Detection of Anomalous User Activity for Home IoT Devices[a]

Vishwajeet Bhosale[1] Lorenzo De Carli[2], and Indrakshi Ray[1]
[1]*Colorado State University, Fort Collins CO 80523, USA*
[2]*Worcester Polytechnic Institute, Worcester MA 01609, USA*

Keywords: IoT Security, IoT Traffic Classification, IoT Privacy.

Abstract: Home IoT devices suffer from poor security, and are easy to commandeer for unskilled attackers. Since most IoTs cannot run host-based detection, detecting compromise via analysis of network traffic is in many cases the only viable option. Unfortunately, traditional Deep Packet Inspection techniques are not applicable: many IoT devices encrypt their traffic and common attacks (e.g., credential stuffing) cannot be described via signatures. Anomaly detection on traffic features, while effective to identify egregious misbehavior (e.g., a DDoS) cannot identify privacy violations, where an attacker triggers legitimate functions (e.g., streaming video, unlocking a door), but without consent of the user. In this paper, we propose a novel anomaly detection technique based on the analysis of user activities. Our approach builds a model to identify *user-performed activities* on the device from packet sequences, and uses unsupervised learning to identify deviations from normal user behavior in activity sequences. Thus, it can flag situations where an attacker misuses an IoT device, even when such attacks do not involve protocol-level exploits and do not result in significant anomalies in traffic-level features. Preliminary results show that our approach can effectively map device traffic to activities, and suggest that such activities can be used to distinguish malicious and benign users.

1 INTRODUCTION

Many home IoT devices suffer from poor security. IoT manufacturers tend to have limited experience in secure programming, which results in poor code written for firmware, front end application, communicating protocols and APIs. Even if mistakes are identified, updating the devices is in many cases difficult due to limited connectivity and lack of user awareness. Furthermore, many devices are shipped with default or weak passwords, in some cases hard-coded (Hall, 2018).

The problem is further compounded by the significant privacy risks inherent in IoT usage. Devices are installed inside the home and have various types of sensors (e.g. camera) as well as vast data collection capabilities. Some devices are also actuators, influencing the physical space around them (e.g., a thermostat). Given these premises, it is not surprising that IoT devices can enable theft (Osborne, 2020) and have been used as vectors for privacy violations, up to serious domestic abuse cases (Bowles, 2018). We collectively term this class of abuses *privacy invasions*.

Detecting privacy invasions is difficult. Host-based threat protection tools are unsuitable for IoTs, which are resource-constrained and hard to upgrade. More promising is the analysis of device network traffic to identify compromise. However, traditional techniques are unlikely to be effective for this threat model. DPI tools work by identifying protocol-level misuse and byte-level patterns (e.g., shellcode) which are indication of compromise. Unfortunately, many attacks do not involve any such misuse; in many cases, the attacker simply takes control of a device using a known or easy-to-guess password (Goodin, 2019), or bypasses authentication (Reynolds, 2013). Furthermore, a large fraction of modern IoT device traffic is encrypted and inaccessible to DPI. Traffic-level anomaly detection (e.g., (Mirsky et al., 2018)) works even in the presence of encryption, but it has other limitations. While such a detector can easily identify egregious misbehavior (e.g., an attacker using a device to commit DDoS), privacy violations involve an attacker using a device in its intended way (e.g., change the setting on a thermostat), which is unlikely to generate any useful anomaly signal purely

[a]This work was supported in part by funding from NSF under Award Number CNS 1822118, CableLabs, AMI, NIST, Cyber Risk Research, Statnett and from Cyber Security Center supported by State of Colorado.

at the network traffic level.

In this paper, we propose a novel approach to IoT anomaly detection that focuses on *user activities*, rather than traffic features. Our core observation is that, due to the semantic gap between attack activity and its footprint in terms of packets, it is difficult for a traditional detector to distinguish benign and anomalous activity. We solve this problem by lifting the analysis at the level of user activities, i.e. discrete, basic operations a user can initiate by remotely operating the device (e.g., streaming video from a smart camera). A detector working at the level of activities can easily identify abnormal user behavior (e.g., use of functionality not normally triggered by the legitimate owner).

