Hestia: Designing network views for Smart Homes

Sanket Goutam
North Carolina State University
sgoutam@ncsu.edu

William Enck
North Carolina State University
whenck@ncsu.edu

Bradley Reaves
North Carolina State University
bgreaves@ncsu.edu

ABSTRACT

Smart home devices are notoriously vulnerable to being exploited in many different ways. The current best practices for securing smart homes against vulnerable devices compromising each other suggest using separate WiFi networks, one for all the IoT devices and one for users' personal computing devices. But this increases the network management overhead for an average user. Ideally, to prevent device compromises, an explicit network policy should be defined for each individual device that controls its functionality within the local network. While access control specifications, like MUD, for IoT devices have been proposed there is no guarantee of them being readily available anytime soon. To this extent we present *Hestia*, a network policy design that provides default network access, based on the principle of least privilege, to devices in smart homes. Hestia uses a classification of devices as controllers and non-controllers to create these policies, with controllers being the devices that receive user commands to execute actions on non-controllers . We validate Hestia's policies against a large public data set of smart home devices and find that it accurately preserves selective network isolation for all devices on the local network without affecting their functionality. We prototype Hestia as a SDN controller and the least privilege policies generated as OpenFlow rules on the wireless access point. Through selective network isolation, Hestia provides smart home owners with an effective way to readily enforce least privilege access control for their devices thereby acting as a default network security measure for smart homes.

KEYWORDS

IoT & network security; smart home; least privilege policy

ACM Reference format:
Sanket Goutam, William Enck, and Bradley Reaves. 2018. Hestia: Designing network views for Smart Homes. In *Proceedings of ACM Conference, Stockholm, Sweden, June 2018 (WiSec'18),* 10 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Smart Home is the term which refers to a household where home automation or domotics [28] is used to seamlessly network devices and appliances to provide users with a complete control over all aspects of their home. Home automation has been around for many decades in the form of lighting and simple appliance control. But the advancements in wireless IoT network protocols have caught up with the idea of the interconnected world, providing automation at our fingertips on smartphones, and through voice commands to virtual assistants. Interfacing of IoT devices in the household has offered us the convenience of remotely monitoring temperature and carbon monoxide sensors, access video surveillance, track our pets' movements, and control door locks among many other things. Additionally, connecting these devices together has enabled us to programmatically play music while making coffee in the morning, open window blinds when we snooze our alarm multiple times, and have Alexa trigger disco lights on a Friday night.

The majority of IoT devices - from smart plugs to thermostats, door sensors to security cameras - all developed by different vendors come with their own apps, their own systems and connections, and many of them can only interact with products from the same manufacturer. So, in order to foster inter-vendor compatibility and encourage community based software development ecosystems, few smart home platforms have been developed like Samsung's SmartThings [49], Apple's HomeKit [5], Google's Brillo and Weave [13], Vera Control's Vera [63], and AllSeen Alliance's AllJoyn [23]. These platforms act as one unifying box enabling users to pair the functionality of devices from different vendors through automation recipes. All of the users devices communicate through a local gateway (e.g., a hub or a base station) or through cloud back-end servers. One command to rule them all, One hub to find them, One network to bring them all, and in automation bind them.

Although not really intended to win domination over the middle earth unlike the one ring, a survey has shown that Smart Home technology has already achieved 33.2% penetration in households in 2019 and is expected to hit 53.9% by 2023 [59]. Along with this increasing popularity of unifying myriad devices under the same network is also the increasing concern on security and privacy breaches in smart homes. Security problems in the Internet of Things (IoT) have been widely known to cause severe operational failures [12], mass exploitation [3], and catchy headlines [65]. Since IoT devices typically are embedded deep inside networks, they become attractive attack targets and often the weakest link for breaking into the users house [50] or for leaking sensitive information about users and their behaviors. Prior incidents [64, 65] have shown how seemingly simple IoT devices have been exploited to compromise other systems on the same network. Even popular smart home devices, including the WeMo switch and motion sensors [33, 44], Belkin NetCam [56, 57], baby monitoring devices [42, 53, 56], and smart light bulbs [29], have been found vulnerable to a local network attack: an adversary within the same network is shown to be able to control these devices or steal sensitive user information. Security experts suggest that this is mostly because such devices have poor or no security features, primarily attributed to manufacturers rush to the market, and hardly having any good way to patch them [50].

WiSec'18, Stockholm, Sweden

DOI: 10.1145/nnnnnnn.nnnnnnn

Traditional mechanisms of securing users computing devices include software updates, endpoint security solutions, and network-based security solutions. However securing smart home IoT deployments using these traditional mechanisms introduces significant challenges. Firstly in most cases there is limited, if any, product life-cycle support available for IoT devices due to the lack of regular firmware updates or security patches [40, 67]. Secondly providing effective end-point security, like anti-malware, on IoT devices is not really feasible due to the lack of firmware support, software APIs and limited resources.

Securing network communications is a more practical solution for IoT devices because they require constant network connectivity for their operations. However, traditional network-based security solutions involve network managers that are responsible for setting up traffic filtering rules at the network perimeter e.g. Firewall, IDS, etc. Traffic between wireless IoT devices in a smart home typically stays within the WiFi network, and thus firewall policies would be oblivious to any communication between these smart home devices. Solely relying on existing network configurations and firewall abstractions is insufficient. So the challenge that our work addresses is providing an effective network security measure for smart homes.

In this paper we present *Hestia* as a default network-based security mechanism for smart home devices which approaches a least privilege policy in networks by providing selective network isolation among the connected components. The key insight of *Hestia* is that communication between existing smart home devices is not as complex as we might think, and an effective least privilege network access can be created with a simple classification of devices as controllers and non-controllers . We define **controllers** as the devices that get information or commands from the users to perform actions on other IoT devices. Examples of controllers may include automation hubs, voice assistant devices like Echo and Google Home devices, and smart entertainment systems with voice assistants that can also perform actions on other devices based on users command. We associate all the other IoT devices, primarily including sensors and actuators, that perform these actions and are informing the controllers in the network as **non-controllers** . Examples of non-controllers devices may include light switches, presence sensors, door locks, light bulbs, thermostats, etc.

By extension of our definition of controllers and non-controllers , we build a hypothesis that an ideal communication in a smart home will be so that devices identified as non-controllers will only be ”controlled” by the controllers in the network. We validate our hypothesis using a data set of labeled packet traces from the SoK project on Home-Based IoT deployments [2] and find that a least privilege network access can be provided without affecting any ideal functionality. We also provide a prototype of Hestia which can be readily installed by smart home owners as a SDN controller which uses this classification of connected devices to create least privilege access policies as explicit OpenFlow rules on their wireless access point.

