ComplexIoT: Behavior-Based Trust For IoT Networks

1st Kyle Haefner
Department of Computer Science
Colorado State University
Fort Collins, Colorado USA
kyle.haefner@colostate.edu

2nd Indrakshi Ray
Department of Computer Science
Colorado State University
Fort Collins, CO USA
indrakshi.ray@colostate.edu

***Abstract*—This work takes a novel approach to classifying the behavior of devices by exploiting the single-purpose nature of IoT devices and analyzing the complexity and variance of their network traffic. We develop a formalized measurement of complexity for IoT devices, and use this measurement to precisely tune an anomaly detection algorithm for each device. We postulate that IoT devices with low complexity lead to a high confidence in their behavioral model and have a correspondingly more precise decision boundary on their predicted behavior. Conversely, complex general purpose devices have lower confidence and a more generalized decision boundary. We show that there is a positive correlation to our complexity measure and the number of outliers found by an anomaly detection algorithm.**

By tuning this decision boundary based on device complexity we are able to build a behavioral framework for each device that reduces false positive outliers. Finally, we propose an architecture that can use this tuned behavioral model to rank each flow on the network and calculate a trust score ranking of all traffic to and from a device which allows the network to *autonomously* make access control decisions on a per-flow basis.

***Index Terms*—Internet-of-Things, Machine Learning, Anomaly Detection, Software Defined Networking, Access Control**

I. Introduction

Unlike networks of the past, made up of a small number of general purpose machines, IoT networks will increasingly be made up of a large number of specialized devices designed to do a single task. The single purpose and often constrained nature of these devices make them harder to intrinsically secure, but easier to extrinsically analyze. A single temperature sensor, for example, will not be able to run an anti-malware application, but does have a simple and predictable network traffic footprint.

This work exploits the single purpose nature and predictable network fingerprint of IoT devices to autonomously derive several measures of complexity based entirely on their network traffic. This allows not only for the evaluation of IoT devices based on their complexity, but also enables each IoT device to be accurately modeled based on its behavior on the network.

This behavioral model is then used to calculate a trust ranking for each and every network flow. Flows that have been deemed of low trust can be logged, rate-limited or blocked automatically by the network, conversely flows of high trust can be prioritized.

A behavior based flow routing model can be used as the first line of defence; to slow or prevent botnets and DDoS attacks at their source by detecting anomalous traffic at the granularity of individual flows from specific devices. Proper implementation would allow the network to selectively isolate and block malicious flows, leaving devices continuing to perform their primary function.

A reference network architecture for behavioral based flow routing is included in section VI.

Research Contributions

- We formalize the complexity of the behavior of devices based on their network traffic.
- We also propose metrics for measuring such complexity.
- We hypothesize that devices with smaller complexity values will show less of an aberration in its behavior compared with those of higher complexity values. Our results justify this. Thus, we tune the outlier threshold for the anomaly detection algorithms in accordance with device complexity.

To the best of our knowledge, this is the first work that attempts to tune anomaly detection algorithms using complexity of IoT devices.

This work is organized as follows, in section II we review why IoT security is a problem and current research on how to analyze and secure devices. In section III we describe the lab setup and what data was collected. In section IV we develop methods for measuring complexity and how devices are classified into discrete groups. In section V we develop a method for modeling learned behavior of IoT devices and propose a confidence model for each device. Finally in section VIII we conclude with a proposed access control architecture to selectively route flows based on the derived confidence for each device.

II. Background

A. Complexity and Predictability

The correlation between complexity and predictability is an intuitive and foundational principle of probability theory that has roots dating back to at least Aristotle [1]. The challenge is to determine a statistically significant way of measuring the complexity of a system to form meaningful confidence in a predictive model of that system.

Formalized measurement of complexity as applied in a computer science context is probably most often associated with the works of Andrey Nikolaevich Kolmogorov, who defined the complexity of an object as the shortest computer program to produce the object as an output [2]. This simple notion arises again in the work of Jorma Rissanen whose work on the minimum description length principal that establishes that the best model for a set of data is one that leads to the best compression of the data [3].

In the paper Predictability, Complexity, and Learning authors Bialek et al. establish a formal result that predictive information provides a general measure of complexity [4]. In this work we propose that the relationship between predictive information and complexity is commutative, i.e. not only does predictive information lead to a measure of complexity, but that complexity provides a general measure of predictive information.

In machine learning this relationship leads to the logical notion that the less complex the model the more accurately it can be modeled. Specifically, this work builds an anomaly based behavioral model, where the device's complexity directly affects the decision boundary that differentiates between inliers and outliers.

B. IoT and Security

The Internet of Things term was first coined by Kevin Ashton [5] in 1999 when describing RFID use in supply chain management. This definition has ballooned to become a catch-all phrase for any device that connects to or interacts with the Internet. Examples of these devices include medical sensors that monitor health metrics, home automation devices, traffic monitoring, and scientific research sensors. Some of these devices will be designed to last for a few weeks and be disposed of, like a sensor on food packaging. Others will be embedded into infrastructure that will be around for decades such as sensors embedded into roads. Some devices will need to run on batteries for years; have limited processing and storage capabilities and spend the majority of the time in sleep mode. Others will have powerful processors, a constant power source and high bandwidth connection to the network. This diversity in function, capability, and life-span is at the core of what makes securing these devices so challenging.

Security baselines [6], [7] and strong endpoint security in international standards [8] are steps in the right direction, but there will always be insecure devices; either because they were manufactured that way or did not receive software patches. This is highlighted in the large corpus of research [9]–[13] that highlights how and why vulnerable IoT devices are prone to security attacks.

