs to inform the user when previously unseen behaviors occur. This design constraint further motivates an ensemble or averaging based classifier, as they are more resistant to under-fitting. To detect unknown behaviors, HomeSnitch uses the probability prediction score as a measure to define a threshold of acceptance. Data points below the threshold are predicted as unknown behaviors and a new classification label is created.

We define *Unknown Behavior Miss Rate* (UBMR) as the percentage of data-points in a given data-set that fail to be identified as a new classification. True Positive Rate (TPR) is the total number of true positives divided by the sum of the true positives plus false positives. Ideally, the classification should minimize the UBMR without adverse effect to the TPR. As discussed and empirically evaluated in Section 6, we found a threshold of 0.89% as the best balance between UBMR and TPR. The evaluation further demonstrates the utility of Random Forest over kNN.

Training Data: Supervised machine learning approaches require initial training data to determine classes for unclassified data-points. In this paper, we consider two approaches for constructing the initial training data. For the accuracy evaluation in Section 6.1, we construct initial labels from a corpus of time-stamped behaviors. However, this approach does not scale well. For the empirical study in Section 7, we also use clustering algorithms to identify behavior classes in spatial data-sets. Specifically, we use the density sampling algorithm, DBSCAN, as it performs well with small sample sizes and ambiguous sized clusters [10, 29]. For both methods, we use cross validation scoring to improve the accuracy of training data.

Security Service Provider: In practice, we found that the trained model is often specific to devices by the same manufacturer. To ensure HomeSnitch can detect behaviors for new devices, we propose the use of a security service provider (SSP) to share features and expand the set of supported smart home device behaviors. This approach allows the development of a model that expands as the heterogeneity of devices increases. Both IoT-Sentinel [18] and IoTurva [13] show promise as techniques for an SSP to collect normal and anomalous device activities from consumer deployments. HomeSnitch could use a publish-subscribe interface similar to virus reporting. For example, BitDefender [7] and Norton [23] offer subscription-based services for defending smart home devices.

Approaches that send information out of the home network raise concerns of privacy and data quality. To ensure the privacy of users, HomeSnitch can limit data collection to the features and labels collected in a smart home IoT deployment. When doing so, the labels associated with time-stamps must be appropriately protected (e.g., excluding day, hour, and minute information). Labels must also be associated with devices. Fortunately, the higher order bits of MAC addresses indicate the Organizationally unique identifier (OUI). End users can also be consulted to help identify devices. Finally, such crowd sourcing approaches must address data quality, e.g., via user reputation, voting, or threshold-based acceptance. We leave such considerations for future work.

Behavior Reporting: HomeSnitch offers novel behavior detection and descriptive reporting. Reporting provides an understanding of the function of newly detected behaviors. Appendix B displays a behavior report that predicts closest matching behaviors while offering behavior-centric details (i.e., frequency, duration, and throughput). This approach allows a user to generalize and discern the purpose of a previously unseen behavior. This approach enables identifying behaviors that have a high degree of similarity.

4.2 Context Monitor

Smart home devices are designed to connect and exchange data with one another. Devices connect directly through explicit communication channels or implicitly through back-end platforms. As illustrated by the Alexa API abuse in Section 2, users cannot always trust third party platforms to protect security and privacy. HomeSnitch's context monitor maintains the context and behaviors of all peer smart home devices on the network. It archives historical flow data (source, destination, transport layer ports, time-stamp, flow duration, and classification labels) in a database at the controller. Our context manager prototype currently provides the time period, the state of being encrypted or not, and the existence of a management device on the network at the time of the flow. In the future, we envision correlating flows to servers to identify the implicit communication occurring between devices or contrarily the lack of such communication. The context is used in three ways. First, it is provided in an activity report to the user in the HomeSnitch control interface. Second, it can be used for access control policies (see Section 4.3). Third, we envision a security service provider can collect aggregated historical context to help detect malicious device behaviors. We also envision a system that can identify when less secure devices are leveraged to attack more restricted devices.

4.3 Policy

The labels derived in Section 4.1 can be used to provide transparency of network traffic as well as access control. In this section, we describe our access control policy language. The typical user will never interface directly with our policy language. However, the expressiveness of the policy language corresponds to the high-level capabilities and may be built upon by enhanced agents.

HomeSnitch Syntax: HomeSnitch enforces policy rules that prevent unwanted device behaviors. HomeSnitch uses a *Device, Behavior, Context* $\rightarrow$ *Action* model to express access control policies.

Listing 1: Examples of HomeSnitch's configurable policy.

```
drop Ring-Doorbell, [Ring,Doorbell,Motion-Upload], (T: 0100-0300)
log WeMo-BabyCamera, [WeMo,BabyCamera,Firwmare-Update], (C: unencrypted)
reject LockState-Lock, [Lockstate,Lock,Unlock], (M: notpresent)
```

The *Device* predicate is a general description of the device (e.g., Ring-DoorbellCam, WeMo-MotionSensor, Nest-Alarm) that maps to a set of link layer MAC addresses for devices. Next, the *Behavior* predicate is a behavior triple, as defined in Section 4.1: (<vendor>, <model>, <activity>). For example, the triple (WeMo, Motion-Sensor, Upload-Motion) describes the behavior of a WeMo Motion Sensor uploading motion sensor data. Note that the actual string names need to be defined by either the security service provider or the end user. Finally, the *Context* predicate allows the user to describe complex scenarios based on information in the Context Manager (Section 4.2). Our prototype supports connection encryption, time period, and co-located management devices. When the criteria specified in the *Device*, *Behavior*, and *Context* are met, the rule implements the specified *Action* to enforce fine-grained access control to *alert, log, pass, drop, reject, or redirect.* Appendix A lists the context free grammar for the HomeSnitch policy language.