2 MOTIVATION AND PROBLEM

The notion of an access control policy in networks is introduced here based on an explicit understanding of IoT devices and their purpose in a smart home. It is generally accepted that IoT devices can be defined as a group of objects that are specifically not intended to be used for general purpose computing tasks, but instead have very specific purposes. By definition, therefore, all other uses are not intended. Thus an access control policy for IoT devices should express only these specific purposes. Access control in modern operating systems limit access to files based on the user, incorporating a policy that embeds ownership and privilege based control. Similarly, in the IoT context, access control is needed to make sure that only trusted devices can access sensor data or command the actuators to perform an operation.

2.1 Use Case: Setting up a smart light bulb

We consider the situation where a smart home owner, Alice, is setting up a smart light bulb in her house and has to connect the light bulb to her WiFi network for its functionality. But she is concerned about providing the same level of network access as her work computer, to an off-the-shelf light bulb.

A motivating example explaining access control of a light bulb for such a situation has been provided by the IETF draft on manufacturer usage description (MUD) specification [36]. The purpose of a light bulb is to light a room. That is its sole purpose in any environment. It may be remotely controlled through the network, and it may make use of a rendezvous service of some form through an application on a smart phone. But, by extension of its definition, all other network access is unwanted. It will not speak to the refrigerator or the thermostat, and it has no need of accessing users' printer or the smoke alarm. Therefore, what Alice needs is an access control policy, approaching least privilege, which when applied to the light bulb should state that it will only connect to the specific devices it needs for its functionality, devices which in the context of Hestia are termed as controllers .

2.2 Research Challenges

In order to enforce network access control mechanisms in smart homes, we seek a scalable and user-configurable design that provides a practical least privilege policy specification for all the connected devices. This paper seeks to enable this vision by addressing three key research challenges.

- **Principle of Least Privilege**: How do we define what a least privilege policy of network access should be for smart homes? IoT devices in a smart home are extremely heterogeneous in terms of their system design and support different communication protocols to interact with each other. The policy, then, needs to support the heterogeneity of device communication, and at the same time preserve the notion of least privilege.
- **Implementation**: How do we interpose these policies on the communication between devices in the WiFi network? Communication between wireless smart home devices takes place over the local network and stays within the WiFi. Traditional network security measures like enforcing Firewall rules, and ACLs would not work as the gateway devices will not be able to monitor the connections within the WiFi network.

- **Multicast**: While interposing a least privilege policy on network communication, careful attention needs to be taken for multicast and broadcast requests made by these devices. Preserving least privilege in unicast communication makes sense for most cases, but devices frequently use multicast messages to discover other devices on the network. Multicast traffic is essentially a single packet duplicated for multiple devices connected to the network, and handling this properly in wireless network while interposing a policy is challenging in and of itself.

2.3 Threat Model

While our work is motivated by access control mechanisms for IoT devices, we do not focus on a complete and explicit access control mechanism for the communications of a smart home device. The manufacturer usage description (MUD) [36] provides a much more granular policy specification for IoT devices. Instead, we seek to address the broader challenge of providing a scalable and user-configurable policy specification for the communication of devices in a smart home. Our work achieves this by providing a default network access to all devices in a smart home, which approaches the principle of least privilege for IoT devices. IDIoT [8] makes a similar claim on providing a network security enforcement framework, but they only focus on device to cloud endpoint communications and provide enforcement in the form of rules at the gateway device. Their model remains completely oblivious to the communication between devices in the local network, which is the threat model that we are concerned with.

Our threat model addresses the security and privacy concerns of smart home owners, like Alice, who are worried about connecting off-the-shelf IoT devices to the same network as their personal computing devices and privacy sensitive devices. While current best practices suggest connecting all IoT devices on a separate WiFi network, it leads to a network management overhead for users like Alice and does not address the risks of having privacy sensitive devices, like security cameras, share the network with vulnerable light bulbs.

The work in this paper does not seek to prevent device compromises in a smart home, and neither does it claim of being a complete network security solution. We specifically address the security concerns of users adding new devices to their network and enabling them with a configurable policy specification to implement a least privilege network access based on the type of device they have connected.

3 DESIGN OVERVIEW

Hestia is designed with the singular goal of introducing *principle of least privilege* for smart home wireless networks. As previously discussed, IoT devices are designed for very specific purposes. In the confines of a smart home, the functionality of these devices are often combined by the user to create automation recipes. This is made possible by the introduction of smart home platforms like SmartThings [49], HomeKit [5], Brillo and Weave [13], Vera [63], and AllJoyn [23]. But the unification of myriad devices under the same network, for the promises of complete automation, requires all types of IoT devices to have the same level of network access privilege. Hestia addresses the security and privacy concerns raised

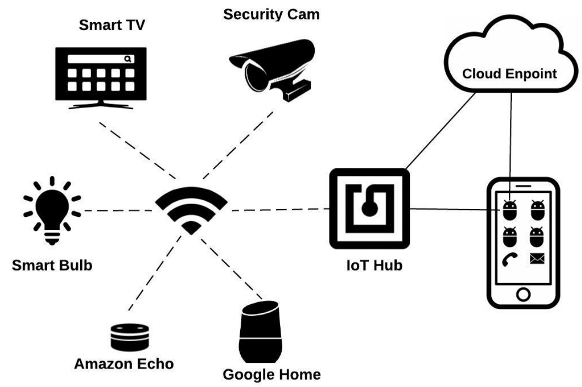

Figure 1: Smart home IoT setup

due to such network implementations by defining the network access required for each device based on its type, and then isolating the wireless communication between devices based on a privilege policy.

Let us consider the use case mentioned in the previous section in order to iterate what we wish to achieve with our policy design. Alice wants to set up an off-the-shelf smart bulb in her home. She has a home automation controller, for instance SmartThings, through which she wants to access the functionality of the bulb. In order to achieve that, the SmartThings hub and the light bulb need to be connected on the same WiFi network. Now suppose that Alice, like many other smart home owners, also has plenty other smart home devices which need to be connected to the same wireless network, and by extension have the same level of network access privilege. These devices can vary from trivial home appliances like a smart refrigerator or thermostat to more user sensitive devices like security cameras or maybe a printer. Addressing the threat model discussed earlier, what Alice then desires is a network policy that would allow the smart light bulb to be able to interact only with the home automation controller and not any other device. This, for the light bulb, would embody a least privilege policy of network access.

Hestia provides a network abstraction to address exactly that. We provide the following insights, through our design, with respect to the research challenges discussed in the previous section.