C. IoT and the Identity Crises

The fundamental challenge in securing IoT devices and enabling networks to apply network-based access control is identifying devices and their corresponding behavior on the network. At a highly abstract level, NIST, the National Institute of Standards and Technology, has defined 5 IoT primitives [14]

1) **Sensor** - a physical device that outputs data relevant to its environment.
2) **Aggregator** - a physical or virtual device that collects and/or transforms data.
3) **Communication Channel** - a link or medium that transfers data between nodes or devices.
4) **External Utility** - an abstraction for a data processor. An example of this is a cloud based AI system.
5) **Decision Trigger** - virtual device that outputs the final results or computations.

The NIST primitives define devices based on broad classifications of behavior, however these primitives are more geared toward policy decisions and are too abstract for it to be useful in an access control model.

A recent IETF proposal standard call Manufacturer Usage Description (MUD) specifies how manufacturers codify expected behavior of devices in terms of access lists and policies [15]. If this is fully adopted by manufacturers *and* secured by a strong Public Key Infrastructure (PKI) based device identity, then this approach represents a reasonable baseline of trust to bootstrap automatic network-based access control, however, the network will still need to verify the device's behavior on an ongoing basis due to the fact that a fully compromised device will still be able to falsely submit a MUD URL to the network.

D. Probabilistic Identity and Fingerprinting

Beyond IoT primitives and relatively static access control lists, several works aim to derive device identity-based on network traffic. Loepz-Martin et al. [16] build a network traffic classifier (NTC) using a recurrent neural network (RNN) and apply it to labeled IoT traffic. The goal of this is to identify the types of traffic and services exhibited by an IoT device as a step toward identifying the device.

Miettinen et al. [17] have developed a method, called IoT Sentinel, that uses machine learning to designate a device type on the network, referred to by the authors as a device fingerprint. Using the random forest algorithm and 23 network features they were able to identify device types on the network based on the device's traffic. The 23 features are based on layer two, three and four of the OSI networking stack. Expecting that the body of the packet will be encrypted, all the features the authors employed are based on unencrypted parts of the traffic like IP headers information.

The authors claim that using the random forest algorithm, they were able to identify the 27 device types in the study with an overall accuracy of 81.5%. They noted that for 17 of the devices they were able to identify the device with a

95% accuracy. For the other ten devices they achieved only a 50% accuracy of identification. These ten devices are largely different devices from the same manufacturer. The authors explain that their classifier is good at discriminating between devices that have different hardware or firmware, but if the device is made by the same manufacturer and has the same firmware their classifier does not accurately fingerprint the device. They claim that this is not important in that the two very similar devices are likely to have the same vulnerabilities and is inconsequential.

While IoT Sentinel presents some interesting solutions, it has a couple of major drawbacks. First and foremost is that they use a supervised learning algorithm that must be individually trained on each device type. Not only that but it must be re-trained if the device firmware changes. This re-training makes the solution difficult to scale to the myriad of heterogeneous devices and requires an extensive on-line database of trained classifiers. This makes their solution reliant on the accuracy of, not just one but, two public databases, the CVE database and a database of trained classifiers. Second, by only analyzing the device during setup, they are missing the vast majority of the device behavior on the network. If the device is compromised after being installed, this solution is unlikely to recognize it. Third, since the author's classifier is unable to distinguish very similar devices, this solution has a high probability of false positives, which makes it unsuitable for critical high-availability medical devices.

Bezawada et al. [18] build on the work done in [17] by using a machine learning approach to broadly identify the device and place it in a predefined category, such as a light bulb. According to the authors, even devices from different manufacturers can be placed into general categories such as two separate light bulbs can be identified and placed into a lighting category.

This solution of fingerprinting devices suffers from a similar problem as IoT Sentinel in that it uses a supervised approach to fingerprinting and categorization. This is challenging in that it requires labeled data for each device. Additionally, the authors assume that they will be able to detect a distinct command and response structure in the data from any particular device. This coupled with the relatively low sample size of the devices they tested makes this approach potentially ineffectual for more complex devices where aspects like encryption may interfere with the ability to detect the command and response structure.

Moving beyond the supervised limitations authors Marchal et al. [19] developed a technique to generically identify devices based on the periodicity of their network communication. The authors employ signal processing techniques to analyze devices' background periodic traffic with the goal of placing devices into derived identity groups. Using discrete Fast Fourier Transform the authors then generate 33 features that they then train a k-nearest neighbors (kNN) model. The authors then use the model to place devices into one of 23 groups based on the clusters found by the kNN.

This work creates 23 generic device identities by analyzing 33 commercial IoT devices. This model generates a unique derived identity for nearly 70% of the devices, i.e. only slightly better than a derived identity per device. For example, the largest derived identity group contains only five devices, all of which are from D-Link. Extrapolating this to a wider set of devices across a wider set of manufacturers, this work would produce a very large number of derived identities. This large quantity of derived identities would not do much to simplify network and security policies attempting to use them as input for decisions.

E. IoT Behavior

The following works use various means of autonomous and unsupervised machine learning approaches to identifying devices and device behavior. This has advantages over statically defined access control lists and firewall rules.

HoMonit [20] develops a anomaly detection engine for analyzing the behavior of smart devices using a model based on software applications (SmartApps) made by third-party developers on the SmartThings platform. The authors use side channel information sniffed from encrypted wireless traffic to compare the inferred behavior of a device and the expected state machine model based on the SmartApp.