Policy Examples: Listing 1 shows three example policy rules. These examples demonstrate the flexibility to implement access controls for behaviors and context. The first rule prevents disclosing motion-upload data from a wireless doorbell during the early morning hours of 0100-0300. The second rule logs firmware updates over unencrypted connections. The third rule prevents a smart-lock from unlocking unless the management device (e.g., a smartphone) is attached to the smart home wireless network. Analogous to the popular If-This-Then-That model, we envision a community of shared recipes to address the array of smart home IoT threats [37].

Policy Enforcement: HomeSnitch generates OpenFlow Flow-Mods representing policy rules to enforce the fine-grained access control on smart home devices and behaviors. To accomplish this, HomeSnitch maintains a historical database of previously seen behaviors and flow-level information (i.e source address, source port, destination address, destination port). To enforce the policy primitive actions, HomeSnitch queries the database for the flow-level information (i.e., a single or set of ports and/or addresses) that can be used to describe the historical flows without conflicting with other behaviors. HomeSnitch uses this query to create flow modifications. Simple actions including *pass* or *drop* can be accomplished with a single flow modification. However, more complex actions (e.g., *reject* or *redirect*) are brought to the controller with distinct action labels (by overwriting the IP ToS as demonstrated in [24]). For the reject action, the controller forges a TCP reset packet. For the redirect action, the controller develops flow modifications using the *OFPActionSetField* primitive to re-write the source and destination MAC and IP Addresses for both the device and destination.

5 IMPLEMENTATION

We built HomeSnitch on top of a commodity wireless access point running an open source firmware that supports a virtual switch and the OpenFlow SDN protocol. This section details the key the components of HomeSnitch, including the data and control plane.

Table 2: Comparison of accuracy, recall and F1 scores for different supervised learning algorithms.

Model	Accuracy	Recall	F1 Score
k-Nearest Neighbor	99.32 %± .12	87.97 %± 2.16	86.35 %± 2.56
Gradient Boost	64.70 %± 42.10	53.55 %± 33.51	51.72 %± 32.72
Random Forest	**99.69 %± .06**	**94.66 %± 1.21**	**93.93 %± 1.40**

Data Plane: We implemented a prototype on a Linksys router, running our custom OpenWrt firmware. We integrated Open vSwitch Support (OVS) with OpenWrt by compiling OVS as a kernel module. Our prototype, running OVS 2.39, supports up to the OpenFlow 1.51 protocol specification. The router is configured to use simultaneous dual band with support for 802.11n (2.4 GHz) and 802.11ac (5GHz). We created a single OVS bridge to bind both WiFi interfaces and all local Ethernet interfaces, allowing OpenFlow to manage wireless connected devices as well as devices connected via Ethernet.

Control Plane: We implemented our SDN Security Orchestrator on a Raspberry Pi 3 Model B single board computer, connected via Ethernet to our router. The Raspberry Pi has a quad-core 64-bit ARM Cortex A53 clocked at 1.2 GHz. We flashed the Ubuntu 16.04 Operating System onto the Raspberry Pi and installed the necessary packages to support the Ryu component-based software defined networking framework. We developed our controller application as a Ryu management application in 6,503 lines of Python. Our Ryu application implements the logic described in Section 4 that identifies labeled smart home IoT device behaviors and implements the expressed policies of the end-user. We construct our machine learning models using the *RandomForestClassifier* and *DBSCAN* modules from the *scikit-learn* open source machine learning library [29].

6 EVALUATION

HomeSnitch identifies device behavior by analyzing communication flows. It trains supervised machine learning models of device/server dialogues, to classify network flows into behavior classifications, regardless of possible encryption and proprietary protocols. In this section, we empirically evaluate the performance of our prototype by answering the following research questions.

RQ1 (*Accuracy*): Is HomeSnitch effective in identifying and classifying smart home IoT device behaviors?

RQ2 (*Performance*): What are the performance impacts on throughput and jitter?

RQ3 (*Required Training*): How many samples are required to train HomeSnitch to classify different behaviors?

6.1 RQ1: Behavior Classification Evaluation

Data-set and Experimental Setup: We evaluated the accuracy of our behavior classification model against an independent corpus of IoT device behaviors. The YourThings [3] data-set included network captures from 46 devices, clustered into 148 semantic behaviors, connected to over 2,239 cloud endpoints between 13-19 April 2018. The data-set is representative of the broad smart home IoT market.

Accuracy: To evaluate accuracy, we compared the performance of Random Forest (our chosen model) against kNN (used in Peek-A-Boo [2]), and Gradient Boost (used in IoTSense [6]) against the data-set. We determined the accuracy through a stratified 10-fold

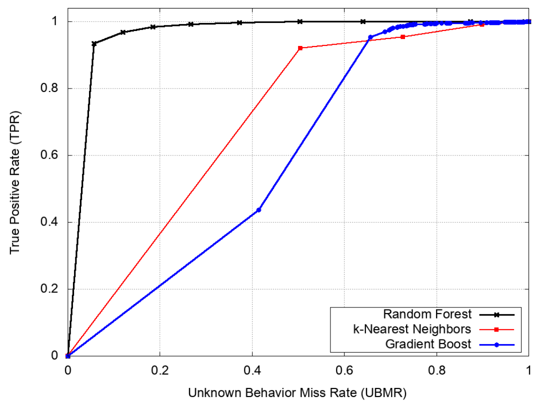


Figure 3: Receiver operating characteristics curve for comparison of different supervised learning algorithms abilities' to classify behaviors while detecting new behaviors.

cross-validation procedure against the labeled data-set [29]. Further, we calculated the recall and F1 for all models. Table 2 shows the results of our evaluation. As previously described in Section 4.1, our approach relies on a Random Forest classification model. Ensemble-based algorithms, including Random Forests and Gradient Boost, typical perform well for numerical features without scaling and perform implicit feature selection. Random Forest yielded the highest score using cross validation scoring. K-Nearest Neighbor (kNN), known for high accuracy and no *a priori* assumption, classified the labeled data-set with high accuracy and moderate recall.