3.1 Control Interfaces

We address privilege based network access by building on an hypothesis that all smart home devices can essentially be classified into two categories, namely controllers and non-controllers based on whether they support a control interface for the user to provide commands. Consider the smart home setup demonstrated in Figure 1. Alice has multiple IoT devices setup on her network, but all her devices do not need the same level of network access. Alice wants to access the functionality of the smart bulb through an app on her phone, or maybe through an automation recipe on Alexa or Google Assistant. So in every possible scenario, the light bulb will be "controlled" by another device on the network based on some command from Alice, and can not be controlled directly. We posit

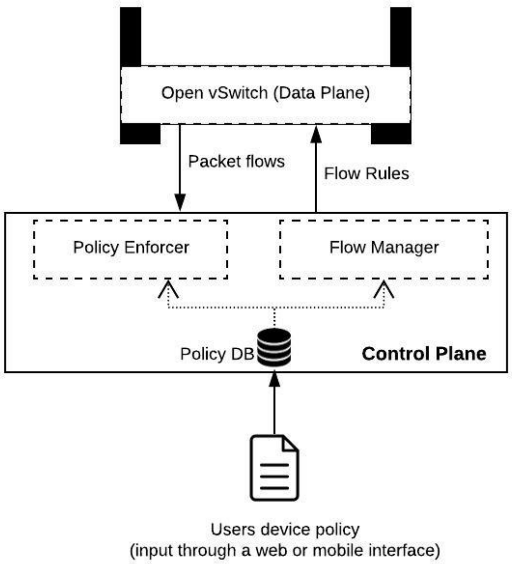

Figure 2: Hestia architecture

that this distinction between devices can be made based on whether they have the capability of taking inputs directly from the user, or support a control interface. For the devices mentioned in Figure 1, we identify Amazon Echo, Google Home, IoT Hub, and the Smart TV as the controllers in the network because they can take commands from the user. By making this clear distinction, we enable a policy of least privilege by stating that all devices identified as non-controllers can only interact with controllers in the network in order to perform their required function. In no scenario could non-controllers send "control" messages to other non-controllers , and thus this ability is an over-privileged network access. We find this distinction of IoT devices to be trivially sufficient in order to provide least privilege network access in a smart home.

3.2 High level implementation design

Fundamentally, our goal is to create a network abstraction that conserves the wireless network communication between devices based on the distinction described earlier. While traditional network security measures prove to be insufficient, software defined networking (SDN) [35] with openflow provides a flexible access control by separating the network traffic (data) from its management (control). In Figure 2, a high level abstraction of our design is shown. We have implemented two programs running in the control plane of the access point, which govern the openflow rules in the data plane. These two programs, for the ease of understanding, are termed as Flow Manager and Policy Enforcer. Flow Manager is responsible for tracking all unique packet flows that are logged by the Open vSwitch, and then installs the desired data flows on the switch based on an algorithm which we describe in the next section. Policy Enforcer monitors changes made to the policy design and pro-actively updates the data flows on the switch. The policy in reference is a list of devices identified as controllers in the smart home network. We envision two ways of obtaining this list. A straightforward way would be to have the user identify controllers during setup, either through a web interface (routers configuration page) or through an application on their smart phone [7, 38, 43, 52, 60]. The other way we envision accomplishing an automated distinction of controllers is either through looking at the higher order bits in the MAC-layer address or using more advanced classification systems such as IoT Sentinel [40]. However, this is orthogonal to the work presented in this paper and thus we identify it as an open research problem.

3.3 Multicast device discovery

Service Discovery layer has a prominent role in an IoT architecture. It is the service discovery which differentiates an IoT network from that of typical internet network. In order to avail a service, IoT devices must be connected with the devices that are hosting those services or applications. So there are various protocols (like uPnP, mDNS, SSDP, Wi-Fi Beacons, 6LowPan, etc) designed to resolve host IP addresses rendering such services and applications in the local network.

Referencing our smart home model, each IoT device provides some sort of service and uses any of these protocols to discover other services in the network. But as we have discussed earlier, every device on the network has the same level of network access privilege and thus even a (vulnerable) light bulb can discover the services being provided by the (privacy sensitive) security camera. Multicast packets are essentially a single packet broadcasted for multiple destinations. Hestia provides a selective isolation mechanism for multicast discovery messages through a generic way of message duplication and then limiting the broadcast destinations based on the policy defined for the source device.

We demonstrate this feature in Figure 3, where a multicast message generated by the light bulb is duplicated and forwarded (denoted by double dashed lines) to only the controllers in the network. The non-controllers are isolated from receiving the multicast message (security camera) while still maintaining a shared network connection. We achieve this by utilizing **Group tables** feature supported by OpenFlow 1.3 and later. Group Mod essentially defines action buckets for each flow, where the message is cloned and assigned the desired action for each destination. Our implementation

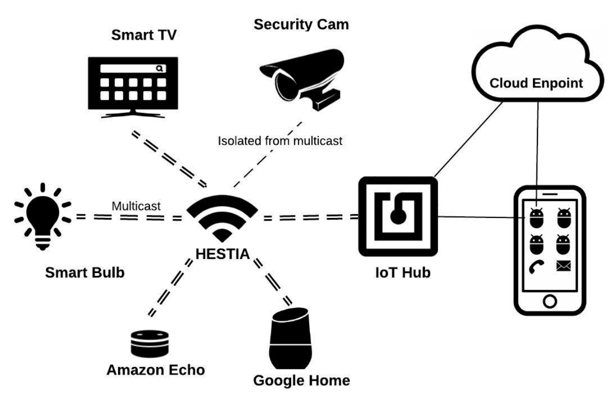

Figure 3: Handling a multicast packet

creates action buckets containing only controllers for all multicast flows originating from non-controllers thereby providing a selective isolation of devices for service discovery. This implementation is based on the insight we provide for achieving least privilege where only controllers are allowed to send messages to non-controllers and any message generated by non-controllers will be isolated to the controllers in the network.

4 POLICY DESIGN

In this section, we describe our implementation of the policy design to achieve least privilege in the smart home network. Quite specifically, we explain the *Flow Manager* application as mentioned in the previous section. Flow Manager monitors the connections being logged by open vswitch on the access point, and creates data flows based on the policy defined for the connected devices on the network.