IoT-Keeper [21] is an edge based IoT anomaly based access control system that uses correlation-based feature selection to determine which features do not contribute to the anomaly detection. AuDI [22] implemented an autonomous device-type identification that uses the periodicity of device communications resulting in abstract device categories that could be used to enforce access control policies. DioT [23] extends the AuDI classification model to create a federated approach by aggregating device anomaly detection profiles.

Ren et al. use a privacy focused approach to enumerating and analyzing IoT behavior [24]. The authors set up two labs, one in the US and one in the UK. The labs consisted of a total of 81 devices with 26 device being common between the two labs. The authors then proceeded to analyze the traffic for each lab looking at behavior of an IoT device during boot and when it was actively being controlled and/or interacted with.

Ortiz et al. set up a probabilistic framework to monitor device behavior using an LSTM (Long Term Short Term Memory) neural network, to learn from inherent sequencing of TCP flows to automatically learn features from device traffic with the intent of categorizing devices and distinguishing between IoT devices and Non-IoT devices [25]. The authors are able to identify previously known devices after only 18 TCP-flow samples and categorize devices into two classes IoT and Non-IoT.

Authors O'Connor et al. [26] use supervised machine learning to map high level semantic device behavior to 148 behavior types, such as heart beat, firmware downloading, configuration change, sending video, etc [26]. The authors build an abstraction model of client-server interactions using TCP/IP header information and a multi-class classifier to map these interactions into the known behavioral types. This research then provides an access control model based on the classified behavior, the device and context, where the context is to be provided by owner.

Our model does not try to place devices into artificially or arbitrarily devised classes, nor try to categorize devices or their behavior based on their function. It does not rely on any contrived labels of behavior, user interaction, or device type. Instead everything is learned autonomously from flow data features shown in Table I, with the aim to measure devices based on their network complexity and use that to refine our model of their behavior.

Our model can be used to determine a representative set of flows that define the behavior of a device and these flows can be directly loaded into flow tables of Openflow enabled switches. We believe that this will scale to the broad spectrum of devices and adapt to any new configurations of devices in the future.

Take, for example, a refrigerator that is also an Android tablet, the methodologies in the related works above would struggle to characterize such a device. Our method does not try to recognize this device as either a refrigerator or a tablet, it does not try to guess at the service or characterize the device's application layer data. Our model does not rely on learning specific human interactions with the refrigerator, nor determining if those interactions are anomalous. Our model only relies on how complex the refrigerator appears on the network and how much it stays within our learned boundary of behavior.

III. Data Format and Collection

Data was collected from a real residential network with approximately 25 devices over the course of 37 days. These devices range from general computing devices like laptops and smartphones, IoT hubs with several IoT Devices using Zigbee or Zwave behind them, to single-purpose devices such as light bulbs and temperature sensors. Data was collected by a central MicroTik router shown in (Figure 1) that sends Netflow/IPX data to nprobe running on a Raspberry Pi. Flows were stored in a MariaDB relational database. Table I shows the features of the data collected.

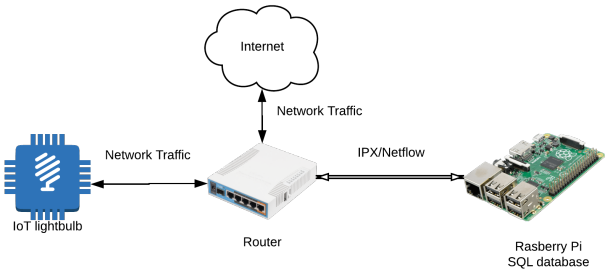
Flows were aggregated with a maximum of 30 minutes per flow. Inactive flow timeout is 15 seconds. If the devices have not exchanged traffic in 15 seconds the flow is completed and recorded. For training and test data sets the data is filtered by an individual IP address. The test environment is configured such that the devices always receive the same IPv4 address. Flows are sorted by time-stamp eliminating time as a variance factor.

Definition 1. Network Flow: A sequence of packets where all the packets in the flow have the same tuple: source address, destination address, source port, destination port and protocol.

Table I
Data Features

Feature	Description
IPV4_SRC_ADDR	IPv4 Source Address
IPV4_DST_ADDR	IPv4 Destination Address
IN_PKTS	Incoming flow packets (Device->Destination)
IN_BYTES	Incoming flow bytes (Device->Remote)
OUT_PKTS	Outgoing flow packets (Remote->Device)
OUT_BYTES	Outgoing flow bytes (Remote->Device)
L4_SRC_PORT	IPv4 Source Port
L4-DST_PORT	IPv4 Destination Port
PROTOCOL	IP Protocol Identifier

Figure 1. Data Collection Architecture



IV. Device Complexity Classification

Device complexity is an aggregate measurement of a device's IP connections, d_{ip}, and how much its traffic varies over time d_v.

A. Device IP Complexity

This research examines how devices form connections. One aspect of this analysis is to study *IP spread* and *IP depth*. IP spread is the number of unique source and destination IP addresses that interact with the device. IP depth is the number of IP addresses that belong to the same higher level octets.

Definition 2. IP Tree: An IP tree is a unique first order octet. This comprises the root of the tree.

Definition 3. IP Branch: A second, or third order octet that has one or more fourth order octets (Leaf) under it.

Definition 4. IP Leaf: An IP leaf is a unique fourth order octet.