Unknown Behavior Miss Rate: A key feature of HomeSnitch is identifying when new behaviors occur and finding the closest match. To measure the effectiveness of new behavior detection and resistance to under-fitting, we measure the *Unknown Behavior Miss Rate* (UBMR). UMBR is the percentage of data-points in a given data-set belonging to previously unseen classes that fail to be identified as a new class. To calculate UBMR, we used a leave-one-out approach in which we iterated through each of the different behavior classes. For each behavior, we removed all flows corresponding to it from the training data-set. We then trained the classifier using the remaining behaviors and then attempted to label flows belonging to the examined behavior class, determining the confidence score of each classification. We compared results of both kNN, Gradient Boost, and Random Forest classifiers. Since we are attempting to identify unknown behaviors, a low confidence score indicates a better result (more likely to be identified as a new class).

In our evaluation, Random Forests provided better calibrated confidence scores of unknown behavior over kNN and Gradient Boost. In contrast, the other approaches attempted to pull data points belonging to unknown behavior classes into existing classifications. This result supported our initial intuition that an bagging-based ensemble classifier would be more resistant to under-fitting. Figure 3 depicts a ROC curve demonstrating the impact of threshold selection on true positive rate (TPR) and UBMR for the YourThings data-set. Our results indicate that a threshold of 0.89 provided the best trade-off between accurately classifying behaviors against identifying unknown behaviors with a 11.96% UBMR and 96.82% TPR. However, we acknowledge that a TPR of 96.82% would certainly raise false alerts for a real-world appliance. Thus, the selection of the threshold must consider the balance of security and usability.

Table 3: Performance Impact on a Single Connection

Test	Throughput (Mbps)	Latency (ms)
Physical (Baseline)	15.848 ± 0.087	183.988 ± 2.154
Physical (RYU SimpleSwitch)	15.332 ± 0.096	278.552 ± 3.024
Physical (HomeSnitch)	15.375 ± 0.238	283.452 ± 4.333

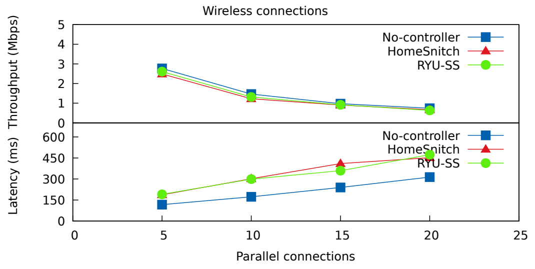


Figure 4: Comparison of the impact of multiple wireless connections on latency and throughput for a controller-free wireless router, a simple layer-2 controller application and HomeSnitch.

6.2 RQ2: Network Performance Evaluation

In this section, we examine the performance associated with monitoring network flows. Setting up a testbed with many physical devices is prohibitively complex. Therefore, we perform experiments in two testbeds: physical and emulated. For the physical testbed, we simulate many IoT devices by having one device increase its connection rate to match that of many devices. For the emulated testbed, we emulate many devices in a *de facto* SDN emulator and confirm the efficacy of the results of the physical testbed.

Experimental Setup: We used the *iPerf* toolkit to actively measure throughput and jitter (i.e., the latency variations among packets). To parameterize *iPerf*, we drew from the flow duration, bandwidth, and protocols of devices of the YourThings data-set discussed in Section 6.1. We found an average smart home IoT flow duration of 22.72 seconds. 64.72% of devices connected for less than a second while 17.64% of flows exceeded a minute. The maximum flow in the YourThings data-set lasted 286.32 seconds. Connections occurred both with and without encryption. We compared the results of our HomeSnitch prototype against a SDN controller application performing MAC-layer matching, and a traditional MAC-layer switch. To replicate a traditional MAC-layer controller we connected our virtual switch to the RYU *Simple_Switch.py* application included in the RYU documentation. To measure against a baseline, we configured a separate router with the default firmware.

Physical Testbed: We measured the performance of HomeSnitch over 1,000 iterations using the iPerf toolkit and report average results with a 95% confidence interval. Each flow lasted 10 seconds, and we measured the achieved throughput and jitter at 100ms intervals. Although we cannot completely guarantee outside interference in our experiment, we took conservative actions to prevent frame interference on the wireless standard. The IEEE 802.11n 2.4 GHz

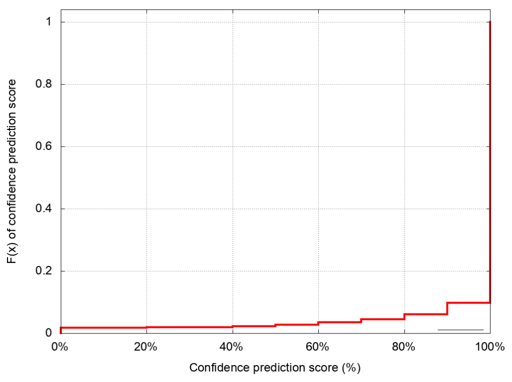


Figure 5: CDF for the confidence scores for the 148 predicted behaviors in the 46 device YourThings data-set.

wireless standard supports three non-overlapping 20 MHz width channels in the US (1,6, and 11). We connected the workstation to channel 11, as channels 1 and 6 were used by neighboring networks. We disconnected all other wireless devices from the workstation to avoid introducing collisions. Table 3 shows the impact on a single wireless device. Figure 4 illustrates the impact of additional devices (simulated by parallel connections) on throughput and latency. As depicted in Table 3 and Figure 4, there is a 100ms latency overhead incurred for using an SDN controller HomeSnitch implementation to process flows. This is consistent for both the SimpleSwitch and HomeSnitch implementations. However, this has little impact on throughput, retaining 97.01% of the performance.

Emulated Testbed: To understand the performance trade-off associated with the increasing scale of devices, we emulated twenty-five unique wireless stations using the *MiniNet WiFi* framework [11]. We hosted this emulation on a workstation physically connected to our router. Stations were created using the 802.11 protocol, placed on the same channel, with randomized distances from the access point. We measured the throughput and jitter for connections to twenty-five unique iPerf server instances. Our results mirrored the results on the physical testbed with a nominal effect on throughput.