Algorithm 1 Flow Manager

```
1:  procedure GET_BUCKETS                          ▷ Action buckets for multicast
2:      controllers ← query(select controllers from devices)
3:      buckets ← []
4:      for newdst ← controllers do
5:          temp_port ← find_port(newdst)
6:          action = set(eth_dst = newdst)
7:          if temp_port = in_port then                         ▷ WLAN port
8:              action.append(output = in_port)
9:          else
10:             action.append(output = temp_port)
11:     buckets.append(action)
12: procedure PACKET_IN_HANDLER(src, in_port, dst)     ▷ monitor
    packet flows
13:     if dst is multicast and src is not controller then
14:         b ← get_buckets()
15:         group_msg ← GroupMod(b)
16:         send_msg(group_msg)
17:     else if dst is multicast and src is controller then
18:         actions = set(output = FLOOD)   ▷ Full network access
19:     else
20:         actions = set(output = out_port)            ▷ Unicast flows
21:     add_flow(actions)
22:     if src is not a controller and dst is not a controller and
    not(multicast) then
23:         action ← DROP
24:     add_flow(match(src, dst, in_port), actions)
```

Algorithm 1 depicts the logic used to generate the openflow policies to attain least privilege in the network. Hestia logs each packet_in event on the switch and retrieves the tuple (src, dst, in_port) from the packet headers. The *GET_BUCKETS* procedure is used to create the action buckets required for the group policies to handle multicast packets. As mentioned earlier, this feature is used to restrict multicast packets from non-controllers to only reach the controllers in the network. In line (2), the variable *controllers* is populated by fetching the list of devices identified as controllers from the policy data base. In lines (4-10), we create the openflow action buckets for each device in that list. Line (7) ensures that multicast packets are sent out of the WLAN port as well if the source is a wireless client.

The *PACKET_IN_HANDLER* function receives every packet flow event from the switch and then extracts the *src, in_port, dst* from the packet headers. Lines (13 - 20) essentially convey our policy that messages originating from controllers should be allowed to reach all connected devices, but messages from non-controllers should only be limited to controllers in the network. Line (22 - 23) checks for unicast communication between two non-controllers and then drops the flow. Line (24) installs an openflow rule on the switch with the given set of desired actions for all future messages that match the tuple (src, dst, in_port). This rule in the data plane of the switch avoids any future packet_in event to be forwarded to the controller for a decision.

5 IMPLEMENTATION

We implement *Hestia* on top of a commodity wireless access point [37] running an open source firmware which supports Open vSwitch and the OpenFlow SDN protocol. This section describes some key components in the implementation, design challenges faced, and how we address these limitations in our design.

Router firmware: Since our approach requires an AP that allows custom firmware and software to be run on the network, we use one that is supported by OpenWRT. For our use case we built a custom OpenWrt firmware with Open vSwitch and Hostapd pre-compiled, as the default image for our router did not support wireless authentication modes. The default configuration uses a Linux bridge to bridge the wireless and ethernet interfaces into a single bridge interface. We removed the default Linux bridge and separately added an Open vSwitch bridge. The bridge was then configured to use a dedicated system connected to the access point as the SDN controller in the network. We use RYU as our SDN framework to implement Hestia on the controller.

AP Isolation Mode: In normal APs, the wireless interface behaves like a bridge between the wireless clients that are connected to it. If a frame originating from a wireless client on the network is destined for another client connected to the same wireless interface, then the frame is sent straight to the other client without going through the actual network stack of the AP. Most of the IoT device exploits, as identified by Acar et al [1], use this method of communication in wireless networks to their advantage. To combat this, our implementation utilizes the wireless isolation (or AP isolation or peer-to-peer blocking) mode that is available in standard APs. This mode forces the bridge to send all frames to the upper network stack, and denies wireless clients from communicating between each other by not forwarding wireless frames between them. We interpose Hestia's policy enforcement on top of the AP isolation mode using the primitives of off-the-shelf software-defined WiFi networks [25].

Forwarding to ingress port: With AP isolation mode enabled, all the network traffic between wireless devices is visible to the AP. However, in default isolation settings the switch forwards the packet to all other ports but the one from where the packet came, preventing clients on the same interface (WLAN) to communicate with each other. In order to let wireless clients communicate with each other and still allow all the frames to be passed through the higher network stack, we need to let the openflow switch explicitly send packets to the ingress port when the communication is between two wireless clients. We solve this by using an additional openflow action, `OFPActionOutput(ofproto.OFPP_IN_PORT)`, for all such cases to allow the packet to be sent back on the ingress port.

6 EVALUATION

We now evaluate *Hestia* by addressing the effectiveness of its policies in achieving least privilege, the need and appropriateness of its security guarantees, and measuring its performance for smart home deployment.

6.1 Policy Evaluation

Hestia is designed to introduce the principle of least privilege in network access for smart home devices. We evaluated Hestia's policies against a large data set of IoT devices and present our findings in this section. Alrawi et al [2] evaluated over 50 devices to provide a systematized study on the security of home-based IoT deployments. At the time of writing of this paper their data set provided the largest number of smart home devices deployed on the same network. The data set included 46 labelled smart home devices, and their generated traffic during a manual assessment of each device. They recorded the device interactions on the network at 5 minute intervals over a period of 10 days. We use the network PCAPs from their study to validate our hypothesis for the policy design, that a distinction of controllers and non-controllers is trivially sufficient to achieve least privilege policy of network access in smart homes.

6.1.1 Process of Evaluation. In order to test our hypothesis, we first manually classified the smart home devices in their data set into controllers and non-controllers based on the criteria of control interface introduced earlier. Our classification was inspired to mimic the understanding of an average smart home user, and thus we performed this distinction based on just the information obtained through the manufacturer website about the functionality of these devices. For instance, if a smartTV has Alexa integrated and the manufacturers website details that it can be used with other devices in the network then we identify it to be among the controllers in the network. Additionally in order to ascertain the triviality of this classification for an end user, we made sure that such information about the device was accessible directly from the manufacturers landing page for their device.

Control Interface	No. of Devices	Category
None	26	non-controllers
Voice Assistant	10	controllers
Remote Control Hub	9	controllers
Home Router	1	controllers

Table 1: Device categorization

Table 1 shows the number of devices that were classified as controllers and non-controllers in the data set and the feature because of which the decision was made. We identified a total of 20 out of the 46 labelled devices to be the controllers in the network.

The next step was to filter the recorded network data for local network communication and evaluate our policy. We used scapy to filter all the local network packets from each pcap file and aggregated all device communications for each day of the recordings. Once we had the aggregated network communication data, we created a *src - dst* mapping of devices where a unique mapping was created on per day basis. The goal was to achieve an insight on which devices are able to send messages to each other (within each 24-hr period), and thus if we found at least one instance of a packet exchange between two sets of devices on the network then we recorded it in our mapping.

6.1.2 Findings. Using the *src - dst* mapping of device interactions obtained for each day, we found that there were a total of 426 device-to-device communications that took place over the 10 day period. Since we created these mappings on a day to day basis, these include repeated instances of D2D interactions over several days. Validating against Hestia's generated policy based on identification of controllers further revealed that 54.69% of these device interactions were between devices that we had identified to be acting as the controllers in the network. We also found that 45.07% of connections occurred between controllers and non-controllers preserving our hypothesis that non-controllers need controllers to function. All the network packets that followed either of these communication patterns were allowed by default with Hestia's policies.

