# A Hub for IoT devices in the home

Authors: Shaun Burley, Jason Hong, Aayush Bhutani, Dhruva Kaushal.

## Abstract

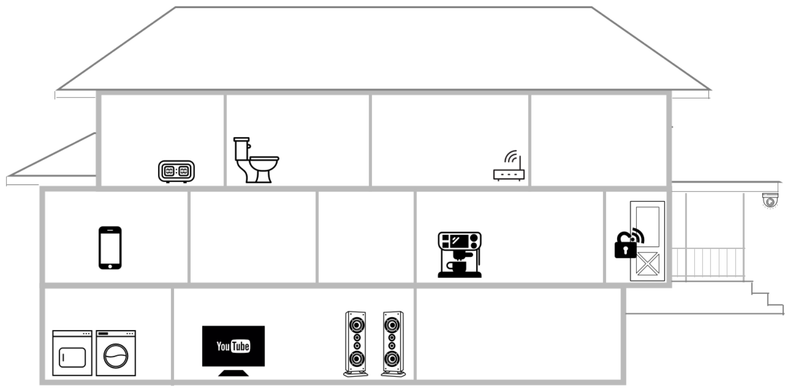
We offer a solution to some common security vulnerabilities of IoT devices in the home by encapsulating all device management to a single point of contact between the consumer and their devices: “the IoT Hub.” This “Hub” can offer new kinds of services that would help improve the security of these devices. In particular, we explore two security features. The first is Manufacturer Usage Descriptions that specify a whitelist of what sites a device will communicate with. The second is automated software updates, where a device can specify URLs for its software updates, with the hub periodically checking those locations for updates. These features are intended to be simple to deploy on existing and future IoT devices in the form of metadata that can be read in by the Hub via the device itself and a connection to the device manufacturer’s remote server. Complexity and common functionality is factored out from IoT devices and onto the Hub, analogous to the relationship between a smartphone’s OS and the individual apps on the smartphone.

We demonstrate a simple prototype of the two security features discussed above. Using mobile phone screens to represent the Hub interface and an interface for the device (in our case a connected coffee machine), we demonstrate the process of adding new devices and fetching their usage descriptions, along with facilitating software updates for a particular device. We foresee our design being implemented directly on a wireless router that can host the Hub’s interface and store the metadata for each device on the network. In this way, no hardware is needed besides the devices and the router. This design protects devices against the most common types of cybersecurity attacks, and offers some very exciting possibilities for making homes even smarter in the future. Moreover, it takes the accountability of security out of the hands of the end consumer and into the hands of the manufacturer and their security team.

----

## Introduction

The vision of IoT is rapidly becoming a reality due to advances in processors, sensing, displays, storage, wireless networking, and battery life. Two decades ago, computers were primarily large beige boxes that came with a monitor, keyboard, and mouse. Today, computers come as smartphones, tablets, smart toys, thermostats, fitness trackers, standalone webcams, health monitoring devices, electronic locks, smart mirrors, and more.



One rapidly growing market for IoT is consumer electronics in the home. However, there are a large host of challenges for making IoT work reliably in the home. How can we make it easy to manage dozens or hundreds of devices? How can we help people ensure the privacy and security of their devices, for example, making sure software is up to date, tracking what devices and apps are doing, pinpointing unusual behaviors, and managing credentials?

