Towards a Unified Trust Framework for Detecting IoT Device Attacks in Smart Homes

1Hussein Alsheakh* and Shameek Bhattacharjee*
Department of Computer Science, Western Michigan University, Kalamazoo, Michigan 49008, USA
Email: {hussein.s.alsheakh, shameek.bhattacharjee}@wmich.edu

Abstract—Trust in Smart Home (SH) Internet of Things (IoT) technologies is a primary concern for consumers, which is preventing the widespread adoption of smart home services. Additionally, the variety of IoT devices and cyber attacks make it hard to build a generic attack detection framework for smart home IoT devices. In this paper, we present a roadmap towards building a unified approach towards establishing trust scores as an indicator of the security status of an IoT device in a smart home that works across multiple attacks and device types/protocols. Specifically, we first introduce artificial reasoning inspired evidence collection approach by introducing a small set of factors that are affected significantly if a smart home IoT device is under attack. Thereafter, we propose an explainable trust scoring model that maps the device level evidence into trust scores in a way that produces lower trust scores when devices are under attack. Specifically, the trust model involves an Augmented Bayesian Belief based Model embedded with novel non-linear weighing functions; explicitly designed to account for the severity of the attack, probabilistic discounting of parts of the evidence caused by benign changes, thus explaining our success. For evaluation of the framework, we use two real datasets that contain a variety of actual cyber-attacks and benign traffic from seven different smart home IoT devices. Our evaluation seeks to investigate the generality of our framework across multiple datasets, with various classes of IoT devices and cyber attacks.

Index Terms—Internet-of-Things, Trust, Security, Smart Home, Artificial Intelligence based Security, Machine Learning

I. INTRODUCTION

Smart Homes are environments offering services to home inhabitants via a network of communication-enabled IoT devices that contain embedded software. IoT devices communicate with each other/remote service providers and often make decisions without human intervention. However, the concept of a smart home is a cornerstone of smart connected communities. Smart home exposes the most private and vulnerable of our personal spaces to the internet, which is known to be vulnerable to several classes of cyber-attacks [1].

Nonetheless, traditional cyber threats at best had economic impacts on businesses and breaches of personal data at rest. In contrast, in the evolving paradigm of smart home IoT, the cyber attacks have an immediate civilian impact that is riskier. Moreover, there are new IoT technology-specific security challenges that need to be handled explicitly.

Since IoT devices have constrained memory and computational resources, strong on-device security solutions are barely possible without significantly reducing their usability. Additionally, there is a lack of vendor motivation to implement security schemes in IoT products [2] due to cost versus utility trade-offs, while other vendors do not implement too much security to improve latency and quality of service and reduce costs [3]. For example, after analysis of smart home data, researchers in [2], [3] found only one IoT device that uses secure encrypted SMTP protocol, and only 'some' IoT devices use secure HTTP port (HTTPS 443), and most other IoT devices are using insecure HTTP over port 80. We did not find any device use the MQTT secure port 8883 [4]. There is a lack of standards in IoT protocols and port mappings that makes it hard to have an established rule or specification based attack detection that is common to all IoT devices.

A. Challenges and Motivation

Prior work on IoT device attacks can be broadly classified into the following categories network-level data flow [5], device-specific fingerprinting [6], attack specific [7], network segmentation [8], localization approaches [9], IoT network or device components classification [11], embedded security architecture [12], or adopt known machine learning techniques for anomaly detection [10]. Localization based approaches only take care of physical proximity issue that is more related to authentication but does not detect attacks on already authenticated devices. Similarly, network segmentation involves splitting the smart home network channel to various sub-networks to mitigate the movement of attacks, but do not detect them actively. Device-specific approaches do not scale due to the large heterogeneity of device types that can be found in smart home IoT markets. Similarly, attack specific approaches need separate solutions for each cyber attack. Therefore, the number of solutions for providing a secure smart home increases.

Therefore, there is a need for a more unified solution that is not attack or device-specific, but that which generalizes across most attack types, devices, and services in smart home IoT.

B. Contributions of this Work

In this paper, we propose a ‘unified’ approach towards establishing trust scores as a reliable indicator of the security status of IoT devices in a smart home. Our framework contains...
an IoT device is an IP-enabled device (e.g., smart camera) that has limited computational abilities compared to laptops/smartphones/tablets, and offers a small set of services to the smart home (SH) owner (e.g., smart surveillance). Each IoT device has an API (or simply app.), that enables it to offer corresponding SH services. The service is offered to the SH owner when an IoT device’s corresponding app. connects to specific servers (on the rest of the internet/within the LAN) on which IoT service providers host their services. Thus, there is a mapping between an IoT device apps and services.