Anomaly: It also identified and blocked a network exchange, a communication between two non-controllers in the network. Upon further investigation of the packet traces relating to this anomaly, we found that the packet exchange occurred between a D-Link DCS5009L camera and a Belkin Netcam. What is more interesting is the fact that there were only 2 packets that were exchanged between these two devices. From accessing the payload of these packets, we identified that this was a UPnP device discovery initiated by the D-Link camera requesting the device details of Belkin Netcam. This is an inspiring finding because we aim to address exactly this in our threat model. Hestia prevents such service discovery by (potentially) malicious devices on the network thereby mitigating device compromises like Belkin Netcam's UPnP command injection attack [47] by another malicious device on the local network.

6.1.3 Result. Our findings suggest that the network communication policies generated by Hestia accurately portray ideal device communication. Additionally, our results suggest that these policies are also able to mitigate potential misuse of network access by limiting the service discovery and broadcast packets to only the required set of devices. This validates our hypothesis that a least privilege policy of network access is indeed achievable with a trivial distinction of controllers in the network, and bolsters the application of *Hestia* as the default network based security measure for smart homes.

6.2 Security Guarantees

6.3 Performance Measurement

We aim to provide *Hestia* as a substitute to the standard access point in a home network, thus in this section we empirically demonstrate the performance of our implementation against a standalone router.

Since Hestia affects every device connection that traverses through the router, it is important that it does not introduce any significant overhead for the users' devices. To evaluate Hestia's performance for device-to-device communications, we measure both first-packet latency and overall throughput. As the policy check is performed by the controller only at the first instance of a packet and a data flow rule is installed based on the policy decision for the future communications, we expect the performance overhead to be minimal.

6.3.1 First-packet Latency. To measure the first-packet latency, we sent various ICMP requests between two laptops connected to the wireless access point, with both Hestia disabled (standard AP) and enabled with different configurations – both laptops categorized as controllers, and only one as controller. Additionally, we performed these experiments while using the 5GHz radio configured to use a channel with minimum overlap with other wireless signals, thus ensuring minimal overhead from wireless interference. We evaluated first packet latency for four different categories of wireless traffic, which are commonly used by IoT devices, namely direct communication, broadcast, service discovery, and service advertisement. The connection between the devices

Figure 4 shows a comparison of the average round trip time for communication between these devices.

7 RELATED WORKS

Our work touches several areas of prior research, which we have divided into three broad categories. First, we re-iterate our threat model for smart homes by discussing studies on IoT device compromises and the security of smart home platforms. Next, we discuss the current measures being taken by the industry and security researchers in providing a defense mechanism for smart homes. Finally, we present access control mechanisms proposed for IoT devices and provide insights on the need of a network based access control for smart homes.

7.1 Internet of (vulnerable) Things

Home automation and connected home environments have become an irresistible trend for both manufacturers and consumers. However, more often than not, features tend to take priority over security in these devices. Several security flaws have been identified by researchers in these off-the-shelf devices for smart home. It has been shown that some of these devices mistrust other devices on the same LAN, thus leaving them vulnerable to a local adversary [20, 46]. Several incidents of device compromises include light bulbs performing DoS attacks [29], attackers unlocking smart door locks using ZWave protocol vulnerabilities [22], and directed attacks on baby monitors from the internet [42, 56, 58]. Denning et al. [17] presented a taxonomy of IoT attacks including threats to smart home devices like cameras, microphones, motion sensors, door locks, HVAC air pressure sensors, thermostats, washing machines, and health care technologies. It seems like most of these devices tend to suffer due to control signals emerging from unauthenticated sources. For instance, researchers have found that an adversary can control a smart TV with a speaker playing synthetic voice commands [39]. Ho et al. [30] demonstrate an attack on Bluetooth smart lock, similar to the attack on ZWave smart lock [22], where improper trust allows attackers to open locked doors. Additionally, implementation flaws by users can also lead to severe attacks against smart homes as demonstrated by Oluwafemi et al. [45], where the compromise of a networked device can even affect non-networked devices such as compact fluorescent light bulbs. HoMonit [69] compared the SmartApps activities as infered from side-channel analysis of encrypted wireless traffic against their expected behaviour as dictated in their source code, and found 60 misbehaving apps which conducted event-spoofing attacks.

The threat model for our implementation is based on these attack scenarios where the compromise of one connected device may affect other devices in the same network. Our work addresses these issues with smart home devices by providing a default policy of network isolation that can mitigate device compromises in a smart home. Consequently, we do not intend to prevent users from connecting potentially vulnerable devices to their network but instead provide a scalable and effective measure of isolating network communications between these IoT devices, which in turn would prevent device comprises in the local network.

7.2 Smart Home defenses

Several different methods have been proposed to defend the communication between devices in a smart home. We discuss these methods in the following categories.

Authentication Schemes: In order to control communications between IoT devices, authentication schemes [27] have been tailored for resource constrained devices. Zenger et al. [68] proposed a vicinity based pairing mechanism which delegates trust from one node to another based on physical proximity. However, these mechanisms require additional implementation on all devices of the system and do not consider the trivial problem of the existence of legacy devices in the network.

Behavioral security measures: Fully automated techniques have also been proposed to identify malicious communications within the smart home network. These mechanisms depend on intrusion detection systems tailored for IoT scenarios [48]. IoT Sentinel [40] is such a system that identifies the types of devices connected to an IoT network and enable autonomous enforcement of rule-based-control policies. But the main limitation of their work is that they enforce these policies in three pre-determined isolation levels, *strict*, *restricted*, and *trusted*. This greatly affects the scalability of their system for increasing number of smart home device functionality. IoTSense [9] presents another such behavioral fingerprinting method which can then be used to train a behavioral classifier to identify abnormal behavior in devices as demonstrated in IoTurva [24]. But a drawback in using such de-centralized behaviour based policy generation is that the ground truth for behavior classification is obtained through crowd sourcing or by analyzing the CVE database. This methodology is not really a dependable measure for users, and will in more cases than not cause limitations in the device connectivity in smart homes.

Network-level security: This security measure is routinely used in enterprise networks as an addition to protection software installed in clients. Considering the heterogeneity of devices in smart homes, network-level security is even more relevant. The privacy control mechanism proposed in [55] uses SDN-enabled switches of the ISP to dynamically block/quarantine devices, based on their network activity and on the context within the house such as time-of-day or occupancy level. However their implementation assumes that ISPs have complete visibility of the subscribers household devices, which raises a lot of privacy concerns as discussed in [6]. Preliminary prototype envisioned for IoTSec [67] demonstrates various ideas of a network based abstraction for IoT devices and shows its promises in implementation by providing security gateways for insecure devices and by enforcing policies for cross-device interactions. Thereby reiterating the importance of achieving an efficient network abstraction in smart homes. Their design principles are the most similar to our approach in concept however, our network abstraction provides a much more simplistic architecture and doesn't have the trust issues associated with crowd sourced repositories for policy enforcement. Recently certain commercial products have also been announced which address the security of IoT devices in a smart home. One such product is the Sense router from F-Secure [18]. However, it is only focused on traditional protection technologies like anti-virus capabilities, blocking of botnet controllers and preventing devices from accessing malicious websites [19].