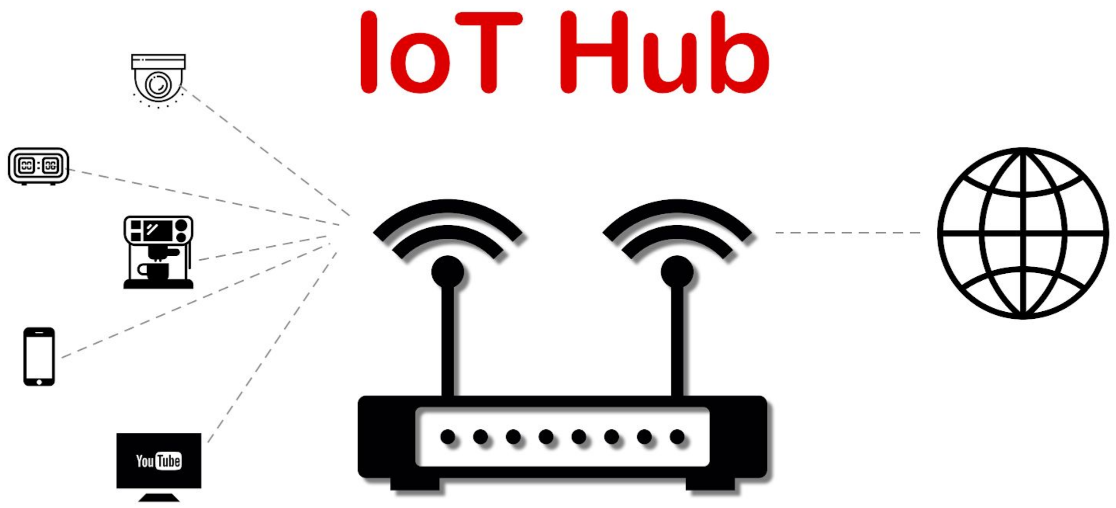
There are two major design constraints for IoT. The first is scale. It’s possible to manually manage a handful of devices, but unrealistic for dozens or hundreds of devices, especially if they all have different user interfaces. The second is the wide range of device capabilities. People will likely have a few high-end devices that have a lot of CPU, storage, networking, and sensing capabilities, but will have many more mid- and low-end devices that are highly constrained in terms of compute power, networking, storage, and battery life. These mid- and low-end devices will have limited functionality for management and security.

In this paper, we sketch out our vision for an IoT Hub to help people manage their devices. This hub acts as a centralized device for adding, managing, monitoring, and securing devices in a home. In some ways, this hub is like a network firewall, in that we intend that IoT devices go through our hub rather than connecting directly to the Internet. The hub also facilitates the deployment of IoT devices by offering a common platform and a suite of useful services important for mid- and low-end IoT devices, assists with the rapid deployment and evolution of new kinds of services, and presents new ways of connecting devices together in a seamless manner.

Here, we focus on two key pieces of functionality. The first is Manufacturer Usage Descriptions that devices have that specify a whitelist of what sites that device will communicate with. The second is automated and software updates, where a device can specify URLs for its software updates, with the hub periodically checking those locations for updates. We describe the design and implementation of our ideas below.

**When IoT devices scale, so does cognitive load**
Encapsulating connected devices in the home into a single endpoint does not disenfranchise the consumer so they no longer have control over their devices; instead it unifies control of their devices, thereby reducing the cognitive load of keeping track of multiple modes of interacting with different devices. When there are only a few devices in the home, individual device management may be tractable, but when it scales to dozens or even hundreds of IoT products, it becomes quite a daunting task. In fact, Gartner predicts there will be over 20 billion IoT devices by 2020, and most of these will lie far in the background of our attention: RFID-enabled ID cards and badges, clothes, HVAC, digital light bulbs, smart toilets, etc. These kinds of devices will have very little computational resources, might have few (if any) software updates, and use a wide range of software and operating systems, which will make cybersecurity especially weak.

**Risks of weak IoT device security**
This ‘soft underbelly’ of the internet-connected world can and is being exploited by hackers who have begun to cause considerable damage over the last several years. Many vulnerabilities in IoT devices stem from two common weaknesses: 1) software that does not get updated regularly (if ever), and 2) easy-to-guess/default passwords. Both of these vulnerabilities give hackers access to internal settings that in turn allow them to a) hold user data for ransom, b) wrest control of the devices away from their owners, or c) form botnets that can lead to denial of service attacks, such as the Mirai DDoS attacks seen in October, 2016.