Definition 5. IP Spread: Sum of total unique IP addresses that interact with a device

Definition 6. IP Depth: The number of addresses that interact with a device and belong to the same higher order octets i.e they have common first, second or third order octets.

a) Device IP Spread:

$$IP_{Spread} = \sum IP_{trees} \tag{1}$$

b) Device IP Depth:

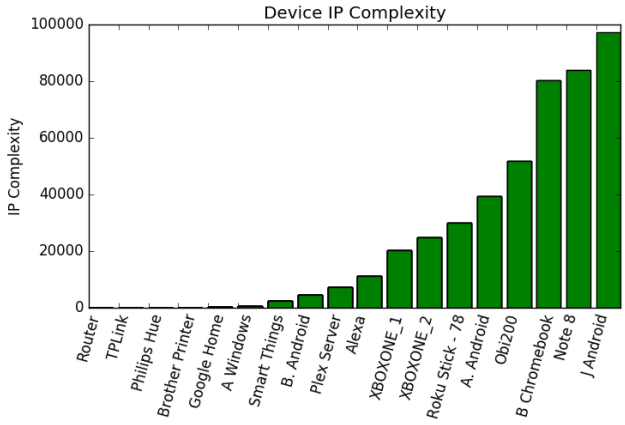
$$IP_{Depth} = \frac{\sum IP_{leaf}}{\sum IP_{branch}} \tag{2}$$

c) Device IP Complexity:

$$d_{ip} = \frac{IP_{Spread}}{IP_{Depth}} \tag{3}$$

To calculate IP spread and depth we build unordered trees of each IP address where the first order octet is the root and lower octets are children. Then we can calculate how many trees, branches and leaf nodes each IoT device contacts. A large number of IP trees with few branches indicates a large IP spread. A small number of IP trees that have many branches and leaves indicates a large IP depth. IP depth/spread is used as one measure of a device's complexity. Devices belonging to a single ecosystem such as Google Home have a small number of broad trees as they connect to mostly Google's networks dedicated to these types of devices. Other devices such as laptops and smart phones have a larger IP spread with each IP having fewer branches and leaves. Figure 2 shows the total IP complexity of each device.

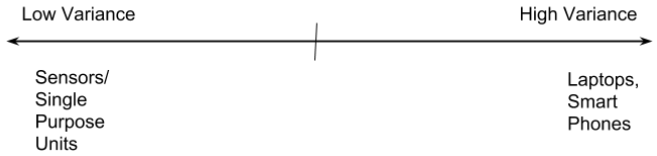
Figure 2. IP Device Complexity



B. Device Variance

The variance metric comes from the simple notion that devices on a network present different variances based on what they do on the network. Here we employ the explained variance score (standard deviation squared) computed over the flow history of the device. The explained variance score gives us a normalized measure of the dispersion of the data between a training subset and a test subset. Explained variance requires an even split of samples for training and test. Figure 3 shows the spectrum of variance expected per device type.

Figure 3. Spectrum of Network Variance



1) Explained Variance of Network Flows: This is the explained variance score d_v of the device averaged over a past historic window. Initially this research used a time-based window of the past thirty days. Future work will examine if windows should be based on a minimum number of flows, time or both. Figure 4 shows the measured variance of devices on the network.

$$d_v = \sum_{n=0}^{n=Current} \frac{d_{vn}}{n} \tag{4}$$

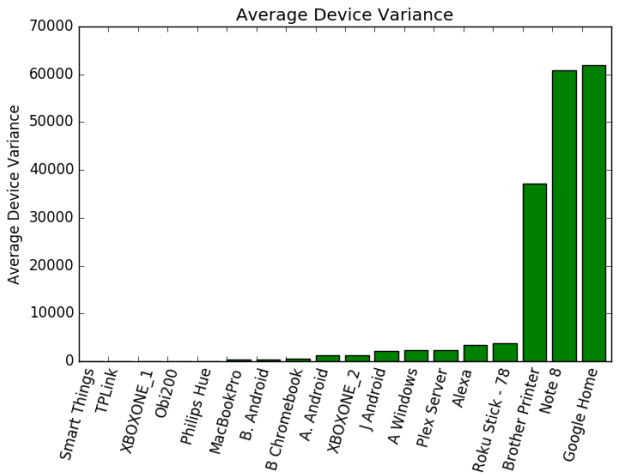
$$d_{vn}(f, \hat{f}) = 1 - \frac{Var(f - \hat{f})}{Var(f)} \tag{5}$$

Variance σ^2 is the standard deviation squared

$$Var(f) = \sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1} \tag{6}$$

where: $\hat{f} = win_{n-1}$ flows
$f = win_n$ flows

Figure 4. Average Device Network Variance



C. Aggregate Complexity

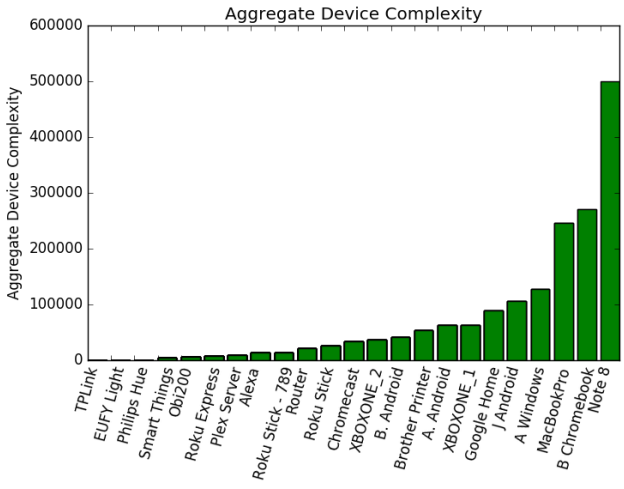
Overall device complexity is the sum of the average device variance and the average device IP complexity shown in Figure 5.