**Low computation IP Devices (IoT):** An IoT device is an IP-enabled device (e.g., smart camera) that has limited computational abilities compared to laptops/smartphones/tablets, and offers a small set of services to the smart home (SH) owner (e.g., smart surveillance). Each IoT device has an API (or simply app.), that enables it to offer corresponding SH services. The service is offered to the SH owner when an IoT device’s corresponding app. connects to specific servers (on the rest of the internet/within the LAN) on which IoT service providers host their services. Thus, there is a mapping between an IoT device apps and services.

**High computation IP devices (non-IoT):** These are devices that may host apps/APIs that allow the use of IoT services offered by IoT devices. Examples of such high computation IP devices include smartphones, tablets, laptops. These devices typically install many apps and software on their OS. All other sensors and actuators that are not enabled by an IP address, but collect data and perform actions, are not considered as IoT devices. The security of such devices is beyond our scope.

**Smart Home Gateway (SHG) and FloT:** Each smart home has a gateway router (SHG), that connects the IoT devices to the rest of the internet. This SHG supports services such as Network Address Translation (NAT), Software Defined Networking (SDN), and can also host middleware services using a Raspberry-Pi micro-controller. Our proposed trust model is deployed on the smart home gateway router (SHG) as a FloT middleware. All smart home services are managed and controlled by the edge/Fog Internet of Things Middleware (FloT) that is within the home network. Thus, FloT will be able to monitor SH network activity originating from a given smart home.

**B. Smart Home Operations**

Let each IoT device be denoted by \( i \in \{1, 2, \ldots, I\} \), where \( I \) is the total number of IoT devices in the smart home. Each such IoT device, provides a set of services with the help of one (or more) apps. The set of services is represented by the set \( S(i) \in \{1, \cdots, k, \cdots, K(i)\} \), and the set of apps is represented by \( A(i) \in \{1, \cdots, j, \cdots, N_j(i)\} \), where \( k \) and \( j \) denote any given service and an app respectively, while \( K(i) \) and \( N_j(i) \) are total number of services and apps offered by the i-th IoT device, respectively. In most cases, IoT devices have a single app that provides multiple services, but occasionally it might contain multiple apps. Therefore, the set of services offered by a home is \( \bigcup_{i} S(i) \).

**Scope of Our work:** We make the following assumptions for our work: 1). The use of a cellular network for any communication directly from IoT devices is beyond the scope of our work. We only consider the smart home network, which communicates through the WiFi or wired LAN that is managed and controlled by the FloT middleware hosted on SH Gateway. This paper only intends to detect misbehavior by smart home IoT device applications connect via a smart home.
In the direct attack in SH, the attacker flood the network. This kind of attack takes advantage of the IP/port forwarding feature that is argued to be a vulnerability in many device architectures. In this work, our experiments show movement and grow their attacks from one to more entities. In many instances, such malwares are known to attack IoT devices with a spoofed source IP of the victim server. The attacker uses SH IoT devise as a weapon for targeting the victims. Types of Reflection attacks include:

(i) **Custom Malware Campaigns:** These are malicious campaigns, where software designed to disrupt IoT device operations, usually used to create a Distributed Denial of Service (DDoS) attack. In many instances, such malwares are known to show movement and grow their attacks from one to more SH IoT devices and services, taking down a large section of important network entities. In this work, our experiments contain data from the malwares Hide and Seek, Muhstik, Torii, the most advanced botnet malware, according to Avast security researcher, that can attack many device architectures.

(ii) **SMURF:** it is a network attack where the attacker floods the SH router with spoofed ICMP traffic via sending a request to IoT devices with a spoofed source IP of the victim server. Thus, the device will reply to the targeted victim IP. The attacker’s goal is to disturb the victim’s server operations. Such attack behavior is to use the SH IoT device as a weapon for targeting victim servers and is related to the number of packets being sent per time interval.

(iii) **SSDP reflection:** it is a malware app running on smart home IoT devices by using the Simple Service Discovery Protocol (SSDP) used to discover IoT devices in SH. It also uses Universal Plug-n-Play (UPnP) port forwarding to allow a direct attack on IoT devices by an external attacker [15]. Mainly, the SSDP and UPnP port forwarding are conventional in SH network because some IoT devices support peer to peer applications. The adversary uses the SSDP protocol to discover the devices and services in SH without explicit configuration using SSDP vulnerabilities.