**IoT Hub**
Our general solution is to have an IoT Hub that makes it easy to add, manage, monitor, and secure devices in the home. The general design philosophy of the IoT Hub is that it can offer common and useful functionality so that IoT devices themselves do not have to implement it themselves. As an analogy, a smartphone operating system offers many useful services for managing apps, for example seeing how much data or battery life each app is using, or managing notifications. Analogously, we want the IoT Hub to offer services that can help manage devices, which can lead to simpler and more secure devices.

The IoT Hub also acts as a level of indirection for network traffic, so that devices are not connected to the Internet directly. This approach allows us to block suspicious traffic, as well as monitor activities of each device.

Each device is expected to have metadata that describes it, for example the name of the device, a picture of the device, the manufacturer, serial ID, OS version, software version, and so on. There are two specific pieces of metadata that we are interested in, namely a Manufacturer Usage Description (MUD) and a URL for software updates. This metadata is expected to be read in when the device is added to the IoT Hub. This metadata should also be digitally signed so that an IoT Hub can verify that the data has not been modified. Users can use a smartphone app to interact with the IoT Hub. Currently, this app lets users see what devices have been added to the IoT Hub, and get notifications as to when software updates are available for devices.

**A better approach to IoT security**
The better way to address both weaknesses comes from the oft-observed fact that consumers do not go to their devices to “do security” just as they do not go to their internet settings or web browser settings to “do security.” Studies have shown that the average consumer pays little heed to and dismisses many of the security and privacy warnings presented to them in web browsers without reading them. This problem is exaggerated with IoT products which are already designed to exist in the periphery of our attention - the user is prone to forget about the device’s existence, let alone its security and privacy settings.

We have two observations here. First, it does not make sense for all devices to be able to connect to any site on the Internet. Most devices will be relatively simple and thus only need access to a few web sites. For example, a smart toy might only need to contact its manufacturer’s website for new content, or a smart weight scale might only send updates to a health web site so that it’s owner can track their change in weight over time. Second, device manufacturers can help with security by specifying elements of a device’s behavior. This kind of specification can help with security, as it makes it easier for an IoT Hub to know if a device’s behavior should be allowed or not.

The IETF had working proposal on these Manufacturer’s Usage Descriptions (MUD), and we have incorporated a simplified form of this in our IoT Hub. The MUD is a document created by a manufacturer that contains metadata describing a particular device like a smart light bulb, and explicitly defines the intended uses and the unintended uses of the device. The MUD knows the light bulb is NOT a general-purpose computing machine but a device with a specific purpose (giving you control of lighting a room) and then limits its functionality accordingly (allowing it access to your phone and forbidding it to post on Twitter, for example). This metadata is separated into two sections: variables and functions, similar to the ‘attributes’ and ‘methods’ of a class of objects in a programming language like Java. The variables include static attributes like **model number** and **software update URL**, and dynamic attributes like **device name, current software version,** and **current battery level**. The functions could include actions like **turnOn**, **turnOff,** and **checkForUpdate**. Note that defining useful metadata for privacy and security is a major area of interest for us, given that these are two of the biggest concerns that consumers have expressed regarding IoT.

## Manufacturer’s Usage Description

```
Device Model: WS2801
Device Manufacturer: Adafruit
Picture: 345i34htigr.jpg
Instruction manual URL: http://www.adafruit.com
MUD URL: http://mud.adafruit.com
Software update URL: http://update.adafruit.com
…

Device Name: Shaun’s NeoPixels
Location: bedroom
Battery Level: 100%
Current software version: 3.1.4
Last used: 5/20/2017 8:45 pm
…

turnOn()
turnOff()
sendImg()
show()
setBatteryLevel()
…
```

**How does the Hub get access to a device’s MUD?**
Every device needs a way of pointing the Hub to the remote server that hosts the device’s metadata, and it needs to be secure. The way we suggest is through a small web server inside the device that stores a single URL that is served to the Hub the first time it is added to the network of IoT devices. Each device added makes a single file request to the Hub which then fetches the device’s MUD from the manufacturer’s website. The MUD is returned and stored in the Hub as a list of intended uses, including what types of access are permitted and what types are restricted. Later, when the device requests to connect to a remote server, this request goes first to the Hub, which determines whether this request is valid or not by checking the MUD in its database.