a) Aggregate Device Complexity:

$$AD_C = w_{ip} d_{ip} + w_v d_v \tag{7}$$

where: d_{ip} = device IP complexity
d_v = device variance
w_{ip} = weight of IP complexity
w_v = weight of variance
$initial\ weights\ w_v, w_{ip} = 1$

Figure 5. Aggregate Device Complexity



V. Behavior

The word behavior is an abstract word especially as applied to a network device. However, given the complex interactions that IoT devices have with the physical world, behavior adequately represents the dynamic and changing network footprint exhibited by these devices. The sensing and actuating response of IoT devices that bridges the network and physical world requires new methods of defining what is normal and what is abnormal. IoT devices, even the same make and model from the same manufacturer will exhibit slightly different behavior based on how they interact with the human inhabitants, each other and the environment. Two very similar devices, that have different apps installed act very differently. This variance in behavior requires that the model is tailored to these specific and individual devices.

We begin by defining IoT device behavior based on the past history of network interactions of the device, bounded by the most extreme of these interactions. To model the degree of normality and extremity of behavior we turn to classic outlier detection algorithms, adding what we believe to be a key contribution of this research, we tune the hyper-parameter of the outlier detection algorithm to the specific device based on the measure of complexity as defined in the previous section.

This method has the direct affect of making the decision boundary of the trained model a more precise fit for simple devices and more generalized for complex devices. This allows the detection algorithm to be more strict in identifying outliers for simple devices and more lax for complex devices. This enhances the model, enabling it to adaptively prioritize new extreme behavior for simple devices and reduce false positives for complex devices.

A. Anomaly Detection

To derive a behavior for the device we take the isolation forest anomaly algorithm as defined by Liu et al [27]. Isolation forest is a tree ensemble method that builds decision trees based on a random partition of the feature space. The tree's path length is inversely proportional to the abnormality of the observation, ie. the shorter the path to divide the observation space from other observations the more anomalous the observation is.

Isolation forest is somewhat unique among outlier detection methods in that it explicitly measures the abnormality of the observations rather than attempting to profile the normal data points.

a) Isolation Forest:

$$s(x,n) = 2^{-}\frac{E(h(x))}{c(n)} \tag{8}$$

where: $h(x)$ = path length of observation
$c(n)$ = average path of unsuccessful search.

Isolation forest also lends itself to good performance on low-end hardware. The total number of nodes of an isolation tree is $2n-1$. Thus the tree and its required memory grows linearly with n. Additionally, because the search is for the shortest path, the algorithm does not need to spend processing time or memory searching observations with the longer path trees, as these are explicitly in-liers.

B. Behavior Tuning Using Device Complexity

The contamination parameter specifies the percentage of observations that are believed to be outliers for training. To properly tune the isolation forest we use the calculated complexity value AD_C for each device to set the contamination parameter. For low complex devices this is set to nearly zero and high complex devices will be closer to 0.5. This sets a narrow bound on the decision function for devices with low complexity and a broader bound on the decision function for highly complex more general purpose devices.

C. Behavioral Boundary

Definition 7. Device Learning Period: A window of past network traffic from the device is called the device learning period D_{LP}. The D_{LP} should capture the device's initial joining to the network and a period of normal use.

The behavioral boundary for a device is calculated from the device learning period D_{LP}. The behavior boundary consists of the cardinal set of all unique flows from a device with a path length less than the average path length. Average path length is the average path length for the dataset that is comprised of all the flows from each device. We call this set of flows *significant flows*.

Definition 8. Significant Flow: A flow that is marked as an outlier by the isolation forest algorithm during the learning period, meaning that it has a shorter than average path length.

Definition 9. Device Behavioral Boundary: The set of all unique significant flows, their path length, and the average path length for the device during the learning period.

D. Device Confidence

Device confidence gives us a measure of trust that we can have in a device that is based on its historical behavior. Device confidence is calculated over a validation period D_{VP} that follows the learning period D_{LP}. During the D_{VP} any new novel flows, i.e. flows that are new and considered outliers are added to the behavior model. The rate of change in added flows, from the end of the D_{LP} to the end of the D_{VP} is the device confidence score.

Definition 10. Device Confidence: The rate of change in the cardinal observations for a device over the validation period.

$$D_C = \frac{1}{\sum^{V_{VP}} f'} \tag{9}$$

where: D_C = device confidence
f' = new unique flows after initial training
D_{VP} = device validation period

E. Flow Trust Score

The goal of this research is to arrive a flow trust score F_T for each flow on the residential home network.

Definition 11. Flow Trust Score: Score applied to each flow of each device on the network. It is calculated on the current anomaly fit and score of the current flow multiplied by the device confidence score D_C.

The flow trust score is calculated below:

$$F_T = \frac{A(fit)D_C}{fa_c fa_a} \tag{10}$$

$$A(fit) = \begin{cases} 1 & A_c \;\&\&\; A_a > 0 \\ -1 & A_c \parallel A_a < 0 \end{cases} \tag{11}$$

where: A_c = isolation fit of connection function
A_a = isolation fit of aggregation function
fa_c = measured flow anomaly connection score
fa_a = measured flow anomaly aggregation score
D_C = Normalized device confidence in equation 9

1) Flow Anomaly: There are two flow anomaly scores that need to be considered, flow connection anomaly fa_c that is the anomaly of the connection calculated across the tuple of IP headers, and flow aggregation anomaly, the anomaly calculated over the aggregate of the flow fa_a, ie total bytes, total packets.