IDS and Firewall: Several works on intrusion detection systems (IDS), personal and application firewalls [4, 11, 15, 32] focus either solely at the host or at a network node. IDIoT [8] claims to provide a network security policy enforcement framework for IoT devices, but their design only creates policies on the gateway system monitoring device to cloud endpoint communications. Additionally, these traditional mechanisms remain oblivious to device-to-device communications in wireless networks.

The policy design in *Hestia* aims to provide device-to-device granular policies like IoT-Sentinel but doesn't depend on any third party to make these decisions. Hestia provides the smart home owners with a scalable and easy to configure policy framework to employ a least privilege policy in their home network.

7.3 Access Control in Smart Homes

Ur et al. [62], in their study of access control in commercial smart devices, found that all these devices lack the mechanisms of access monitoring therefore making users incapable of identifying who has accessed their devices. This issue has been partially addressed with user-based access control policies [26, 34] which envision a policy enforcement for different types of network users (owners, spouses, maids, neighbours). Some recent work more relevant to our design, which provide device specific access control are mostly either a permission-based control on appified platforms, or situation based control.

Permission-based access control: Since smart home platforms are accompanied by appified platforms, permission-based access control has received a lot of attention by the security research community. In the meantime, researchers have ventured into identifying vulnerabilities in the emerging IoT frameworks that enable building apps to communicate with smart home devices and wearables [49]. Evidently so, researchers have demonstrated [54] how malicious apps running on a users phone could exploit the lack of security mechanism in UPnP protocol to scout home network for IoT devices and expose them at will to external attacks. To address these kind of exploits, researchers have focused on providing access control mechanisms for malicious apps running on a smart phone in a smart home. SmartAuth [61] provides users with an authorization user interface to bridge the gap between the functionalities explained to the user and the operations the app actually performs. The security frameworks proposed in FlowFence [21] provide a sandbox mechanism to operate on sensitive data by forcing applications to declare their intended data flow patterns. HanGuard [16] addresses this issue through SDN-driven data protection by running a user-space monitor on the users phone that detects authorized application access to IoT devices. ContexIoT [31] provides a context-based permission system for appified IoT platforms, with access control at the inter-procedure and data flow levels. While these approaches promise enhanced security for accessing IoT devices, a key observation lies in their adversary model which is focused on the situation where a malicious app is installed on a smart phone device. However, as covered in the earlier section on IoT vulnerabilities, we need to consider unpatched devices with vulnerable firmwares in our adversary model as well.

Situational access control: Few works have addressed access control in IoT based on a situation - like "user is at home" - that can only be tracked using multiple devices. Schuster et al. [51] provide a system which uses environmental situation oracles (ESOs) which encapsulates how a situation is sensed, inferred, or actuated. This methodology has been widely studied for Android platforms and other mobile OSes [10, 14, 41, 66]. But the critical distinction between IoT and mobile OSes is that access control in the IoT is fundamentally decentralized. While ESOs seems like a promising way of incorporating access control in the IoT stack, it needs a configuration change on all existing devices and automation systems. It might be an access control that we deserve, but not the one we need right now.

We posit the design goals envisioned in [67], that rather than chase the hopeless goal of IoT devices that are secure-by-construction, more pragmatic solutions that incorporate better network abstractions need to be the key vantage point for enforcing security policies in smart homes. Our design of Hestia tries to do exactly that, by providing a network based security measure for smart homes which can act as a default least privilege policy and also providing users with flexible design configuration.

8 CONCLUSION

Smart Home devices have a bleak reputation regarding their security practices. Current best practices for securing IoT device compromises in a smart home are not practically manageable for end-users. In this work we present *Hestia*, a building block of network based security mechanism introducing the principle of least privilege for smart home network access. We leverage the characteristic feature of IoT devices being designed for specific purposes to distinguish devices which act as controllers in the network. We use this classification of connected devices to create a local network access control policy for device communication which approaches the principle of least privilege. We provide an implementation of Hestia, using SDN primitives and the least privilege policies as OpenFlow rules showcasing its abilities to act as a default network based security system for IoT devices. Through these efforts we demonstrate the need and appropriateness of introducing network based access control in smart homes.