(iv) **SNMP reflection:** During an SNMP reflection attack, the adversary sends out a large number of SNMP queries with a spoofed IP address (of victim’s IP) to smart home IoT devices, that in turn reply to that victim IP instead of the attacker IP. The attack volume grows as the number of compromised smart home IoT devices (that continue to reply to the victim) increases, the victim becomes crippled, due to the huge volume of SNMP responses.

**Co-Domain DDoS and Leakage:** This kind of attack takes advantage of the IP/port forwarding feature that is argued as necessary to support QoS guarantees by ensuring server availability and load balancing. Therefore, a designated server may forward IoT device requests to other servers in the same network/cloud domain or third party providers. In such an instance, the IoT device gets a response from a different server other than the typical server of the IoT provider. Normally, such occurrences are less during benign conditions; however, increased frequency and volume of such events should indicate suspicion. This is because many servers belonging to the business rivals/competitors/adversaries may be co-located in the same network/cloud domain as the designated server for that service. Taking a completely conservative approach of treating every such instance as suspicious will either increase false alarms, reducing the efficiency of IoT services.

**Attack Strength Variation:** IoT devices have a limited processing capability for network traffic. Therefore, if the reflection attack is too high in volume, the device may stop working altogether. Intelligent attacks would like to keep compromised devices still operational to allow launching attacks on other machines. In such cases, the attacker may keep traffic rates low because it wants to conceal the attack’s effect or if its real target is not the IoT service provider.

In another attack, the attacker may be willing to make the IoT device less or not functional, and thus keeps traffic rate medium or high accordingly. Thus, traffic volumes, even for the same attack type, vary according to the goals or motivations of adversaries who are difficult to predict. To embed this, we parameterize the attack volume by varying attack strengths in our simulated dataset. From the real dataset obtained from [10], [19], we used three different attack strengths per attack: (i) high attack traffic rate, which generates 100 packets per second (pps), (ii) medium attack traffic rate, generate 10 pps, (iii) low attack traffic rate that generates 1 pps.

**IV. PROPOSED FRAMEWORK**

Our proposed framework will be deployed as a framework in the FloT, which inspects and manages data plane traffic flow (e.g., recording source and destination IPs:port mappings) to/from all SH IoT devices. Our framework has three main phases: (a) Access Control Policy Engine (b) Evidence Collection via Unified Factors, (c) Trust Scoring Model.

The Access Control Policy engine sets baseline rules of benign access for the apps of each IoT device. Thereafter, based on which parts of traffic flow attributes align with the service level access control policy, we propose a set of unified factors that remain largely unaffected under legitimate benign changes but show variations under attacks. These sets of factors produce a set of features per device as a body of evidence that serves as an input to our proposed trust scoring model. This evidence collection phase partly uses the ACL to compare network traffic attributes captured at the FloT, although there are non-ACL aspects of evidence collection. Our trust scoring model is an artificial intelligence inspired approach that maps evidence into the non-IoT devices/applications of the smart homeowner.
trust scores, such that devices under attacks show lower scores compared to when they are not attacked.

A. Service Access Rule Policy

Typically, an access control policy engine sets static and dynamic access rules (known as Access Control List (ACL)) for devices explicitly registered with the FloT. The decision on whether a device should have authorized access to a service is contextually related to the functionality.

While IoT devices exhibit heterogeneity, the number of services they offer is limited. The Manufacturer Usage Description (MUD) is a novel IETF specification that offers suggestions on the expected behavior of an IoT device contextually related to the services it offers. Specifically, the key idea of the MUD is to return a list of acceptable or expected destination DNS names and destination port numbers, which identifies one or more designated servers for the corresponding service that an IoT device should offer as well as ancillary end-points it might communicate to. Although not a security specification itself, MUD profiles can be used as a first step towards protection.

Note that what is included in the MUD specification, and how security engineers/administrators interpret the MUD specification for access control are deciding aspects for achievable security level. In a previous work [10], the idea of the access control list was synonymous with the MUD profile. Their idea of the MUD profile was to include all communication patterns observed in the benign dataset. However, our treatment of access rules is different from previous work, since we differentiate between MUD and access compliance.

Our FloT uses the DNS lookup to resolve the domain name in the MUD to the IP address of the servers that are supposed to offer the services. Thus, MUD can be used to point all IPs and port no.s that should be offering services or communicating with that IoT device. Now, this most commonly includes the IoT service provider and the hardware vendor. For specific IoT devices like smart cameras and home assistants, they connect to a lot of different servers other than the service provider, and there the volume of packets exchanged is important. Hence, our ACL design goal is to manage a table that sets baseline benign access rules of each IoT device’s app/firmware j both the client-side (Source devices) and server (Destination) side based on the above awareness.