The first challenge we tackle is that, for our detector to work, activities must first be inferred from network traffic. Establishing such mapping requires extracting a large amount of traffic from a given device, while labeling each flow with the activity that caused it. For the purpose, we built an infrastructure enabling us to trigger a large number of scripted activities for a variety of IoT devices, while capturing traffic labeled with the corresponding activity. This resulted in a 19.8-GB traffic dataset which we plan to release to foster further experimentation. Once labeled traffic is available, a reliable mapping must be established between flows and activities. For this purpose, we train a random-forest classifier to map packet sequences to activities. Finally, patterns of device use are user-specific, and should be learned, ideally in an unsupervised fashion. We use clustering to identify recurring sequences of user activities, and sequences of activities which deviate from the expected behavior.

Preliminary results are promising: we report accuracy in the range of 86%-98% for activity identification. We also built a proof of concept tool to perform anomaly detection, and present an example scenario to demonstrate its working.

2 Related Work

Anomaly detection for IoT devices is a widely researched area (Chandola et al., 2009). Approaches based on both supervised learning (Alrashdi et al., 2019; Pacheco et al., 2019) and unsupervised learning (Bhatia et al., 2019; Hoang and Duong Nguyen, 2019; Alhaidari and Zohdy, 2019) have been proposed. Furthermore, (Hamza et al., 2018) propose signature-based detection based on manufacturer usage descriptions. (Jung et al., 2020; Myridakis et al., 2017) build a power consumption model using Convolution Neural Networks (CNNs) to detect IoT devices turned into botnet. Finally, (Haefner and Ray, 2019) proposes a complexity metric for IoT devices which is used to fine tune the anomaly detection algorithm for each device based on its complexity. Regardless of the specifics, the approaches above work at the network level, that is, they fail to detect anomaly in higher level user activities.

Other works investigate orthogonal aspects of IoT network security. IoT device fingerprinting (Meidan et al., 2017; Ortiz et al., 2019; Bezawada et al., 2018; Msadek et al., 2019; Miettinen et al., 2017; Thangavelu et al., 2019; Miettinen et al., 2017) uses various machine learning classifiers to generate unique network behavioral patterns of IoT devices. However, these works do not focus on user activity identification. (Ren et al., 2019) is a comprehensive study of privacy in IoT devices. (Acar et al., 2018) looks at privacy leakage from network traffic and suggests mitigation techniques such as traffic shaping.

Closer to our goal, (Wang et al., 2020) identifies the voice commands given to Amazon Echo and Google home using deep learning. This work focuses on a specific class of user activity and device. In (Apthorpe et al., 2017), the authors identify activities of IoT devices using Random Forest and K-Nearest Neighbor (KNN) classifiers. However, the amount of traffic data collected is small and further experimentation with large datasets is required to make conclusive statements.

3 Our Approach

3.1 Threat Model

In this work, we focus on an attacker who acquires the credentials of the legitimate user and controls the IoT devices in unintended ways. This includes activating/deactivating a device, performing various operations, and configuring it in a manner that compromises user security/privacy. Conventional anomaly detection based on analyzing network flows may be unable to distinguish between normal and anomalous activities. Our work aims to classify user activities based on network traffic and identify if the activity pattern of the user has changed. We make two assumption based on previous work – 1. IoT devices can be distinguished from conventional computing devices (e.g. - laptop) and 2. IoT devices can be fingerprinted to identify them.

3.2 Experimental Setup

Conventional network anomaly detection techniques focus on distinguishing normal and anomalous flows.