**Validating a device’s MUD request**
One major challenge of this design is security - how do we make sure that the URL stored in the device is valid and hasn’t been corrupted? The most straightforward solution would be to use the same signed certificates (with public/private key encryption) that web browsers such as Chrome use to validate the authenticity of websites. These certificates can be preloaded into the Hub software and when it receives a URL from a new device, it can check to make sure the URL it’s fetching matches one of its certificates. Furthermore, when a device communicates with the Hub, it needs to do it over a secure channel that cannot be observed by an interloper. We expect communications from devices to hubs to be encrypted using standard TLS/SSL encryption to secure the data in transit.

## Scalable, supportive, and secure

The MUD approach is different from existing cybersecurity approaches because it uses the subject matter expertise of device manufacturers and network security professionals to minimize the effort of securing IoT endpoints. There are many benefits to managing dozens of IoT devices through a Hub with access to MUDs, including easy scalability, legacy support for obsolete devices, and increased security by default.

### Scalability

The manufacturer must put in some work up-front developing the MUD for each of their devices, but once created, they will remain fairly static and when they're updated, those changes will trickle down to all hubs that have that URL. The current MUD specification dictates that 1) a manufacturer only need to maintain and distribute one XML file per device model, 2) the network management system (in our case the "Hub") does not need to retrieve the same file when subsequent devices with the same model number are added to the local network, and 3) the manufacturer specifies the frequency that device managers (our "Hub") look for updates to the MUD file.

### Legacy support

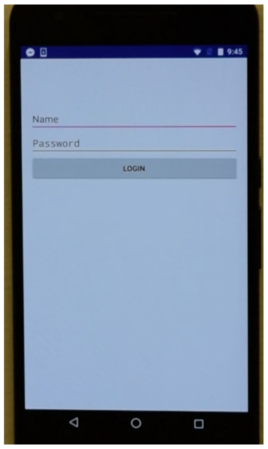
Devices that are no longer actively supported will still remain protected via the restrictions defined in the original MUD, or the latest version of the MUD that was downloaded to the Hub. This means a device with an obsolete set of permitted URLs will "break" safely: it will simply be unable to connect to the new server, rather than expose itself to third-party hacking attempts.

### Default security

Generally, it's better to start with no permissions and add a few, than to start with every permission and take away ALL but a few. Using the MUD-Hub combo, devices are much more secure even when no further action is taken by the end users. If the Hub is the sole point through which a device can gain access to the internet, then when it is onboarded, the Hub automatically fetches its MUD and sets up rules for the device's intended usage such that the consumers who don't not have security on their mind are still protected by default.

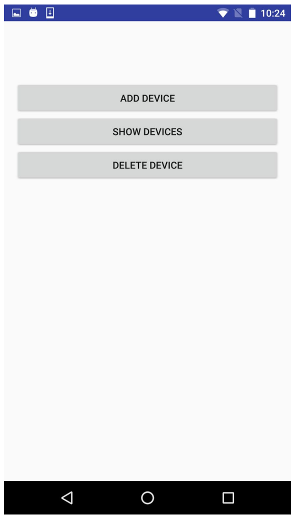
## A few examples

Let us now walk you through an example of how the IoT Hub and MUD work together to create a simple and straightforward way to manage the security of IoT devices in the home. As mentioned before, our design of the Hub exists on a wireless router that has the capacity to host the Hub's interface and database of metadata for each connected device. Similar to most wireless routers, the Hub's interface is interpreted as HTML pages rendered in a web browser on any device that is on the home's wireless network. Access to the Hub interface is controlled by a username and password:



**login screen**

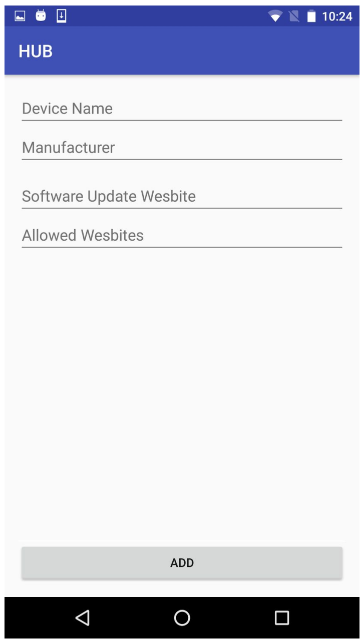
Once logged in, a user has options to add new devices, see a list of currently connected devices, and remove devices from the network.

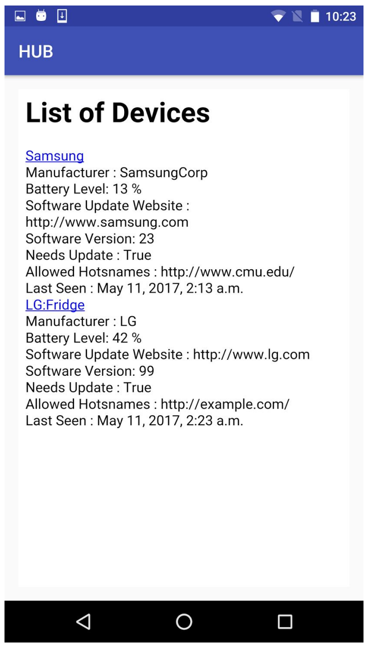


**home screen on Hub**

If a user wants to connect say, a new 'smart' refrigerator to their home network, they can onboard it onto the Hub by using its own interface to first connect it to their home wireless network using the normal network name (SSID) and password (WPA2, etc.). As soon as the connection is established, the Hub requests a URL for the refrigerator's usage description metadata (MUD) file and the device's server serves this URL to the Hub, who then fetches and downloads the file and adds the fridge's metadata to its database.

This is the onboarding process for a smart object that has its MUD URL preloaded into its firmware. For devices currently on the market or that do not have enough computing power to host a small web server with the MUD URL, the user can enter the URL for its usage description manually on the Hub's interface, which will then be fetched and parsed by the Hub and stored in the database:





List of Devices

Samsung
Manufacturer : SamsungCorp
Battery Level: 13 %
Software Update Website :
http://www.samsung.com
Software Version: 23
Needs Update : True
Allowed Hotsnames : http://www.cmu.edu/
Last Seen : May 11, 2017, 2:13 a.m.
LG:Fridge
Manufacturer : LG
Battery Level: 42 %
Software Update Website : http://www.lg.com
Software Version: 99
Needs Update : True
Allowed Hotsnames : http://example.com/
Last Seen : May 11, 2017, 2:23 a.m.

**Adding devices to the Hub**

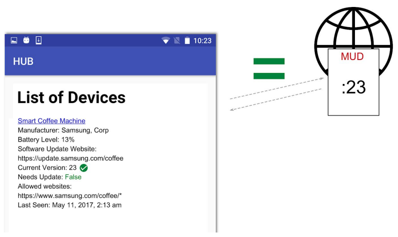
If the manufacturer hasn't provided a usage description for their device (this especially holds true for legacy devices or very small manufacturers) the user can enter information directly into the Hub's database, like the fields for accessible URLs, the device's model number, and its software version.

## Operation

Once a device has been added and its MUD is in the Hub's database, it will operate normally because the remote sites that it tries to connect to will be on the list of allowed URLs in its MUD. Additionally it will send a list of the functions it's performing and these will be added to the device's log on the Hub. If the device tries to contact a remote address that's not on that list, the Hub recognizes this and simply doesn't allow the request to go through. Theoretically, this also would also work in the reverse case, if an unrecognized remote address tries to connect to the device. At intervals the Hub will request the device send back the variables (or attributes) which are dynamic, such as current battery level, and update them in its database. In this way, users can keep track of the status of any device by simply looking at the Hub interface.