1) Client Side: When an IoT device is first introduced in SH, it has to be registered with the FloT as an authenticated device. The Source IP, Source MAC, and Source Port Number is first added to the ACL. In this way, FloT makes sure that all active data flows at least do not directly come from outside the network and are registered IoT devices based on their IP and MAC addresses.

2) Service Side: To find a mapping between IoT devices and their corresponding designated server’s (that provide its service) destination port numbers, destination IPs, and DNS names, our FloT queries the MUD server that returns a port-based list that specifies those DNS names, and ports and that should be typically accessed by a given service offered by an IoT device. The MUD URLs are updated via various IoT providers and manufacturers. While not all manufacturers subscribe to the MUD specification, it is becoming common over time, and thus will become a practical approach. The information added to the ACL includes Destination Port No., Destination IP, Service ID, Service Type, as shown in Fig. 2. Note that we only put the designated IoT service provider server as a whitelisted entry. Everything else is subject to different levels of suspicion as detailed in the factors for evidence collection. The reason for doing this is given below:

From studies of datasets, we found gray areas in IoT devices such smart cameras (that use STUN protocol) or IoT home assistants, which need to access random unauthenticated servers that may or may not be servers in the same cloud domain, as the IoT provider. Making them part of ACL may cause missed detection of attacks. Additionally, we have found several cases where IP and port forwarding causes the response to arrive from a different server than the one listed on the MUD profile. Thus, using the MUD directly as ACL will increase false alarms. Additionally, packets are exchanged for maintenance or monitoring purposes between the hardware manufacturer and device, but such an occurrence is ‘infrequent’.

Our view of a service based ACL includes the following: Each IoT device has a list of authorized IPs (corresponding to servers that provide services related to that IoT device) for each app(s) hosted by that device. For example, the services of IoT device i is provided by using two apps j1 and j2 services app, where both j1 and j2 allowed to have a conversation with only specific devices and servers using IPs of these devices as presented in Fig 2. The app/firmware is present in the form of destination IP:port entry.

<table>
<thead>
<tr>
<th>MAC Source IP</th>
<th>j 1</th>
<th>Source Port</th>
<th>Service ID</th>
<th>SH Services Description</th>
<th>Destination Port</th>
<th>Destination IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>.2a .100 4</td>
<td>2</td>
<td>42805</td>
<td>01</td>
<td>surveillance camera app</td>
<td>5222/443</td>
<td>.296</td>
</tr>
<tr>
<td>.5e .101 4</td>
<td>2</td>
<td>42804</td>
<td>02</td>
<td>firmware</td>
<td>2457</td>
<td>.223</td>
</tr>
<tr>
<td>.74 NA NA NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>.2a .11 5</td>
<td>1</td>
<td>8500</td>
<td>02</td>
<td>firmware</td>
<td>2795</td>
<td>.33</td>
</tr>
<tr>
<td>.2a .11 5</td>
<td>2</td>
<td>75450</td>
<td>05</td>
<td>door locks security</td>
<td>4651</td>
<td>.22</td>
</tr>
<tr>
<td>.2a .11 5</td>
<td>3</td>
<td>7451</td>
<td>11</td>
<td>home security on/off</td>
<td>8745</td>
<td>.74</td>
</tr>
</tbody>
</table>

Figure 2: Smart Home access control policy

B. Evidence Collection Mechanism

A communication (packet exchange) not matching the ACL does not automatically mean an attack and requires further checks that we incorporate in our evidential model. Typically, anything that does not match with an ACL is denied or deemed suspicious, thus increasing false alarm rates, decreasing usability. Our interpretation of service level access control is not strictly binary due to the observed variations in IoT behaviors. Similarly, interaction confirming the access control list does not necessarily imply no attacks, which we also incorporate in our evidential model. The above awareness is central to our
design of unified set of factors. From here, we will refer to the packet exchange via the FloT as an interaction.

Intuitively, the body of evidence should point towards degrees of positive, negative, possibly suspicious interactions based on the body of evidence. The goal is to find (i) a methodology for labeling the interactions; (ii) corresponding mathematical representations of these factors (known as features) that captures the effects of attacks in a numerically quantifiable way. Thus, features form the mathematically tractable evidence that is driven by our proposed set of factors.

We propose the following set of factors: (1) Service Access Uncertainty (2) Cumulative Volume of Uncertain Accesses (3) Service Access Violation (4) Access Violation Diversity. Below we describe each of these factors.