Example of high trust flow score:

$F_T = \frac{1*0.85}{0.1*0.01} = 850$

where: $A(fit)$ = 1 (normal observation)
D_C = 0.85 (high deice confidence
fa_c = 0.1 (normal observation)
fa_a = 0.01 (normal observation)

Example of a low trust flow score:

$F_T = \frac{-1*0.195}{0.95*0.89} = -0.230$

where: $A(fit)$ = -1 (abnormal observation)
D_C = 0.195 (low device confidence
fa_c = 0.95 (abnormal observation)
fa_a = 0.89 (abnormal observation)

VI. Architecture

The ComplexIoT architecture is based on a centralized model where there is a single device that acts as a router, gateway and access point. This central device is an off the shelf low power x86 computer that is running Kali Linux. It has 4GB of RAM, an Intel processor, and a Broadcom wireless chipset.

Software defined networking (SDN) decouples the control and the data plane in routers and switches. This opens the network to new services, features, and a level of dynamism that was previously not possible. This work leverages the programmability of the network to dynamically allow, block, rate-limit and route traffic based on the confidence score of the flow.

Openflow is an open specification for software defined networking led by the Open Networking Foundation (ONF) [28]. In June, 2012 version 1.01 of the open flow specification was released. There are many hardware and software switches that support OpenFlow.

The prototype architecture developed for this work uses the OpenFlow reference soft switch called OpenVSwitch [29]. OpenVSwitch supports OpenFlow versions 1.0-1.5.

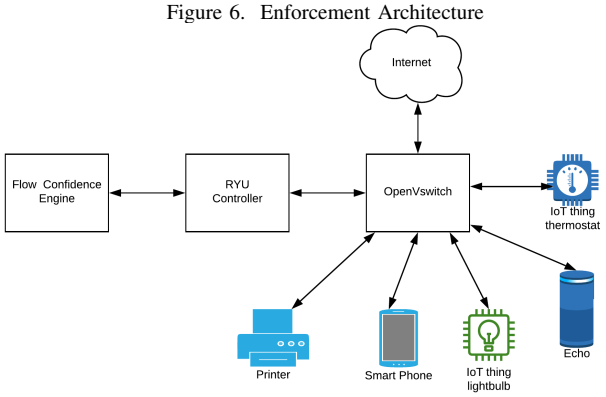
RYU is a software defined network controller that implements OpenFlow. In this prototype we use RYU to setup and control OpenVswitch [30]

The flow collector consists of a Raspberry Pi running a netflow collection software called nprobe. Nprobe stores the flows into a MariaDB database.

In figure 6 the flow trust engine analyzes past device flows and calculates device complexity, behavioral boundary, device confidence, and flow scores. It sends the flow score information to RYU, the SDN controller. RYU uses the received flow score to add,remove or modify flows rules based on the confidence score. RYU then pushes flow rules the openVswitch flow tables. Device confidence scores dictate to RYU if new flows should be allowed, rate limited or dropped.

The architecture in Figure 6 allows the network to make extremely granular flow decisions on every flow in the network, including inbound/outbound traffic to the Internet and intra-network device traffic. Based on the behavioral boundary there is no need to isolate an entire device, just the flow that is not trusted.

Figure 6. Enforcement Architecture



A. Enforcement

Because training of the confidence model is based on aggregate flow data there must be two stages for enforcement. These two stages are the connection stage and the aggregation stage.

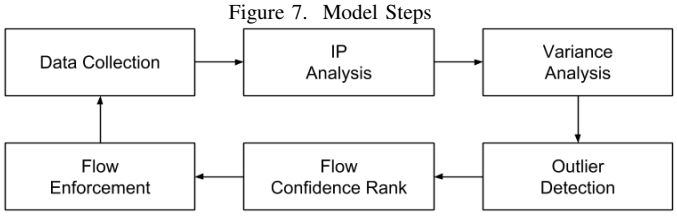
1) Connection Stage: The connection stage examines features of the confidence model that are known at connection time. The connection stage is only run once at flow connection setup. The connection features include IP header attributes such as IP source, IP destination, port, and protocol. If the model detects an outlier based on the connection features it will use the current device confidence scores and the outlier degree to the flow to calculate the flow trust score.

2) Aggregation Stage: The aggregation stage gathers flow aggregate information that includes incoming flow bytes, incoming flow packets, outgoing flow bytes, and outgoing flow packets. The aggregation stage is run continuously while the flow is active. The aggregation stage compares the trained model to the current flow. If the current flow is detected as an aggregate outlier it will take the current device confidence score and the outlier degree and calculate a flow trust score.

B. Continuous Model Update

For the behavioral model to be up to date, continuous training over the device history data is needed. Because it is expected that the device histories will become very large, training and testing will be done across a a window of past data for each device. The model needs only to update the significant flows and the average path length for each device, negating the requirement to store the entire device history.

Figure 7. Model Steps



C. Bootstrapping

When a device is added to the network it begins in the learning period, and with a neutral device confidence $D_C = 1.0$. Additionally, until a sufficient behavioral record can be established all flows from the device are assumed to be neutral flow trust F_T. Other ways of establishing initial confidence, such as a device presenting a valid certificate and a query of a vulnerability database such as CVE is left to future work.