6.3 RQ3: Required Training

We constructed the cumulative distribution function graph represented in Figure 5 by recording the predicted class probability scores for samples from each of the 148 behaviors in our model. Figure 5 plots our results comparing the confidence prediction score against the cumulative distribution function. The resulting graph demonstrates our model predicts behaviors with a high degree of confidence. In our experiments, we discovered the monolithic nature of smart home devices require few samples to train our model to identify multiple behavior classes. To evaluate the number of required training samples, we used a test-training set approach in which we iterated through 200 samples for each behavior. For each unique class, we removed all the samples from the class. We then incrementally added back each sample as a member of the behavior class. At each increment, we averaged the prediction score of the remaining unlabeled samples from the behavior class. We observed that behaviors required an average of 58.82 and 62.28 samples to achieve 90% and 95% prediction confidence.

7 EMPIRICAL STUDY

HomeSnitch uses supervised learning to classify features from connection-oriented application data unit exchanges. The behavior classification allows HomeSnitch to inform the user of device activities and restrict unwanted activities. In this section, we empirically evaluate HomeSnitch's ability to classify device behaviors, identify unknown behavior, and prevent an unwanted behavior by answering the following research questions.

RQ4 (*Behavior Classification*): How effective is HomeSnitch at classifying behaviors in a typical smart home environment?
RQ5 (*Malicious Behavior*): Is it possible for HomeSnitch to identify a malicious attack scenario?
RQ6 (*Policy Expression*): Given a known behavior, can a HomeSnitch policy prevent the behavior?

Experimental Setup: Our experimental setup consisted of 20 devices in active use at the residence of one of the authors of the paper. These devices represent different categories including automation, fitness, appliance, and security camera functions. We configured HomeSnitch to log and classify all smart home IoT device traffic in our environment over two days. We correlated user activities (e.g., turning on/off lights, resetting the coffee pot filter, and triggering motion events) to the observed behaviors classes.

7.1 RQ4: Behavior Classification

We observed HomeSnitch's ability to manage security and privacy by classifying behaviors and enforcing policies against 20 devices on the home-network of one of the primary authors.

Encryption Observations: HomeSnitch classifies behaviors regardless of the use of encryption or proprietary protocols. In our empirical evaluation, we observed that 76.22% of smart home traffic in our environment was encrypted. These findings support the observations by Sivanathan et al. [32], who identified 65% of IoT device and 70% of all traffic on the Internet is encrypted. This result confirms that packet payload inspection cannot effectively classify IoT application layer protocols and behaviors.

Labeling Behaviors: We applied descriptive labels to the over 34,000 application exchanges recorded over two days. To apply descriptive labels to behavior clusters, we correlated our user-driven actions and leveraged insight from behavior reports. Utilizing our behavior reporting framework, and correlating user-driven activity, we were able to assign descriptive names to behavior triples: *(<vendor>, <model>, <activity>)*. For example, we correlated motion sensing from the Ring Companion application to a specific cluster and labeled the cluster as *(Ring, Doorbell, MotionUpload)*.

Results: Figure 6 shows the top 35 behaviors from our empirical evaluation, which accounts for over 90.39% of all application data exchanges over the period. Heartbeat behaviors (i.e., a fixed pattern that occurs periodically) account for over 55.09% of all network flows. Several devices have multiple Heartbeats. For example, the Canary device has two similar Heartbeat behaviors that poll at ten and one minute intervals. Heartbeats enable perpetual connectivity for devices by establishing meet-in-the-middle connections on

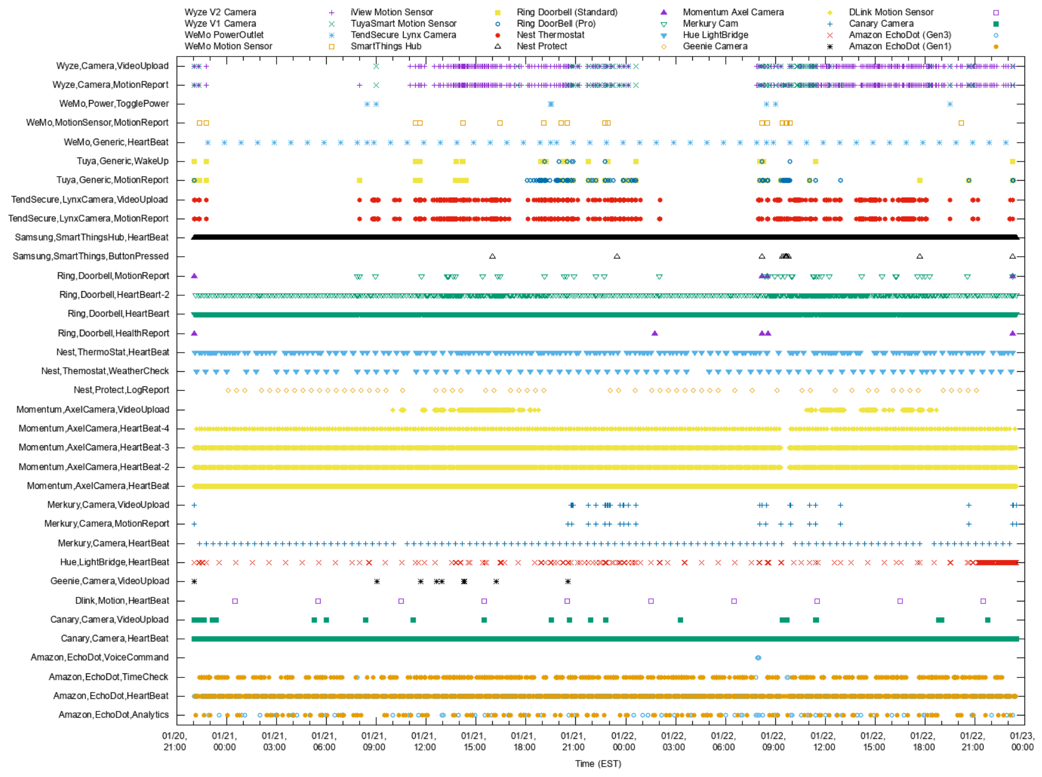


Figure 6: Top 35 behaviors over 48 hours sample. HomeSnitch provides semantic behavior transparency and reporting for smart home devices. Note that all clusters were identified by HomeSnitch without training on prior data-sets. The semantic labels were manually created using HomeSnitch's behavior reporting feature to determine and assign descriptive labels.