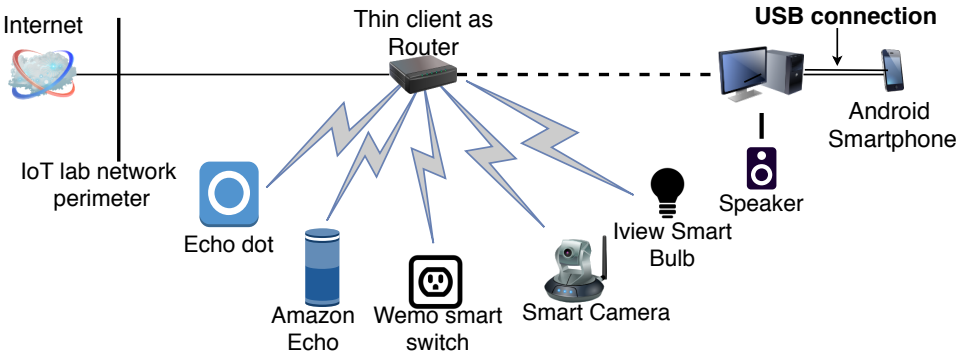


Figure 1: Setup Diagram

Our work, in contrast, focuses on distinguishing between normal and anomalous activities performed by the user. We use machine learning for activity classification and anomalous activity detection. *Activity* is an action being performed by the user on an IoT device (e.g., starting a video stream from a camera). Activity classification and anomalous activity detection setup involves the following pre-processing steps.

Packet Capture: This step involves capturing the network packets of the device and storing them in a trace.
Activity Log Generation: This step involves generating a log of the timestamps of when a particular activity has been started.
Packet Labeling: This step involves labeling packets with activity. Packets generated between the interval t and $t+\delta$ are assigned to the activity that started at t. Note that δ depends on the specific activity.

Packet Capture We are interested in annotating packets with user activities. At the time of experimentation, there were no available datasets, except for (Ren et al., 2019), which however does not allow experimenting with the values of δ. Since our goal is to annotate packets with associated activities, we provided our own infrastructure for packet capture. Most of the experiments were carried out at IoT laboratory at Colorado State University. Some of the experiments were repeated in home settings. The results we obtained at these two settings were comparable, helping to demonstrate the repeatability of results.

We used a Acer Veriton 2620G thin client with Open vSwitch as the router and wireless access point. For wired devices, an Aruba switch was connected to the thin client. The thin client was connected to the university network for Internet connection. An Android smartphone was connected to the desktop computer which was a HP workstation with Intel Xeon E3-1230 V2 @ 3.30GHz processor via USB connection. This desktop in turn was connected to the thin client. It also had a speaker connected to it. Figure 1 illustrates the packet capture setup.

Activity Log Generation We initiated activities using two methods:
Android applications on smart phone: We used `AndroidViewClient` (AndroidViewClient,) python library which makes use of android debugger bridge to control the smartphone from a python script. It can start and stop applications on the smartphone and can simulate a screen touch using XY coordinates.
Voice commands: For devices which use voice commands to take inputs from users, we converted text to speech using google translate API. The converted voice commands were then played on the speaker. We simulated power cycle using a WeMo Smart Plug (WemoInsight,) and connected the power cord of the IoT device to this plug. This plug can be controlled by smartphone.

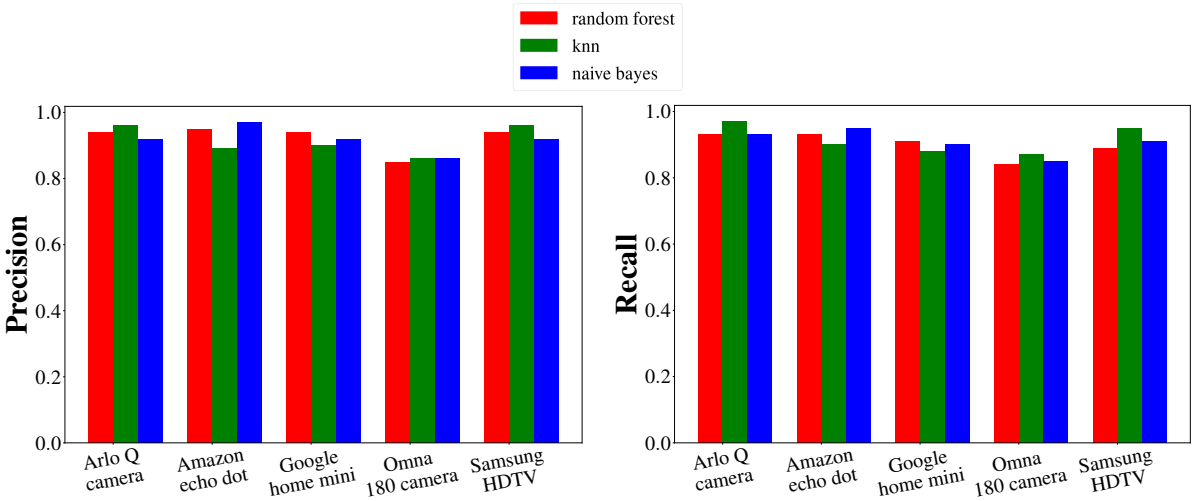
The packets were captured on the thin client using `tshark` and then sent to the Desktop, which generated the activity log. A python script was used to automate the packet capture and activity log generation. The dataset currently has captures for 9 IoT devices with total size of 19.8 GB.