VII. Results

The preliminary goal of this research is to develop a method of measuring devices on a network based on their complexity and using this complexity measure to refine the isolation forest algorithm. Table II show the complexity, total flows and the percentage of flows that are outliers.

Table II
Device Behavior Boundary

Device	Total Flows	Significant Flows	Percent
TP-Link	8375	1	0.012%
Smart Things	10000	537	5.37%
Google Home	838784	1714	0.44%
Alexa	199489	1810	0.9%
MacBook Pro	1545311	2751	0.18%
Samsung Note	199767	1921	0.96%

As can be seen in table II only about 1% of flows need to be stored to capture all unique significant flows. The behavioral boundary is the set of unique significant flows, the path length of each significant flow, and the average path length of the forest for each device measured during the learning period.

Figure 8 shows the scatter plot of aggregate complexity AD_C vs outliers using the *untuned* isolation forest algorithm. Linear regression run on this data produces coefficient of determination (R^2) = 0.037. The smart things hub did not follow this trend having higher outlier flows than would be expected based on its complexity, this is likely due to the inherent nature of a hub with many devices and a single IP address. Figure 9 shows the trend with smart things Hub removed, this shows a much stronger positive correlation (R^2) = 0.626.

Figure 10 shows the same trend of complexity for un-tuned and tuned models of the device. The tuning of the algorithm reduces the correlation of complexity to outliers slightly (R^2) = 0.494, this is expected as the tuning actively dampens the number of outliers found proportional to the complexity of the device. The tuning also there is an overall reduction of outliers per device and shows a reduced slope (0.67) in the tuned trend-line. This reduced slope is the direct result of the decision boundary being more lax in determining outliers in devices that are more complex.

Figure 8. Un-Tuned: Complexity vs Outliers

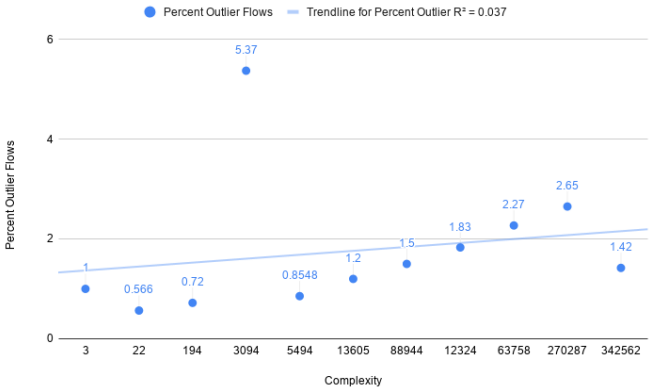


Figure 9. Un-Tuned: Smart Things Removed

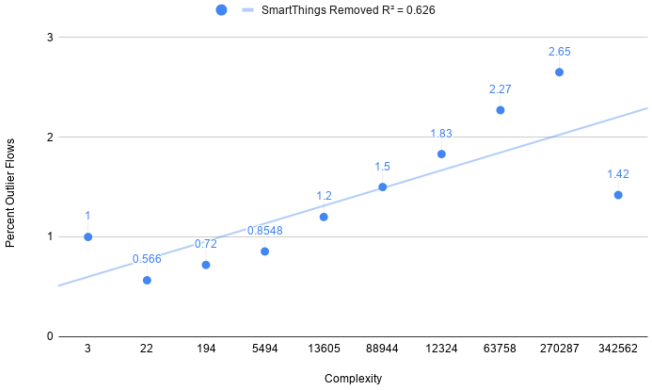


Figure 10. Un-Tuned Vs Tuned Isolation Forest

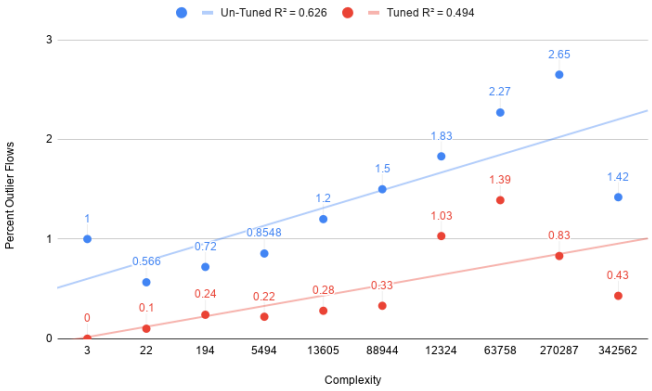


Table III
SAMPLE OF DEVICE CONFIDENCES

Device	Device Confidence
TP-Link	1.0
Philips Hue	0.952
Obi200	0.8548
Google Home	0.812
Roku	0.798
B Chromebook	0.765
Alexa	0.746
XBOXONE 1	0.698
EUFY Light	0.566
Smart Things	0.512
Note 8	0.311
MacBook Pro	0.182

Device confidence begins at 1.0, this represents a zero rate of change during the validation period. In table III the device with the highest confidence is the TP-Link switch, it exhibited no change in its significant flows from the end of the learning period to the end of the validation period. The Macbook Pro has the lowest device confidence as it showed the largest rate of change in significant flows over the validation period. This follows logically with the nature of a general purpose computer that is difficult to accurately model due to its high complexity.

VIII. Further Discussion and Future Work

The next steps in this research are to implement the enforcement architecture and analyze how it scales in terms of bandwidth and CPU resources. With multi-core arm-based CPU architectures and hardware based openflow implementations we will investigate the feasibility of building this architecture on low-power off-the shelf router hardware.