![Diagram showing evidence collection](https://via.placeholder.com/150)

**Figure 3: Evidence Collection**

1) **Service Access Uncertainty:** Packet flows originating from a device not matching the destination IP and Ports in the ACL may or may not be necessarily due to an attack. Hence, an intelligent framework should not blindly label this event as a negative interaction. Not matching flows can be attributed to benign ‘events’ in smart home IoT operations, such as:

   (1) Hardware Vendor checks: Packets are sent from IoT devices to servers that belong to their third-party vendors for monitoring purposes (such as estimating the remaining life of a device, etc.). (2) IP and port Forwarding: Sometimes, traffic is redirected to a different IP or Port than the designated one in the MUD, to guarantee quality of services (QoS) issues (e.g., too many requests concurrently can be forwarded) or in case of server maintenance/upgrades/downtimes. In such cases, the incoming response to an IoT device comes from a different server but is usually within the same network/cloud domain as the authorized server in the ACL. (3) use of STUN servers for smart cameras and video streaming services: From our study of smart home IoT datasets and previous literature, we found most cameras use the UDP port 3478 for communicating with various servers on a peer to peer mode to allow seamless access to smart surveillance from anywhere. Such servers cannot be learned and included from the MUD profile.

   However, the above, if treated as a match in the ACL can allow adversaries to create cyber attacks on servers within the same cloud domain, third party vendors, or those that host streaming. This gives the intuition that we should treat them as between a match and mismatch(i.e., uncertain). Therefore, we believe that once a packet exchange does not match the baseline ACL, we first check if it might be due to any of the above three reasons by the following procedure (summarized in Fig. 3).

   Our FloT queries an IP lookup service Info-Sniper [24], that provides meta information for any IP recorded in the network traffic. This information includes: (i) Name of the Owner/Provider (ii) Geolocation (iii) DNS name (iv) Time Zone. Therefore, packets sent to a given destination IP is checked by our FloT to ascertain whether the Name of the Owner and other attributes match with (1) any of the third party vendor/manufacturer of the IoT device, (2) whether they are within the same cloud/network domain name, or (3) use the STUN protocol’s UDP port (if the device offers streaming video services). To conclude, if a packet exchange does not match the access control list, the FloT checks for the above-mentioned possibilities using the context information queried from the Info-Sniper lookup service. If there is a match, we label this packet flow as ‘uncertain.’ If not, then we would label this packet exchange as a violation. Mathematically, we denote an uncertain packet exchange for services as $U^{(i)}(s)$. **Number of Uncertain Packets Per Time Window:** Therefore, the corresponding feature that we keep track is the total number of uncertain packet exchanges for a given device $i$ within a time window $T$, denoted as

   $$\eta_u^{(i)}(T) = \sum_{T} U^{(i)}(s)$$ (1)

2) **Cumulative Volume of Uncertain Accesses:** One important fact about the events that cause uncertainty is that when they are due to benign behavior, the packet volume is low. Uncertain exchanges can be allowed a benefit of the doubt (discounted) while calculating the trust scores of a device. However, beyond a certain volume of total uncertain exchanges, this situation should be viewed as suspicious, owing to the infrequent pattern of such occurrences. This aspect is later embedded using an uncertainty weight function, which is a function of the total uncertain packet volume.

   **Cumulative Sum of Uncertain in a Time frame:** At each time window, we keep a cumulative sum of uncertain interactions over a sliding frame of previous F time windows.

   $$\mu_u(T) = \sum_{T-F}^{T} \eta_u^{(i)}(T)$$ (2)

3) **Service Access Violation:** The next factor for evidence collection for potential malicious activity is the service access violation. The main idea is that if an interaction from an IoT device does not match the authorized access control list entries and can neither be reasoned as uncertain, then this event of access is treated as a violation represented as $(V^{(i)}_{j})$, i.e., the violation of app $j$ that runs inside the device $i$.

   **Number of Violation Packets Per Time Window:** In fact, for that time window, $T$, the FloT middleware calculates the total number of access violation of K services by all the apps $j_1$ to $j_n$ running on IoT device $(V^{(i)}_{j})$. 

\[\text{Number of Violation Packets Per Time Window:} \sum_{T-F}^{T} V^{(i)}_{j} \]
4) Access Violation Diversity: The Access Violation Diversity $w_d$ represents the extent of the cross-section of services/servers being affected by a set of violations. Suppose if one device is trying only to access one service ten times, whereas another device is trying to access ten different unauthorized services, within the same time window $T$, the second scenario is far more serious from a security perspective. Thus, Access Violation Diversity is an important feature to determine how the attacker is trying to grow its attack impact and compromise more resources.