Table 4: HomeSnitch has accuracy across deployments by using features that are transport and destination agnostic.

Device	ADUs	Accuracy
Amazon Echo Dot	302	90.40%
B&D Crockpot	914	98.47%
Canary Security Camera	15,426	98.88%
Hue Light Bridge	2,362	99.58%

cloud platforms. Our results demonstrate that user driven actuator control (e.g., turning on lights) and sensory reporting (motion, video uploads) have different signatures from behaviors and can be accurately predicted. Evaluating the labeled data-set, we record a 99.40% cross validation score of the 34,109 labeled behaviors.

Generality of Model: To evaluate how our approach extends across deployments, we classified traffic captures from four devices in the YourThings data-set that overlapped our empirical network. We detected the behaviors from the Echo Dot, Crock-pot, Canary Camera, and Hue Light Bridge with high confidence to samples we collected separately as depicted in Table 4. In contrast, we had difficulty classifying a Ring Doorbell. Subsequently, we learned there was a model mismatch between the original and professional versions of the device. The functionality of the newer device, and by extension the behaviors, differed from the original.

7.2 RQ5: Malicious Behavior

We evaluated HomeSnitch's ability to classify a malicious behavior by simulating an attack of a smart plug. The TP-Link HS110 is a smart plug that an end-user can control via an app. By interacting with the smart plug, the app can monitor energy consumption and schedule actions. Stroetmann and Esser [34] identified that the app to device communication consisted of a protocol encrypted with an XOR cipher. We hosted a malicious copy of the device firmware on our server. Further, we modified Stroetmann's work to build a tool that forced the device to download the malicious firmware. Next, we trained HomeSnitch to recognize a firmware download to the device by observing downloads of benign firmware.

Results: HomeSnitch identified each individual attempt to download the firmware and subsequently logged the activity. This alert offers insight to a user that their device has become compromised by downloading firmware from a malicious server. HomeSnitch provides the ability to train the system and write rules to prevent against such a download (i.e., by white-listing downloads to only the official site). However, we envision a service provider (similar to virus reporting) could provide both behavior training signatures and rules for enforcing behaviors for specific devices.

7.3 RQ6: Policy Expression

To illustrate the effectiveness of HomeSnitch's policy enforcement mechanism, we demonstrated the ability for HomeSnitch to create rules to block Ring Doorbell motion detection for a ten-minute period to protect privacy. HomeSnitch provides a web interface to apply policy actions such as disabling motion detection for specific IoT devices. After enabling a rule for our doorbell, we attempted to trigger the motion detection. No motion detection was alerted to the user or logged in the history of the smartphone app. The flows were stopped by the data plane. The flow modifications alerted that 407 packets (31,462 total bytes) were dropped when the doorbell attempted to upload the motion events. When our policy expired, the flow modification was removed and we observed that packets flowed unrestricted to the Ring servers. We generated new motion events and noted that the new events were reported in the app history. However, somewhat surprisingly, we identified that the doorbell did not buffer any of the events that occurred while it was restricted by our policy. Next, we applied a similar rule preventing our D-Link Motion Sensor from uploading motion sensing data. As a result, 26 packets (1,936 total bytes) were dropped. This prevented motion uploads from being displayed on our smartphone's app. However, when the rule expired, the motion sensor uploaded the buffered motion data. This contrast illustrates that the effectiveness of policy enforcement ultimately relies on understanding of the overall buffering and communications model of each IoT devices. [25] offers further insight on restricting per-device behavior.

8 LIMITATIONS

Mimicry Attacks: Our supervised machine learning approach leverages an ensemble-method classifier to identify traffic patterns that represent device behaviors. Our work does not address the case of an informed adversary that is able to compromise a device and mimic the traffic patterns of permitted behaviors. An adversary could mask a command and control channel inside traffic. While this attack would be successful in misleading our classification model, an attacker would still need to learn the contents of the policy to overcome its restrictions. Further, the policy could be specified in such a way to permit behaviors to only specific trusted servers. However, we do not entirely rule out the feasibility for an attacker to both mislead the classification and overcome the policy. In this case, we must consider complementary approaches including device identity, anti-tampering, and device attestation [1, 17, 41]

Partial flow analysis: HomeSnitch requires capturing the full network flow for precise classification. To enable real-time enforcement, we construct flow modification instructions based on historical flow data. However, given our observations of the simplistic nature of smart home device behaviors, we believe it feasible to construct a stochastic model that could real-time predict behaviors with a brief packet queue. We reserve this topic for future work.

9 RELATED WORK

The closest related work to HomeSnitch are efforts designed to identify IoT devices based on their network traffic patterns. IoTSentinel [18] use static features such as protocol types, IP addresses, and port addresses to fingerprint devices. However, these features alone cannot differentiate semantic behaviors performed by the fingerprinted devices. In concurrent work, IoTSense [6] expands on IoTSentinel by using device behaviors to fingerprint IoT devices. To some extent, the features used by IotSense are similar to those used by HomeSnitch. However, their notion of "behavior" is different. They do not seek to actually label behaviors. Rather, their goal is strictly fingerprinting devices. This difference in goal causes IoTSense to not consider *new* behaviors when classifying traffic. As we show in Section 6, the Gradient Boost algorithm used by IoTSense performs poorly when identifying new behaviors.