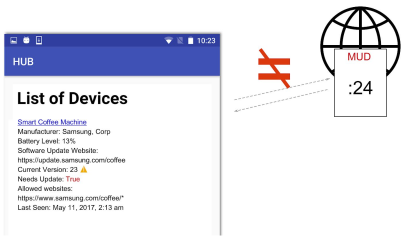
## Facilitating software updates

As mentioned previously, one of the dynamic variables for each device is its current software version. In order to make sure that each device has the most recent version installed, the Hub will periodically fetch a device's most recent MUD file from the manufacturer's website and check for differences between it and the version currently stored in the Hub. When it sees there is a newer software version available, it makes this known to the user and facilitates the update to whatever extent it can: a) it can alert the user whenever they are on the Hub's interface (asynchronous), b) it sends a push notification or other alert to the user (synchronous) as soon as it notices out-of-date software, or c) it actually does the update automatically and notifies the user of the completed update. This third way is the least intrusive and most secure, but it also takes the most agency from the user.



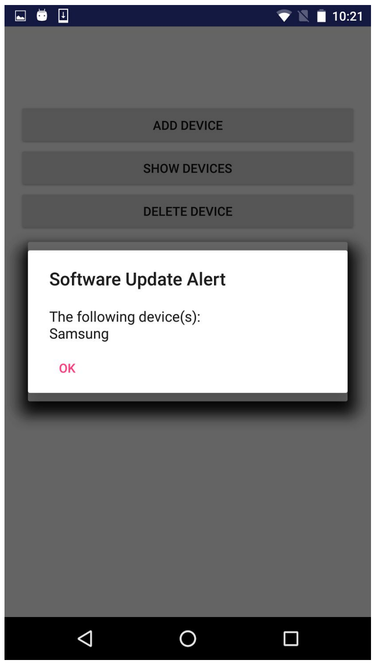
List of Devices

Smart Coffee Machine
Manufacturer: Samsung, Corp
Battery Level: 13%
Software Update Website:
https://update.samsung.com/coffee
Current Version: 23 ✓
Needs Update: False
Allowed websites:
https://www.samsung.com/coffee/*
Last Seen: May 11, 2017, 2:13 am



List of Devices

Smart Coffee Machine
Manufacturer: Samsung, Corp
Battery Level: 13%
Software Update Website:
https://update.samsung.com/coffee
Current Version: 23 ⚠
Needs Update: True
Allowed websites:
https://www.samsung.com/coffee/*
Last Seen: May 11, 2017, 2:13 am



Software Update Alert

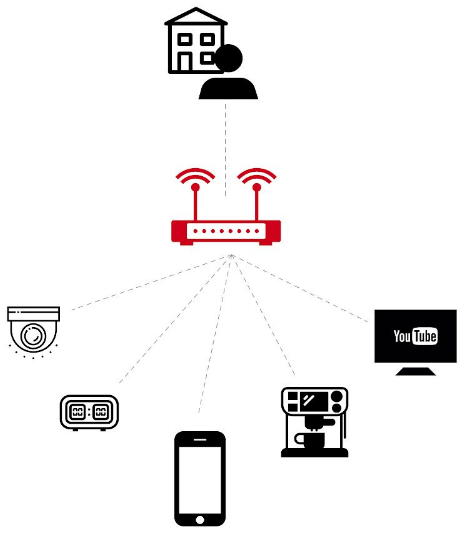
The following device(s):
Samsung

**Keeping the MUDs up to date**
It is during these periodic calls from the Hub to a device's MUD page that the list of allowed and restricted URLs are also updated, along with anything else that has changed in the way a device operates. This process offers device manufacturers incredible scalability. By updating one secure HTML page on their site, they can affect the operation of millions of devices in people's homes. This may sound like it's giving too much power to relatively few, but remember that manufacturers have an economic motivation to keep their devices operating flawlessly and securely, and they generally have more resources than individuals to hire teams of security professionals that can ensure device security and respond quickly to exposed vulnerabilities.