Packet Labeling Packet captures and the activity logs are fed to the packet labeling module. For each activity with timestamp t, all packets from the device having a timestamp T, where $t \leq T \leq t+\delta$ are annotated with the activity id (δ is expressed in seconds and varies for different activities of a device).

The window size δ specifies the time it takes for an activity to finish from the time a user initiated it using smartphone or other interaction method. It is an important parameter and can only be approximated as it is not possible to accurately calculate it due to factors such as network delays, device response time

Device	Random Forest	kNN	Naive Bayes
Arlo Q camera	0.972	**0.974**	0.934
Amazon Echo dot	0.982	0.972	**0.984**
Google home mini	**0.946**	0.916	0.944
Omna camera	0.858	**0.876**	0.872
Samsung smart TV	0.958	**0.968**	0.932

Table 1: 5-fold cross validation average (accuracy)



(a) Precision

(b) Recall

Figure 2: Precision and Recall

variation, etc. Initially we calculated it manually for each activity. However, to make the process scalable, we assigned the same delta to activities of similar nature. The accuracy trade-off of this was not significant enough to affect the results of classification.

3.3 Data Preprocessing and Feature Selection

Errors and retransmissions are removed from the captures to reduce noise. Packet information are extracted and fed, along with the activity log, to the packet labeling module which, based on the provided window size δ, labels the packet according to the activity with which they are associated.

We partition the network traffic in three categories: captures having incoming, outgoing, and both incoming and outgoing packets. From these three categories we extract characteristic and statistical features. Examples include time delay, incoming mean, incoming number of packets, outgoing 30th, 20th percentiles, in out ratio, kurtosis, variance, incoming 10th, 40th, 50th, 60th percentiles, outgoing mean, and outgoing skew.

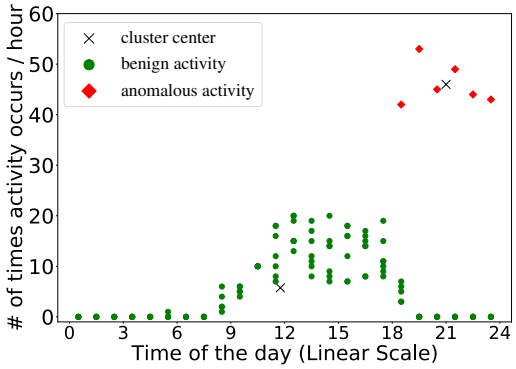
3.4 Devices generating limited traffic

We use deep packet inspection for devices that use unencrypted packets for communication and generate very small amount of traffic when an activity is performed. For such devices, machine learning methods for classification fail. We used this approach on Wemo smart plug (WemoInsight,), Iview bulb, Koogeek smart plug(KogeekPlug,). We extract payload from packets and calculate Levenshtein Distance from training data to perform classification. We get near perfect accuracy, however this approach suffers from scaling issues as each requires manual modeling.