Also concurrent to our work, Peek-a-Boo [2] seeks to identify behaviors, such as turning on a light switch, from network traffic. Peek-a-Boo is an attack technique designed to invade the privacy of smart home users. In contrast, HomeSnitch seeks to provide a defense. In doing so, HomeSnitch considers a broader definition of behaviors, including not only user actions (e.g., turning on a switch), but also hidden activities performed by devices (e.g., heartbeats, analytics). Interestingly, Peek-a-Book also chooses kNN over Random Forests to classify traffic; however, it does not consider the detection of previously unclassified (i.e., new) behaviors. HoMonit [40] also seeks to infer behaviors from encrypted wireless traffic. However, their work is narrowly focused on the communication between devices and a SmartThing's hub to implement anomaly detection.

Several solutions have also been proposed for the detection and prevention of attacks in smart home IoT networks. DioT[22] and IoTPOT [28] focused exclusively on the classification of anomalous traffic in order to detect compromised IoT device bot-net activity. Existing solutions are limited in their ability to enforce policy that considers behavior and context [39]. ContextIoT [15] provides a vendor-specific solution for the SmartThings platform and does not generalize. IDIoT [5] white-lists devices to the minimal set of device behaviors based on flow information. However, the use of perpetual cloud connections and changing functionality of devices hinders the practicality of exclusively using flow data. In contrast, HomeSnitch offers behavioral transparency for both known and anomalous behaviors. This approach allows us to realize behavior transparency and implement practical context-aware policy.

10 CONCLUSION

This paper presented HomeSnitch, a building block for enhancing transparency and control for end-users by classifying smart home device communication by *behaviors*. HomeSnitch leverages the vantage point of software-defined networking to monitor device/server dialogue exchanges to classify device behaviors and identify new and unknown behaviors. By selecting destination and content-agnostic features, HomeSnitch can classify device behaviors even in the presence of encryption and proprietary protocols. We implemented HomeSnitch on a commodity wireless router and built upon SDN primitives to enforce network access controls. Using a Random Forest Classifier, we achieved over 99% accuracy for classifying behaviors from an independent data-set. Through these efforts, we demonstrated the effectiveness of network-level services to classify behaviors and enforce control on IoT devices.

ACKNOWLEDGEMENTS

This material is based upon work supported in part by funding from the German Research Foundation (Deutsche Forschungs Gemeinschaft - DFG), as part of project S2 within the CRC 1119 CROSSING. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] Tigist Abera, N. Asokan, Lucas Davi, Farinaz Koushanfar, Andrew Paverd, Ahmad-Reza Sadeghi, and Gene Tsudik. 2016. Invited - Things, Trouble, Trust: On Building Trust in IoT Systems. In *Annual Design Automation Conference (DAC)*. ACM, Austin, Texas, 121:1–121:6.

[2] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and A Selcuk Uluagac. 2018. Peek-a-Boo: I see your smart home activities, even encrypted! https://arxiv.org/pdf/1808.02741.pdf.

[3] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. SoK: Security Evaluation of Home-Based IoT Deployments. In *IEEE Security and Privacy*. IEEE, San Francisco, CA, 20–25.

[4] Sami Ayyorgun and Wu-chun Feng. 2004. A Deterministic Definition of Burstiness For Network Traffic Characterization. In *International Conference on Computer Communications and Networks*. Los Alamos National Laboratory, Los Alamos, NM, 1–29.

[5] David Barrera, Ian Molloy, and Heqing Huang. 2017. IDIoT: Securing the Internet of Things like it's 1994. https://arxiv.org/abs/1712.03623.

[6] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. 2018. Behavioral Fingerprinting of IoT Devices. In *Workshop on Attacks and Solutions in Hardware Security (ASHES)*. ACM, Toronto, Canada, 41–50.

[7] Bitdefender. 2018. BOX 2: The new way to secure your connected home. https://www.bitdefender.com/box/.

[8] Alberto Dainotti, Antonio Pescape, and Kimberly C Claffy. 2012. Issues and future directions in traffic classification. In *IEEE Network*, Vol. 26. IEEE.

[9] Nitesh Dhanjani. 2015. *Abusing the internet of things: blackouts, freakouts, and stakeouts*. O'Reilly Media, Inc., Sebastopol, CA.

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. http://dl.acm.org/citation.cfm?id=3001460.3001507. In *International Conference on Knowledge Discovery and Data Mining (KDD)*. AAAI Press, Portland, Oregon, 226–231.

[11] Ramon Fontes and Christian Rothenberg. 2018. Mininet-WiFi: Emulator for Software-Defined Wireless Networks. https://github.com/intrig-unicamp/mininet-wifi.

[12] Thomas Fox-Brewster. 2017. Here's How The CIA Allegedly Hacked Samsung Smart TVs – And How To Protect Yourself. https://www.forbes.com/sites/thomasbrewster/2017/03/07/cia-wikileaks-samsung-smart-tv-hack-security.

[13] Ibbad Hafeez, Aaron Yi Ding, and Sasu Tarkoma. 2017. IOTURVA: Securing Device-to-Device (D2D) Communication in IoT Networks. In *Workshop on Challenged Networks (CHANTS)*. ACM, Snowbird, Utah, 1–6.

[14] Felix Hernández-Campos, Kevin Jeffay, and F Donelson Smith. 2007. *Modeling and generating TCP application workloads*. Technical Report. 280–289 pages.

[15] Yunhan Jack Jia, Qi Alfred Chen, Shiqi Wang, Amir Rahmati, Earlence Fernandes, Z Morley Mao, Atul Prakash, and Shanghai JiaoTong Unviersity. 2017. ContexIoT: Towards Providing Contextual Integrity to Appified IoT Platforms. In *Network and Distributed System Security Symposium (NDSS)*. San Diego, CA. https://doi.org/10.14722/ndss.2017.23051

[16] Hyunchul Kim, Kimberly C Claffy, Marina Fomenkov, Dhiman Barman, Michalis Faloutsos, and KiYoung Lee. 2008. Internet traffic classification demystified: myths, caveats, and the best practices. In *Conference on emerging Networking EXperiments (CoNEXT)*. ACM, 11.