REFERENCES

[1] Gunes Acar, Danny Yuxing Huang, Frank Li, Arvind Narayanan, and Nick Feamster. 2018. Web-based attacks to discover and control local IoT devices. In *Proceedings of the 2018 Workshop on IoT Security and Privacy*. ACM, 29–35.
[2] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. SoK: Security Evaluation of Home-Based IoT Deployments. In *SoK: Security Evaluation of Home-Based IoT Deployments*. IEEE, 0.
[3] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the Mirai Botnet. In *USENIX Security Symposium*.
[4] Guido Appenzeller, Mema Roussopoulos, and Mary Baker. 1999. User-friendly access control for public network ports. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Vol. 2. IEEE, 699–707.
[5] Apple. 2018. Apple HomeKit. (2018). Retrieved January 2, 2019 from https://www.apple.com/ios/home/
[6] Noah Apthorpe, Dillon Reisman, Srikanth Sundaresan, Arvind Narayanan, and Nick Feamster. 2017. Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic. *arXiv preprint arXiv:1708.05044* (2017).
[7] ASUS. 2018. ASUS Router App. (2018). Retrieved January 5, 2019 from https://play.google.com/store/apps/details?id=com.asus.aihome&hl=en
[8] David Barrera, Ian Molloy, and Heqing Huang. 2017. IDIoT: Securing the Internet of Things like it's 1994. *arXiv preprint arXiv:1712.03623* (2017).
[9] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. 2018. IoTSense: Behavioral Fingerprinting of IoT Devices. *arXiv preprint arXiv:1804.03852* (2018).
[10] Sven Bugiel, Stephen Heuser, and Ahmad-Reza Sadeghi. 2013. Flexible and fine-grained mandatory access control on android for diverse security and privacy policies. In *Presented as part of the 22nd {USENIX} Security Symposium ({USENIX} Security 13)*. 131–146.
[11] William R Cheswick, Steven M Bellovin, and Aviel D Rubin. 2003. *Firewalls and Internet security: repelling the wily hacker.* Addison-Wesley Longman Publishing Co., Inc.
[12] Catalin Cimpanu. 2018. Over 65,000 Home Routers Are Proxying Bad Traffic for Botnets, APTs. (2018). Retrieved January 2, 2019 from https://www.bleepingcomputer.com/news/security/over-65-000-home-routers-are-proxying-bad-traffic-for-botnets-apts/
[13] CNET. 2015. Google ties Android to the smart home with Brillo and Weave. (2015). Retrieved March 28, 2018 from https://www.cnet.com/news/google-announces-brillo-at-io-developers-conference/
[14] Mauro Conti, Vu Thien Nga Nguyen, and Bruno Crispo. 2010. Crepe: Context-related policy enforcement for android. In *International Conference on Information Security*. Springer, 331–345.
[15] Manuel Crotti, Maurizio Dusi, Francesco Gringoli, and Luca Salgarelli. 2007. Traffic classification through simple statistical fingerprinting. *ACM SIGCOMM Computer Communication Review* 37, 1 (2007), 5–16.
[16] Soteris Demetriou, Nan Zhang, Yeonjoon Lee, XiaoFeng Wang, Carl A. Gunter, Xiaoyong Zhou, and Michael Grace. 2017. HanGuard: SDN-driven Protection of Smart Home WiFi Devices from Malicious Mobile Apps. In *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '17)*. ACM, New York, NY, USA, 122–133. DOI: https://doi.org/10.1145/3098243.3098251
[17] Tamara Denning, Tadayoshi Kohno, and Henry M. Levy. 2013. Computer Security and the Modern Home. *Commun. ACM* 56, 1 (Jan. 2013), 94–103. DOI: https://doi.org/10.1145/2398356.2398377
[18] F-Secure. 2018. SENSE Security Router. (2018). Retrieved January 5, 2019 from https://www.f-secure.com/en_US/web/home_us/sense
[19] F-Secure. 2018. What are the current projection features for f-secure sense? (2018). Retrieved January 5, 2019 from https://community.f-secure.com/t5/F-Secure-SENSE/What-are-the-current-security/ta-p/82972
[20] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security analysis of emerging smart home applications. In *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 636–654.
[21] Earlence Fernandes, Justin Paupore, Amir Rahmati, Daniel Simionato, Mauro Conti, and Atul Prakash. 2016. FlowFence: Practical Data Protection for Emerging IoT Application Frameworks.. In *USENIX Security Symposium*. 531–548.
[22] Behrang Fouladi and Sahand Ghanoun. 2013. Honey, I'm home!! - Hacking Z-Wave Home Automation Systems. (2013). Retrieved March 28, 2018 from http://www.blackhat.com/us-13/briefings.html#Fouladi
[23] Open Connectivity Foundation. 2018. AllJoyn Open Source Project. (2018). Retrieved January 2, 2019 from https://openconnectivity.org/developer/reference-implementation/alljoyn
[24] Ibbad Hafeez, Aaron Yi Ding, and Sasu Tarkoma. 2017. IOTURVA: Securing Device-to-Device (D2D) Communication in IoT Networks. In *Proceedings of the 12th Workshop on Challenged Networks*. ACM, 1–6.
[25] Seppo Hätönen, Petri Savolainen, Ashwin Rao, Hannu Flinck, and Sasu Tarkoma. 2016. Off-the-Shelf Software-defined Wi-Fi Networks. In *Proceedings of the 2016 ACM SIGCOMM Conference (SIGCOMM '16)*. ACM, New York, NY, USA, 609–610. DOI: https://doi.org/10.1145/2934872.2959071
[26] Weijia He, Maximilian Golla, Roshni Padhi, Jordan Ofek, Markus Dürmuth, Earlence Fernandes, and Blase Ur. 2018. Rethinking access control and authentication for the home internet of things (iot). In *27th USENIX Security Symposium (USENIX Security 18). USENIX Association, Baltimore, MD*. 255–272.
[27] Jose L Hernandez-Ramos, Marcin Piotr Pawlowski, Antonio J Jara, Antonio F Skarmeta, and Latif Ladid. 2015. Toward a lightweight authentication and authorization framework for smart objects. *IEEE Journal on Selected Areas in Communications* 33, 4 (2015), 690–702.
[28] Jim Hill. 2015. The smart home: a glossary guide for the perplexed. (2015). Retrieved January 2, 2019 from https://www.t3.com/features/the-smart-home-guide
[29] Kashmir Hill. 2015. This guys light bulb performed a DDoS attack on his entire smart house. (2015). Retrieved March 28, 2018 from https://splinternews.com/this-guys-light-bulb-performed-a-dos-attack-on-his-enti-1793846000
[30] Grant Ho, Derek Leung, Pratyush Mishra, Ashkan Hosseini, Dawn Xiaodong Song, and David A. Wagner. 2016. Smart Locks: Lessons for Securing Commodity Internet of Things Devices. In *AsiaCCS*.
[31] Yunhan Jack Jia, Qi Alfred Chen, Shiqi Wang, Amir Rahmati, Earlence Fernandes, Zhuoqing Morley Mao, Atul Prakash, and Shanghai JiaoTong Unviersity. 2017. ContexIoT: Towards Providing Contextual Integrity to Appified IoT Platforms.. In *NDSS*.
[32] Paul Judge and Mostafa Ammar. 2002. Gothic: A group access control architecture for secure multicast and anycast. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Vol. 3. IEEE, 1547–1556.
[33] Issac Kelly. 2012. Hacking the WeMo WiFi switch Part 1. (2012). Retrieved March 28, 2018 from https://www.issackelly.com/blog/2012/07/30/wemo-hacking
[34] Tiffany Hyun-Jin Kim, Lujo Bauer, James Newsome, Adrian Perrig, and Jesse Walker. 2010. Challenges in Access Right Assignment for Secure Home Networks.. In *HotSec*.
[35] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. 2015. Software-defined networking: A comprehensive survey. *Proc. IEEE* 103, 1 (2015), 14–76.
[36] Eliot Lear, Ralph Droms, and Dan Romascanu. 2018. *Manufacturer Usage Description Specification*. Internet-Draft draft-ietf-opsawg-mud-25. IETF Secretariat. http://www.ietf.org/internet-drafts/draft-ietf-opsawg-mud-25.txt http://www.ietf.org/internet-drafts/draft-ietf-opsawg-mud-25.txt.
[37] Linksys. 2018. Linksys DDWRT 1900ACS. (2018). Retrieved March 28, 2018 from https://www.linksys.com/us/support-product?pid=01t34000003qSgQAAU
[38] Linksys. 2018. Linksys Smart Wifi App. (2018). Retrieved January 5, 2019 from https://www.linksys.com/us/linksys-app/
[39] Benjamin Michéle and Andrew Karpow. 2014. Demo: using malicious media files to compromise the security and privacy of smart TVs. *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)* (2014), 1–2.
[40] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N. Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. 2016. IoT Sentinel: Automated Device-Type Identification for Security Enforcement in IoT. *CoRR* abs/1611.04880 (2016). arXiv:1611.04880 http://arxiv.org/abs/1611.04880
[41] Mohammad Nauman, Sohail Khan, and Xinwen Zhang. 2010. Apex: extending android permission model and enforcement with user-defined runtime constraints. In *Proceedings of the 5th ACM symposium on information, computer and communications security*. ACM, 328–332.
[42] NBCNews. 2014. Man Hacks Monitor, Screams at Baby Girl. (2014). Retrieved March 28, 2018 from https://www.nbcnews.com/tech/security/man-hacks-monitor-screams-baby-girl-n91546
[43] Netgear. 2018. Netgear Genie. (2018). Retrieved January 5, 2019 from https://play.google.com/store/apps/details?id=com.dragonflow&hl=en_US
[44] Matt Noble. 2013. Wemo Hacking. (2013). Retrieved March 28, 2018 from http://mattenoble.com/2013/08/07/wemo-hacking/
[45] Temitope Oluwafemi, Tadayoshi Kohno, Sidhant Gupta, and Shwetak Patel. 2013. Experimental Security Analyses of Non-Networked Compact Fluorescent Lamps: A Case Study of Home Automation Security. In *Proceedings of the LASER 2013 (LASER 2013)*. USENIX, Arlington, VA, 13–24. https://www.usenix.org/laser2013/program/oluwafemi
[46] Paul. 2013. Breaking and Entering: Hackers Say Smart Homes Are Easy Targets. (2013). Retrieved Jan 5, 2019 from https://securityledger.com/2013/07/breaking-and-entering-hackers-say-smart-homes-are-easy-targets/
[47] phikshun. 2018. Belkin Netcam HD UPnP Command Injection - Github. (2018). Retrieved January 5, 2019 from https://gist.github.com/phikshun/9984624