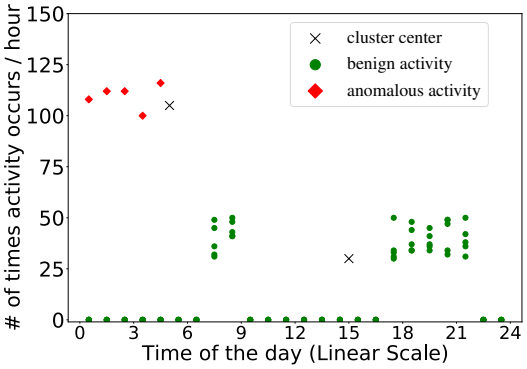
4 Results and Discussion

4.1 Activity Classification

We performed activity classification using random forest, kNN and Naive Bayes classifiers. We used `Python` package `scikit-learn` to build the model and test it on the captured data. We used 75/25 train-test split for the classification. The classifica-



(a) Benign vs malicious activity for video stream on Arlo Q camera



(b) Benign vs malicious activity for voice commands on echo dot

Figure 3: Anomalies in behavioral patterns

tion is one to many. We used the following evaluation metrics to compare the performance of the classifiers - $Accuracy = \frac{TP+TF}{TP+TF+FP+FN}$, $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$. Naive Bayes classifier performed the best overall among the three with the least error rate. The results are given in Table 1. Figure 2 details precision and recall of the Naive Bayes classifier.

4.2 Anomaly Detection

We synthetically generated benign and malicious user activity for a device for a period of 7 days using the same method used for performing activity classification. Both users were scripted according to realistic but different usage patterns (an attacker may attempt to mimic a user's behavior, but leave detecting this to future work). We then passed those packet captures though the activity classification module to get a sample usage pattern for some given activity.

We used K-means clustering algorithm to define cluster boundaries from the generated benign data. As parameters to the algorithm, we used frequency of occurrence of that particular activity per hour and time of the day. The use of clustering instead of binary classification allows us to perform anomaly detection in an unsupervised manner. The model is trained on the generated data. Figure 3a describes the video streaming activity of an Arlo Q camera. The x-axis corresponds to the time of the day. The y-axis denotes the number of times that activity is classified in that hour. The green dots show the normal occurrence of the activity and the red ones shows anomalous behavior. The 'X's indicate the two cluster centers. Similarly, Figure 3b shows benign vs malicious activity for echo dot. We used various voice commands to generate the activities. This figure shows a typical home scenario where the echo dot is used earlier in the day and in the evening. The malicious activity models a bad actor using a physical attack vectors—such as lasers (Sugawara et al., 2020) or ultrasonic audio frequencies (Zhang et al., 2017)—to inject voice commands from a distance. While these are difficult attack vectors to exploit, they are representative of physical attacks with no network footprint other than the unexpected device behavior itself. Note however that a similar attack could also be carried using an Alexa app with compromised credentials. In both proposed scenarios, a boundary between benign and malicious activity can easily be established.

5 Conclusion and Future Work

Home IoT devices often have poor security and are vulnerable to attacks. Our work focuses on detecting compromise; towards this end, we demonstrate how to identify the activities performed on the device by analyzing network traffic, and how to identify anomalies in user activities. Our future work involves extending this work to other types of devices, performing the anomaly detection experiments in a real-world setting, and doing a comparison of results.

REFERENCES

Acar, A., Fereidooni, H., Abera, T., Sikder, A. K., Miettinen, M., Aksu, H., Conti, M., Sadeghi, A.-R., and Uluagac, A. S. (2018). Peek-a-boo: I see your smart home activities, even encrypted! *ArXiv*, abs/1808.02741.

Alhaidari, S. and Zohdy, M. (2019). Hybrid learning approach of combining cluster-based partitioning and

hidden markov model for iot intrusion detection. In *ICISDM*.

Alrashdi, I., Alqazzaz, A., Aloufi, E., Alharthi, R., Zohdy, M., and Ming, H. (2019). Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning. In *CCWC*.

AndroidViewClient. **https://github.com/dtmilano/AndroidViewClient**.