[17] Parikshit Mahalle, Sachin Babar, Neeli R. Prasad, and Ramjee Prasad. 2010. Identity Management Framework towards Internet of Things (IoT): Roadmap and Key Challenges. In *Recent Trends in Network Security and Applications*, Natarajan Meghanathan, Selma Boumerdassi, Nabendu Chaki, and Dhinaharan Nagamalai (Eds.). Springer Berlin Heidelberg, 430–439.

[18] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. R. Sadeghi, and S. Tarkoma. 2017. IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT. In *International Conference on Distributed Computing Systems (ICDCS)*. IEEE, IEEE, Atlanta, GA, 2177–2184.

[19] Brian Naylor. 2016. This Doll May Be Recording What Children Say, Privacy Groups Charge. https://n.pr/2i2FtfS.

[20] Soraya Sarhaddi Nelson. 2017. Germany Bans Ḿy Friend CaylaD́oll Over Spying Concerns. https://n.pr/2kRI5do.

[21] Lily Hay Newman. 2018. Turning an Echo Into a Spy Device Only Took Some Clever Coding. https://www.wired.com/story/amazon-echo-alexa-skill-spying.

[22] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Minh Hoang Dang, N Asokan, and Ahmad-Reza Sadeghi. 2018. DÏoT: A Crowdsourced Self-learning Approach for Detecting Compromised IoT Devices. https://arxiv.org/pdf/1804.07474.pdf.

[23] Norton. 2018. Core Secure WiFi Router. https://us.norton.com/core.

[24] TJ OConnor, William Enck, W Michael Petullo, and Akash Verma. 2018. Pivotwall: Sdn-based information flow control. In *Proceedings of the Symposium on SDN Research*. ACM, San Francisco, CA.

[25] TJ OConnor, William Enck, and Bradley. Reaves. 2019. Blinded and Confused: Uncovering Systemic Flaws in Device Telemetry for Smart-Home Internet of Things. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*. ACM, Miami,FL.

[26] David Olshefski and Jason Nieh. 2006. Understanding the Management of Client Perceived Response Time. In *Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*. ACM, Saint Malo, France, 240–251.

[27] David P. Olshefski, Jason Nieh, and Erich M. Nahum. 2004. ksniffer: Determining the Remote Client Perceived Response Time from Live Packet Streams. In *Symposium on Operating System Design and Implementation (OSDI)*. USENIX Association, San Francisco, California, 333–346.

[28] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. 2015. IoTPOT: Analysing the Rise of IoT Compromises. In *Conference on Offensive Technologies (WOOT)*. USENIX Association, Washington, D.C., 9–9.

[29] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. In *J. Mach. Learn. Res.*, Vol. 12. JMLR.org, 2825–2830.

[30] A. R. Sadeghi, C. Wachsmann, and M. Waidner. 2015. Security and privacy challenges in industrial Internet of Things. In *Design Automation Conference (DAC)*. IEEE, 1–6. https://doi.org/10.1145/2744769.2747942.

[31] Helen Rebecca Schindler, Jonathan Cave, Neil Robinson, Veronika Horvath, Petal Hackett, Salil Gunashekar, Maarten Botterman, Simon Forge, and Hans Graux. 2012. Examining Europe's Policy Options to Foster Development of the 'Internet of Things'. http://www.rand.org/pubs/research_reports/RR356.html.

[32] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman. 2017. Characterizing and classifying IoT traffic in smart cities and campuses. In *Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 559–564.

[33] Vijay Sivaraman, Dominic Chan, Dylan Earl, and Roksana Boreli. 2016. Smart-Phones Attacking Smart-Homes. In *Conference on Security & Privacy in Wireless and Mobile Networks (WiSec)*. ACM, Darmstadt, Germany, 195–200.

[34] ubomir Stroetmann and Tobias Esser. 2017. Reverse Engineering the TP-Link HS110. https://www.softscheck.com/en/reverse-engineering-tp-link-hs110/.

[35] Jeff Terrell, Kevin Jeffay, F. Donelson Smith, Jim Gogan, and Joni Keller. 2009. Passive, Streaming Inference of TCP Connection Structure for Network Server Management. In *Traffic Monitoring and Analysis*, Maria Papadopouli, Philippe Owezarski, and Aiko Pras (Eds.). Springer Berlin Heidelberg, 42–53.

[36] Turris. 2019. Omnia: More than just a router. https://omnia.turris.cz/en/.

[37] Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L. Littman. 2016. Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. http://doi.acm.org/10.1145/2858036.2858556. In *Conference on Human Factors in Computing Systems (CHI)*. ACM, San Jose, California, 3227–3231. https://doi.org/10.1145/2858036.2858556

[38] Nigel Williams, Sebastian Zander, and Grenville Armitage. 2006. A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification. In *SIGCOMM Computer Communication Review*, Vol. 36. ACM, 5–16.

[39] Tianlong Yu, Vyas Sekar, Srinivasan Seshan, Yuvraj Agarwal, and Chenren Xu. 2015. Handling a Trillion (Unfixable) Flaws on a Billion Devices: Rethinking Network Security for the Internet-of-Things. In *Workshop on Hot Topics in Networks (HotNets)*. ACM, Philadelphia, PA, 5:1–5:7.

[40] Wei Zhang, Yan Meng, Yugeng Liu, Xiaokuan Zhang, Yinqian Zhang, and Haojin Zhu. 2018. HoMonit: Monitoring Smart Home Apps from Encrypted Traffic. In *SIGSAC Conference on Computer and Communications Security*. ACM, 1074–1088.