Number of Unique Services with violations $\eta^{(i)}_v(T)$: The number of unique services that experienced violations recorded for a given IoT device $i$ in a time window $T$. Mathematically, it is represented by the following:

$$\eta^{(i)}_v(T) = \sum_{s=1}^{K} I(i, s)$$

(4)

where, $I(i, s) = \begin{cases} 1, & \text{if } i \text{ violated service } s \\ 0, & \text{otherwise} \end{cases}$

(5)

From the network traffic analysis of attacks (in a cross-validation set), we found common links between different attacks. Once our rule of matches, violation, and uncertainty are in place, most real IoT cyberattacks either registers violation with increased packet volumes, or an abnormal increase in the cumulative volume of packet exchange to uncertain destinations, as well as the relative volume of uncertain interactions. In Fig 4, we show a summary of the observed relationship between different types of reflection and direct attacks, and the factors we put forward, which succinctly explains why our factors work in terms of building the appropriate evidence.

<table>
<thead>
<tr>
<th>Attacks Types</th>
<th>Features</th>
<th>Factors</th>
<th>Security Services (Risk)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct</td>
<td>Top/TCP/UDP/TLS Attack</td>
<td>Number of Unique Services</td>
<td>Total Payload Volume</td>
</tr>
<tr>
<td></td>
<td>Denial of Service</td>
<td>Number of Violations</td>
<td>Access Control</td>
</tr>
<tr>
<td></td>
<td>Off Line Attack</td>
<td>Number of Attacks</td>
<td>Security Rollout</td>
</tr>
<tr>
<td>Medium</td>
<td>TCP/UDP/FTP/Email</td>
<td>Number of Unauthorized Sessions</td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>Worm/Spoofer</td>
<td>Number of Malformed Packets</td>
<td>Intrusion Detection</td>
</tr>
<tr>
<td></td>
<td>RDBP</td>
<td>Number of Malicious Servers</td>
<td>Encryption</td>
</tr>
</tbody>
</table>

Figure 4: Attack and Factor Mapping

C. Trust Scoring Model

Note that comparison of packet exchanges with ACL has only three mutually exclusive outcomes, viz., matching, violation, and uncertain interaction. Notice that access violation diversity and volume of uncertainty are not mutually exclusive to violations and uncertain, so we ignore them for now. Based on the evidence and the past evidence (if available), it is well known that the posterior probability of observing a matching, violation, or an uncertain interaction can be calculated using posterior beliefs calculated using Bayesian inference where the data is a multinomial distribution of the number of matches, mismatches, and uncertainty.

Previous studies [20] have shown that for multinomially distributed data $D_i$ with a non-informative prior, the posterior probability belief $\theta_i$ is given by the mean of Dirichlet distribution as the following:

$$f(\theta_i|D_i) = \frac{D_i + 1}{\sum_{i=1}^{K} D_i + K}$$

(6)

Where $K$ is the total number of mutually exclusive categories in the evidence state space, $i$ is the $i$-th category, and $D_i$ is the frequency of occurrences of the $i$-th event in the data. Degrees of Disbelief and Uncertainty Using the result in Eqn. 6, we apply the known derivation provided in [21], to find the degrees of disbelief and uncertainty as:

$$d^{(i)} = \frac{n_v^{(i)} + 1}{N^{(i)} + 3}$$

(7)

Embedding Access Violation Diversity: However, in Eqn. 7, the $d^{(i)}$ only indicates the proportion of violations to the total number of accesses. It does not provide any idea on the attack scale or surface over which this proportion of violation was recorded. If the same proportion of violation is recorded from a unique number of devices, this is a more severe attack than the same proportion of violation targeting only one device. Therefore, we need to add weight to the $d^{(i)}$, which embeds the importance of how many unique devices were involved in the observed proportion of violations. This is similar to neural learning approach. The question is what functional form should describe this behavior. We use a modified softplus function scaled between zero and one, by the following Eqn.

$$\tau_v^{(i)} = \log(ae^{n_v} + 1)$$

(8)

where $0 < \tau_v < \infty$. The above Eqn. is a modified softplus function, with an augmented parameter $a$, which controls both the steepness of the function and the initial bias (y-intercept) of $\tau_v$ when $n_v = 0$. Finally, we scale this to an interval between 0 and 1, giving the final equation for the weight of the violation.

$$w_d = 1 - e^{-|\tau_v|}$$

(9)

Thus the combined disbelief mass $w_d,d^{(i)}$. The product indicates heightened suspicion that cyber attack is more serious, if the surface is larger.