**Current implementation**
Our IoT Hub design has two features that are implemented in this first version: 1) adding a new device to the network and entering its MUD into the Hub's interface, and 2) alerting the user that a device on the network needs a software update. Currently, the user is required to update the software on the device manually after being alerted by the Hub, however in the future we plan to allow the user to authorize the Hub to update devices automatically and only alert the user that they've been updated successfully (or not).

For demonstration purposes, we've used mobile phone screens to represent the Hub interface as well as an interface on the device (in our case a connected coffee machine). This proof of concept is meant to show the feasibility of such a design as described in this paper along with the relative ease with which it is operated. It's important to note that a screen interface is not needed on the IoT device for our design to work; all that's needed for the onboarding process is a URL that points to the device's usage description online. Having phone screens also has the added benefit of making the device's connectivity visible; if you imagine a device like a coffee machine, it might not be clear that it's being blocked from accessing a forbidden website if there is no screen. Importantly, it would remain secure even if the user is unaware of failed attempts to hack into it.

Our demonstration with a 'smart' coffee machine starts by trying to operate it on the wireless network without adding it to the Hub. The device cannot access its manufacturer's website because the Hub does not recognize it as a device. Then we add it to the Hub and manually enter the the URL for the MUD. After this metadata is added, we show that the coffee machine can access the website it has been granted access to (in our case the CMU homepage) and nothing else (like facebook).



## Conclusion

In this paper we address the rise of IoT devices in the home and the challenges of keeping this network of connected devices secure as they scale to dozens or even hundreds of unique products. Right now, hackers are exploiting two key vulnerabilities of these IoT devices: 1) unpatched/out-of-date software, and 2) default/easy-to-guess passwords. These weaknesses allow hackers to fiddle with the internal settings so that they send collected data to 3rd parties or just become part of a botnet meant to attack a remote server. We offer a solution in the form of an ‘IoT Hub’ that encapsulates device management into one interface that serves as the single touchpoint between users and all their devices. Moreover, we pair this Hub with metadata produced by device manufacturers known as the Manufacturer’s Usage Description (MUD). This metadata defines the purpose of the device, including a set of permissions for accessing only a small number of URLs that it needs to operate. This works because unlike general purpose computing machines, IoT devices have limited computational power and are designed to have a very specific and often narrow purpose. The MUD document is what makes this purpose explicit and ensures it’s only used for that purpose.

Taking advantage of a usage description from a device’s manufacturer potentially mitigates security issues in several ways: 1) it substantially restricts the functionality of a device entering a network to those communications intended by the manufacturer, 2) it provides a means to scale network policies to the ever-increasing number and types of devices on a home network, and 3) it provides a means to address at least some vulnerabilities in a way that is faster than it might take to update entire systems. This latter point is particularly salient for legacy systems that are no longer supported by their manufacturer. Finally 4) it is an extremely affordable way to bolster the security of a global network of connected devices.

The metadata and functions that come with the MUD are important because, in conjunction with the IoT Hub, they are what will enable IoT devices to be easily integrated into a person’s home and connected with other devices. For example, using the metadata and functions, the IoT Hub can facilitate end-user programming of devices, check all devices if they have the latest software updates, offer a unified user interface to view and control devices, and have a single centralized feed as to what these devices are doing. This feed also makes it possible to do anomaly detection and to summarize the privacy and security-related behaviors of devices. These logs of device behaviors operating throughout a home might be useful beyond reducing the need for device management and bolstering security - they can serve as ground truth data for powerful machine learning algorithms that might lead to smarter and safer homes in the future.