[41] X. Zhang, R. Adhikari, M. Pipattanasomporn, M. Kuzlu, and S. Rahman. 2016. Deploying IoT devices to make buildings smart: Performance evaluation and deployment experience. In *World Forum on Internet of Things (WF-IoT)*. IEEE, 530–535. https://doi.org/10.1109/WF-IoT.2016.7845464

A HOMESNITCH POLICY GRAMMAR

$$\langle\text{rule}\rangle ::= \langle\text{action}\rangle\ \langle\text{device}\rangle\ \langle\text{behavior}\rangle\ \langle\text{context}\rangle \tag{1}$$

$$\langle\text{action}\rangle ::= \langle\text{alert}\rangle \mid \langle\text{log}\rangle \mid \langle\text{pass}\rangle \mid \langle\text{drop}\rangle \mid \langle\text{reject}\rangle \mid \langle\text{redirect}\rangle \tag{2}$$

$$\langle\text{device}\rangle ::= \langle\text{vendor}\rangle\text{“}\texttt{-}\text{”}\langle\text{model}\rangle \tag{3}$$

$$\langle\text{vendor}\rangle ::= \langle\text{const}\rangle \tag{4}$$

$$\langle\text{model}\rangle ::= \langle\text{const}\rangle \tag{5}$$

$$\langle\text{behavior}\rangle ::= \langle\text{vendor}\rangle\ \langle\text{model}\rangle\ \langle\text{activity}\rangle \tag{6}$$

$$\langle\text{activity}\rangle ::= \langle\text{const}\rangle \tag{7}$$

$$\langle\text{context}\rangle ::= \langle\text{time\_ctx}\rangle \mid \langle\text{enc\_ctx}\rangle \mid \langle\text{mgt\_ctx}\rangle \tag{8}$$

$$\langle\text{time\_ctx}\rangle ::= \text{“}\texttt{(T:}\text{”}\ \langle\text{time}\rangle\ \text{“}\texttt{-}\text{”}\ \langle\text{time}\rangle\ \text{“}\texttt{)}\text{”} \tag{9}$$

$$\langle\text{enc\_ctx}\rangle ::= \text{“}\texttt{(C:}\text{”}\ \langle\text{encrypted}\rangle \mid \langle\text{unencrypted}\rangle\ \text{“}\texttt{)}\text{”} \tag{10}$$

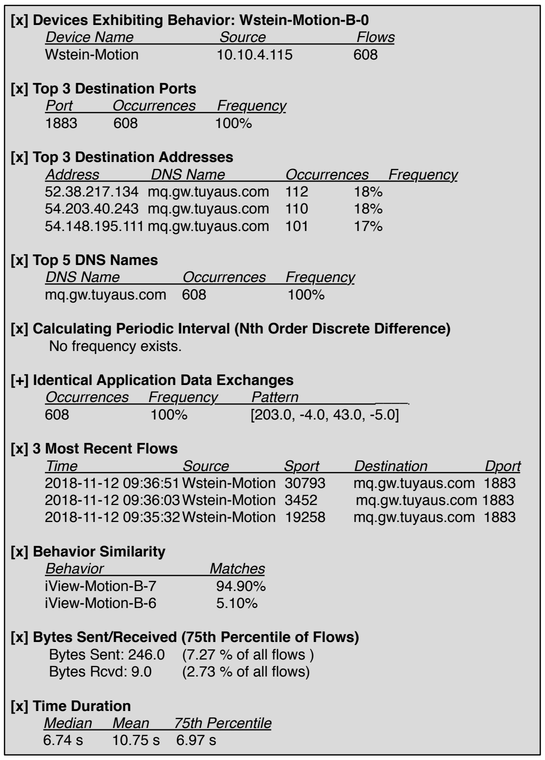
$$\langle\text{mgt\_ctx}\rangle ::= \text{“}\texttt{(M:}\text{”}\ \langle\text{present}\rangle \mid \langle\text{notpresent}\rangle\ \text{“}\texttt{)}\text{”} \tag{11}$$

$$\langle\text{time}\rangle ::= \texttt{[0-9][0-9]}\ \text{“}\texttt{:}\text{”}\ \texttt{[0-9][0-9]} \tag{12}$$

$$\langle\text{const}\rangle ::= \text{“}\texttt{'}\text{”}\texttt{[A-Za-z0-9\_.]+}\text{“}\texttt{'}\text{”} \tag{13}$$

Figure 7: HomeSnitch Syntax in BNF

B HOMESNITCH BEHAVIOR REPORTING



```
[x] Devices Exhibiting Behavior: Wstein-Motion-B-0
     Device Name               Source              Flows
     Wstein-Motion             10.10.4.115         608

[x] Top 3 Destination Ports
     Port      Occurrences    Frequency
     1883      608            100%

[x] Top 3 Destination Addresses
     Address        DNS Name            Occurrences   Frequency
     52.38.217.134  mq.gw.tuyaus.com    112           18%
     54.203.40.243  mq.gw.tuyaus.com    110           18%
     54.148.195.111 mq.gw.tuyaus.com    101           17%

[x] Top 5 DNS Names
     DNS Name            Occurrences    Frequency
     mq.gw.tuyaus.com    608            100%

[x] Calculating Periodic Interval (Nth Order Discrete Difference)
      No frequency exists.

[+] Identical Application Data Exchanges
     Occurrences    Frequency      Pattern
     608            100%           [203.0, -4.0, 43.0, -5.0]

[x] 3 Most Recent Flows
     Time                Source          Sport     Destination         Dport
     2018-11-12 09:36:51 Wstein-Motion   30793     mq.gw.tuyaus.com    1883
     2018-11-12 09:36:03 Wstein-Motion   3452      mq.gw.tuyaus.com    1883
     2018-11-12 09:35:32 Wstein-Motion   19258     mq.gw.tuyaus.com    1883

[x] Behavior Similarity
     Behavior                  Matches
     iView-Motion-B-7           94.90%
     iView-Motion-B-6           5.10%

[x] Bytes Sent/Received (75th Percentile of Flows)
      Bytes Sent: 246.0    (7.27 % of all flows )
      Bytes Rcvd: 9.0      (2.73 % of all flows)

[x] Time Duration
     Median    Mean      75th Percentile
     6.74 s    10.75 s   6.97 s
```

Figure 8: HomeSnitch provides descriptive reports of behavior characteristics and identifies similar behaviors.