Embedding Cumulative Uncertain Volume: The intuition behind the design of the weighing function of the raw, uncertain probability mass is that the suspicion increases as more number of uncertain interactions emerge. However, up to a certain number of uncertain interactions should be discounted.

$$w_u = \frac{1}{(1 + A_b e^{-B_b \nu_u})^{1/\nu}}$$

(10)
We envision trust score is a 'complement of total disbelief,' which includes the weighted violation and weighted uncertainty masses that should contribute to disbelief. This is mathematically represented as the following:

\[ TR^{(i)} = 1 - w_d d - w_u u \]  \hspace{1cm} (11)

Plugging in \( w_d, d, w_u, u \) into Eqn. 11, we get \( TR^i \). Now we know that in logistic regression, the response variable is linked to the independent variable via a non-gaussian distribution and therefore needs a link function. The typical link function for a binary response variable (trusted or not trusted). Typically, when the response is a binary, the following logit link function is an appropriate link function that is given by:

\[ E^{(i)} = \log \left( \frac{TR^{(i)}}{1 - TR^{(i)}} \right) \]  \hspace{1cm} (12)

The Eqn. 12, facilitates trust scores that are linearly separable by a threshold that helps in classification between trusted or non-trusted devices. The \( TR^{(i)} \) is in the interval between \(+\infty\) and \(-\infty\) that is scaled to the final trust score using 13.

The final trust score \( FR^i \) is scaled from the real value plane to an interval between \([-1, +1]\) by the following equation, to maintain convention of trust metrics representation:

\[
FR^{(i)} = \begin{cases} 
+ \left( 1 - e^{-|E^{(i)}|} \right), & \text{if } E^{(i)} > 0 \\
- \left( 1 - e^{-|E^{(i)}|} \right), & \text{if } E^{(i)} < 0 \\
0, & \text{if } E^{(i)} = 0 
\end{cases}
\]  \hspace{1cm} (13)

The supervised approach is feasible if there is the availability of labeled datasets of attacks and no attacks. In such cases, the training set and cross-validation sets can be used to observe the difference in trust scores of the two classes. The cross-validation set includes a set of attack samples as well.

D. Parameter Learning

For optimal parameter selection of the trust model, one needs to decide on an appropriate loss/error function. In this case, our loss function \( e \) is the square difference of the sum of compromised device trust scores and the sum trust scores of honest device set, which will improve the classification. The goal is to maximize the \( e \) which defined as:

\[
e = \left( \frac{\sum_{benign} FR^{(i)}}{H} - \frac{\sum_{attack} FR^{(i)}}{M} \right)^2
\]  \hspace{1cm} (14)

s.t. \( A_0 > 0; \quad 0 < B_0 < 1; \quad 0 < \nu < 1; \quad a > 0 \)

where \( H \) and \( M \) are numbers of devices used in the cross-validation process from the benign and attack sets, in our case, \( H \) and \( M \) are equal since we have for each device an instance of the benign in the training and instance of the attack in the cross-validation set. The solution of Eqn. 14, can be done through two alternatives: (a) Grid Search Approach (b) A Gradient Descent based Approach. A gradient descent approach is more computationally efficient for finding the optimal parameters when the number of parameters is large, and the search space in each parameter is huge.

However, for the gradient descent approach to be applicable, the error/loss function needs to be convex, which is difficult to guarantee. In our problem, the number of parameters is less (i.e., 4), and two out of them have a limited search space. Hence, for simplicity, we use a brute force grid search method to find the parameter values that maximize the error function. Thus, we get an approximate optimal parameter selection.

Our definition can be easily extended to gradient descent by taking a negative logarithm of the above error function (which is concave). However, the resulting approximate convex function may need further processing to ensure the differentiability and existence of global minima. For this work, we have ignored the gradient descent based approach because the scale of our optimization problem has been made drastically smaller than the traditional ML techniques (via the set of unified factors), which have a large parameter set.

After the optimal parameters have been identified, we map the trust scores for a device for the benign and a small portion of the attacked dataset (cross-validation set). We then use an SVM to find the optimal separation between these labels of trust scores for all devices belonging to the UNSW dataset. The learned threshold \( \Gamma \) serves as a classifier in the test set.

V. EXPERIMENTAL RESULTS

Here we present the experimental results of our scoring model. First, we provide a dataset description followed by the trust scoring observations for each device.