A. Ground Truth

With any anomaly detection, there must some ground truth, a method of measuring how well the algorithm is performing. Ground truth can be established in one of two ways; a known and established anomaly in the training data can be employed to verify that the algorithm can identify that example as an anomaly. The second way of establishing ground truth is to introduce an artificial example into the data. In this research, we will use the latter approach and inject artificial examples into the training data and evaluate the efficacy of the detection algorithms based on detecting those examples.

B. Home Vs Lab

This research will examine how two different use cases affect the analysis of the devices. The first is a home environment where users are living and interacting with the network and IoT Devices. The second is a lab environment where the devices are mostly idle and do not have user interaction. This research will compare the two environments and attempt to draw conclusions for a baseline behavior of IoT devices and how human interaction changes that baseline.

IX. Conclusions

In this work we propose a new method for classifying IoT devices based on the complexity and variance of their network

flows. We showed how this complexity can be used to tune the decision function of the Isolation Forest anomaly detection algorithm for each device.

Next, we showed how to model a device to establish a behavior boundary of normal flows. Using the rate of change in the number of unique significant flows we defined a confidence metric for each device.

We take the learned behavioral boundary for each device modified by the device confidence score develop a trust score for every flow from each device.

Finally we suggest a reference architecture that can be used to make per-flow access control decisions for each device.

X. Acknowledgements

This work was supported in part by grants from NSF under Award Number CNS 1650573 and CNS 1822118 and funding from CableLabs, AFRL, Furuno Electric Company, and SecureNok.

References

[1] Aristotle and J. Barnes, *Aristotle's Posterior Analytics*. Oxford: Clarendon Press, 1976.

[2] A. N. Kolmogorov, "On tables of random numbers," *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 369–376, 1963.

[3] J. Rissanen, *Stochastic complexity in statistical inquiry*. World Scientific, 1989.

[4] W. Bialek, I. Nemenman, and N. Tishby, "Predictability, complexity, and learning," *Neural computation*, vol. 13, no. 11, pp. 2409–2463, 2001.

[5] K. Ashton *et al.*, "That internet of things thing," *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.

[6] S. K. S. M. Fagan M, Megas K. (2019) Core cybersecurity feature baseline for securable iot devices.

[7] C.-C. W. Group, "The c2 consensus on iot device security baseline capabilities," Consumer Technology Association, Tech. Rep., 2019.

[8] "Open connectivity foundation (ocf) specification part 2: Security specification," Open Connectivity Foundation, Standard, 2019.

[9] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic," *arXiv preprint arXiv:1705.06805*, 2017.

[10] C. Wilson, T. Hargreaves, and R. Hauxwell-Baldwin, "Benefits and risks of smart home technologies," *Energy Policy*, vol. 103, pp. 72–83, 2017.

[11] V. Sachidananda, S. Siboni, A. Shabtai, J. Toh, S. Bhairav, and Y. Elovici, "Let the cat out of the bag: A holistic approach towards security analysis of the internet of things," in *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security*. ACM, 2017, pp. 3–10.

[12] K. Zhao and L. Ge, "A survey on the internet of things security," in *2013 Ninth international conference on computational intelligence and security*. IEEE, 2013, pp. 663–667.

[13] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in internet-of-things," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1250–1258, 2017.

[14] J. Voas, "Networks of things," *NIST Special Publication*, vol. 800, no. 183, pp. 800–183, 2016.

[15] D. R. Lear, E. and D. Romascanu. (2019) Manufacturer usage description specification.ietfrfc8520. [Online]. Available: https://tools.ietf.org/html/rfc8520

[16] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*, vol. 5, pp. 18 042–18 050, 2017.

[17] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "Iot sentinel: Automated device-type identification for security enforcement in iot," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 2177–2184.

[18] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Behavioral fingerprinting of iot devices," in *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security*. ACM, 2018, pp. 41–50.

[19] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, "Audi: Toward autonomous iot device-type identification using periodic communication," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1402–1412, 2019.

[20] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu, "Homonit: Monitoring smart home apps from encrypted traffic," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 1074–1088.

[21] I. Hafeez, M. Antikainen, A. Y. Ding, and S. Tarkoma, "Iotkeeper: Securing iot communications in edge networks," *arXiv preprint arXiv:1810.08415*, 2018.

[22] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, "Audi: Toward autonomous iot device-type identification using periodic communication," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1402–1412, 2019.

[23] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for iot," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 756–767.

[24] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi, "Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach," in *Proceedings of the Internet Measurement Conference*. ACM, 2019, pp. 267–279.

[25] J. Ortiz, C. Crawford, and F. Le, "Devicemien: network device behavior modeling for identifying unknown iot devices," in *Proceedings of the International Conference on Internet of Things Design and Implementation*. ACM, 2019, pp. 106–117.

[26] T. OConnor, R. Mohamed, M. Miettinen, W. Enck, B. Reaves, and A.-R. Sadeghi, "Homesnitch: behavior transparency and control for smart home iot devices," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, 2019, pp. 128–138.

[27] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 413–422.

[28] (2012) Openflow switch erata, open networking foundation, onf ts-001. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2013/07/openflow-spec-v1.0.1.pdf

[29] (2019) Production quality, multilayer open virtual switch. [Online]. Available: https://www.openvswitch.org

[30] (2019) Ryu sdn framework. [Online]. Available: https://osrg.github.io/ryu/