[48] Shahid Raza, Linus Wallgren, and Thiemo Voigt. 2013. SVELTE: Real-time intrusion detection in the Internet of Things. *Ad hoc networks* 11, 8 (2013), 2661–2674.

[49] Samsung. 2018. Samsung SmartThings. (2018). Retrieved March 28, 2018 from https://www.samsung.com/us/smart-home/smartthings/

[50] Bruce Schneier. 2014. Internet of Things is wildly insecure - and often unpatchable. (2014). Retrieved March 28, 2018 from https://www.wired.com/2014/01/theres-no-good-way-to-patch-the-internet-of-things-and-thats-a-huge-problem/

[51] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. 2018. Situational Access Control in the Internet of Things. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1056–1073.

[52] AT&T Services. 2018. Smart Home Manager. (2018). Retrieved January 5, 2019 from https://play.google.com/store/apps/details?id=com.att.shm&hl=en_US

[53] Sergey Shekyan and Artem Hartutyunyan. 2013. Watching the watchers: hacking wireless IP security cameras. *HITB* (2013).

[54] Vijay Sivaraman, Dominic Chan, Dylan Earl, and Roksana Boreli. 2016. Smartphones attacking smart-homes. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 195–200.

[55] V. Sivaraman, H. H. Gharakheili, A. Vishwanath, R. Boreli, and O. Mehani. 2015. Network-level security and privacy control for smart-home IoT devices. In *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. 163–167. DOI:https://doi.org/10.1109/WiMOB.2015.7347956

[56] Ms. Smith. 2013. Eavesdropping made easy: Remote spying with WeMo Baby and an iPhone. (2013). Retrieved March 28, 2018 from https://www.csoonline.com/article/2225628/microsoft-subnet/eavesdropping-made-easy--remote-spying-with-wemo-baby-and-an-iphone.html

[57] Ms. Smith. 2014. 500,000 Belkin WeMo users could be hacked; CERT issues advisory. (2014). Retrieved March 28, 2018 from https://www.csoonline.com/article/2226371/microsoft-subnet/500-000-belkin-wemo-users-could-be-hacked--cert-issues-advisory.html

[58] Mark Stanislav and Tod Beardsley. 2015. Hacking IoT: A Case Study on Baby Monitor Exposures and Vulnerabilities. (March 2015). Retrieved March 28, 2018 from https://www.rapid7.com/docs/Hacking-IoT-A-Case-Study-on-Baby-Monitor-Exposures-and-Vulnerabilities.pdf

[59] Statista. 2018. Smart Home Market penetration in US. (2018). Retrieved January 2, 2019 from https://www.statista.com/outlook/279/109/smart-home/united-states

[60] TP-Link Technologies. 2018. TP-Link Tether. (2018). Retrieved January 5, 2019 from https://play.google.com/store/apps/details?id=com.tplink.tether&hl=en

[61] Yuan Tian, Nan Zhang, Yueh-Hsun Lin, XiaoFeng Wang, Blase Ur, XianZheng Guo, and Patrick Tague. 2017. Smartauth: User-centered authorization for the internet of things. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*. USENIX Association, 361–378.

[62] Blase Ur, Jaeyeon Jung, and Stuart Schechter. 2013. The current state of access control for smart devices in homes. In *Workshop on Home Usable Privacy and Security (HUPS)*. HUPS 2014.

[63] Vera. 2018. Vera, Smarter Home Control. (2018). Retrieved January 2, 2019 from https://getvera.com/controllers/vera3/

[64] Waqas. 2017. How A Coffee Machine Infected Factory Computers with Ransomware. (2017). Retrieved March 28, 2018 from https://www.hackread.com/how-a-coffee-machine-infected-factory-computers-with-ransomware/

[65] Waqas. 2018. Hackers attack Casinofis fish tank thermometer to obtain sensitive data. (2018). Retrieved March 28, 2018 from https://www.hackread.com/hackers-casinos-fish-tank-smart-thermometer-hack

[66] Primal Wijesekera, Arjun Baokar, Ashkan Hosseini, Serge Egelman, David Wagner, and Konstantin Beznosov. 2015. Android Permissions Remystified: A Field Study on Contextual Integrity.. In *USENIX Security Symposium*. 499–514.

[67] Tianlong Yu, Vyas Sekar, Srinivasan Seshan, Yuvraj Agarwal, and Chenren Xu. 2015. Handling a Trillion (Unfixable) Flaws on a Billion Devices: Rethinking Network Security for the Internet-of-Things. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks (HotNets-XIV)*. ACM, New York, NY, USA, Article 5, 7 pages. DOI:https://doi.org/10.1145/2834050.2834095

[68] Christian T Zenger, Mario Pietersz, Jan Zimmer, Jan-Felix Posielek, Thorben Lenze, and Christof Paar. 2016. Authenticated key establishment for low-resource devices exploiting correlated random channels. *Computer Networks* 109 (2016), 105–123.

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