A. Dataset Description

For validating our model, we use devices from two different datasets: (1) UNSW Smart Home IoT dataset [10] and the (2) CTU Apostemat IoT-23 dataset [22].
The UNSW dataset contains a set of IoT and non-IoT devices and also MUD profiles. We select benign data of 10 days and attacked data of 5 days from four IoT devices (Amazon Echo, Philips Hue Bulb, iHome Smart Plug, Samsung Camera) for showing performance. Our device selection is based on the attacked devices mentioned in previous work on [10] to compare performance. The attack dataset had contained: (i) direct attack (viz., TCP SYN (referred to as TCP SYN (D) in the table), and Fraggle) (ii) reflection attack (viz., SNMP, SSDP, DDoS, TCP SYN (referred as TCP SYN (R) in the table) Samsung Smart Camera, and iHome Plug. Out of the five days, two days were used as a cross-validation set, and the rest of the three days was our testing set.

We also used labeled benign and attacked database for real network traffic obtained from CTU Avast [22]. We used the second dataset to prove the generality of our framework. We chose two devices Somfy Smart Door Lock and Amazon Echo, that contain reflective DDoS attacks through malwares; Torri, Hide and Seek, and Muhstik. In Table I we present a mapping showing the implemented attacks launched against IoT devices in the labs of both UNSW and CTU datasets.

Table I: Mapping IoT devices with attacks

<table>
<thead>
<tr>
<th>Attack types</th>
<th>Echo UNSW</th>
<th>Hue UNSW</th>
<th>S.Camera UNSW</th>
<th>iHome UNSW</th>
<th>Somfy CTU</th>
<th>Echo CTU</th>
<th>Hue CTU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hide &amp; Seek</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Muhstik</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Torri</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNMP</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>TCP SYN (R)</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smurf</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>TCP SYN (D)</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fraggle</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDoS</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

B. Experimental Set Up

We use Wireshark to parse the raw .pcap traffic files into CSV files and partition them to 10, 2, and 3 days of training, cross-validation, and testing sets. We create the ACL table flow rules based on MUD profiles. In this paper, we calculate all factors and features based on packets per second that we derive from the data as well as the flow directions. More details on this can be found [10]. However, how we interpret the MUD for the ACL and factor mapping is done differently than [10], as explained earlier, which explains better accuracy in our method.

In all the figures, we show a continuous evolution of trust by retrofitting the parameters across the whole duration of the dataset, to give a sense of how the real-time evolution of trust values happen as soon as there is some incidence of attack.

C. UNSW Dataset Results

Figs. 6a, 6b, 6c, 6d respectively correspond to results from the UNSW dataset, that shows the evolution of trust values for Amazon Echo, Philips Hue Bulb, iHome Smart Plug, Samsung Smart Camera IoT devices. In each of these figures, both the instantaneous raw trust values per time window $FR(i)(T)$ (dotted blue lines) and the cumulative weighted moving average over the instantaneous trust values (solid orange lines) are shown. Additionally, each result has two regions; no attack days and attack days. Our three days of the testing set corresponds to the attacked days. To give a sense of how quickly our framework responds to the incidence of attacks, we created an augmented dataset with the ten days of benign set appended just before the three days of the test set (containing attacks). We then applied our model with the learned parameter. Below we describe the specifics of observations for each device in the UNSW dataset.

1) Amazon Echo: Amazon Echo device from UNSW contains Fraggle and ARP spoof attacks, as reported by [10]. Fig. 6a, shows that our framework produces reduced trust values in the attacked dataset compared to the benign dataset, prove that the FR value reflects the attack behavior.

2) UNSW Philips Hue Bulb: The Philips Hue Bulb IoT device from UNSW contains attacks such as SSDP and SNMP reflection, Smurf, TCP SYN flood as reported by [10]. Fig. 6b, shows that trust values drop after the attacked dataset starts performing malicious activity. Here the attack volume is targeted to those within the cloud domain of the provider, and the attack volume is lower. Therefore the drop in the trust value is not as strong as in Amazon Echo. The start and stop times of each attack type are not labeled in the dataset, which proves that our method need not know such attack specifics for detection.

3) UNSW iHome Smart Plug: The iHome Smart Plug IoT device from UNSW, contain a DDoS attack launched after an ARP poisoning exploit [10]. Fig. 6c clearly shows the reduction in the trust value during the attack days, although the time it takes to start decreasing is much higher than the other devices. This is because the attack volume is at times lower than the benign total volume, as revealed from the data traces.

4) UNSW Samsung Camera: The Samsung Smart Camera IoT device from UNSW contains attacks such as TCP SYN, Fraggle, SMURF etc. and reported by [10]. Congruent to previous observations Fig. 6d, the progression of decrease
in a smart home gateway. Through this work, we showed that it is possible to move towards the unified treatment of detecting cyber attacks on smart home IoT devices.

ACKNOWLEDGMENT
This work is partially supported by NSF grants SATC-2030611 and OAC-2017289.

REFERENCES