Apthorpe, N., Reisman, D., Sundaresan, S., Narayanan, A., and Feamster, N. (2017). Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic.

Bezawada, B., Bachani, M., Peterson, J., Shirazi, H., Ray, I., and Ray, I. (2018). Behavioral fingerprinting of iot devices. In *ASHES*.

Bhatia, R., Benno, S., Esteban, J., Lakshman, T. V., and Grogan, J. (2019). Unsupervised machine learning for network-centric anomaly detection in iot. In *BigDAMA*.

Bowles, N. (2018). Thermostats, Locks and Lights: Digital Tools of Domestic Abuse. *The New York Times*.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3).

Goodin, D. (2019). 600,000 GPS trackers for people and pets are using 123456 as a password.

Haefner, K. and Ray, I. (2019). Complexiot: Behavior-based trust for iot networks. In *TPS-ISA*.

Hall, K. (2018). Hyperoptic's ZTE-made 1Gbps routers had hyper-hardcoded hyper-root hyper-password.

Hamza, A., Gharakheili, H. H., and Sivaraman, V. (2018). Combining mud policies with sdn for iot intrusion detection. In *IOT S&P*.

Hoang, D. H. and Duong Nguyen, H. (2019). Detecting anomalous network traffic in iot networks. In *ICACT*.

Jung, W., Zhao, H., Sun, M., and Zhou, G. (2020). Iot botnet detection via power consumption modeling. *Smart Health*, 15:100103.

KogeekPlug. **https://www.koogeek.com/p-p1-1.html**.

Meidan, Y., Bohadana, M., Shabtai, A., Guarnizo, J. D., Ochoa, M., Tippenhauer, N. O., and Elovici, Y. (2017). Profiliot: A machine learning approach for iot device identification based on network traffic analysis. In *SAC*.

Miettinen, M., Marchal, S., Hafeez, I., Asokan, N., Sadeghi, A., and Tarkoma, S. (2017). Iot sentinel: Automated device-type identification for security enforcement in iot. In *ICDCS*.

Mirsky, Y., Doitshman, T., Elovici, Y., and Shabtai, A. (2018). Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. In *NDSS*.

Msadek, N., Soua, R., and Engel, T. (2019). Iot device fingerprinting: Machine learning based encrypted traffic analysis. In *WCNC*.

Myridakis, D., Spathoulas, G., and Kakarountas, A. (2017). Supply current monitoring for anomaly detection on iot devices. In *PCI*.

Ortiz, J., Crawford, C., and Le, F. (2019). Devicemien: Network device behavior modeling for identifying unknown iot devices. In *IoTDI*.

Osborne, C. (2020). Smart locks opened with nothing more than a MAC address.

Pacheco, J., Benitez, V., and Félix, L. (2019). Anomaly behavior analysis for iot network nodes. In *ICFNDS*.

Ren, J., Dubois, D. J., Choffnes, D., Mandalari, A. M., Kolcun, R., and Haddadi, H. (2019). Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach. In *IMC*.

Reynolds, J. (2013). Dahua dvr authentication bypass - cve-2013-6117.

Sugawara, T., Cyr, B., Rampazzi, S., Genkin, D., and Fu, K. (2020). Light commands: Laser-based audio injection attacks on voice-controllable systems. In *USENIX Security*.

Thangavelu, V., Divakaran, D. M., Sairam, R., Bhunia, S. S., and Gurusamy, M. (2019). Deft: A distributed iot fingerprinting technique. *IEEE Internet of Things Journal*, 6(1):940–952.

Wang, C., Kennedy, S., Li, H., Hudson, K., Atluri, G., Wei, X., Sun, W., and Wang, B. (2020). Fingerprinting encrypted voice traffic on smart speakers with deep learning. In *WiSec*.

WemoInsight. **https://www.belkin.com/us/support-article?articleNum=42290**.

Zhang, G., Yan, C., Ji, X., Zhang, T., Zhang, T., and Xu, W. (2017). Dolphinattack: Inaudible voice commands. In *CCS*.