A version of this work appears in *The First International Workshop on Pervasive Smart Living Spaces (PerLS 2017)*, a workshop at *IEEE PerCom 2017* [30]. This version appears as University of Washington Technical Report, UW-CSE-17-01-01.

Securing Vulnerable Home IoT Devices with an In-Hub Security Manager

Anna Kornfeld Simpson
University of Washington
aksimpso@cs.washington.edu

Shwetak N. Patel
University of Washington
shwetak@cs.washington.edu

Franziska Roesner
University of Washington
franzi@cs.washington.edu

Tadayoshi Kohno
University of Washington
yoshi@cs.washington.edu

***Abstract*—The proliferation of consumer Internet of Things (IoT) devices offers as many convenient benefits as it poses significant vulnerabilities. Patching or otherwise mitigating these vulnerabilities will be difficult for the existing home security ecosystem. This paper proposes a central *security manager* that is built on top of the smarthome's hub or gateway router and positioned to intercept all traffic to and from devices. Aware of the status of all devices in the home and of reported vulnerabilities, the security manager could intervene as needed to deter or alleviate many types of security risks. Modules built atop this manager could offer convenient installation of software updates, filter traffic that might otherwise exploit devices, and strengthen authentication for both legacy and future devices. We believe that this design offers the potential to increase security for smarthome IoT devices, and we encourage other researchers and regulators to explore and extend our ideas.**

I. Introduction

The Internet of Things (IoT) is rapidly becoming a reality. New IoT technologies are emerging at a rapid rate, with a forecast of 6.4 billion connected devices in 2016 and 20.8 billion in 2020 [14]. When considering homes alone, current estimates suggest that there are hundreds of millions of IoT devices deployed within homes in the United States [24].

These devices bring many benefits, but they also carry with them significant downsides, including potential computer security risks. In this work we propose a central security manager platform for home IoT devices and extend it with modules to increase security at various points of the vulnerability lifecycle.

Recent events show concrete examples of real and significant harm that results from the vulnerabilities in smarthome IoT devices. Perhaps the most publicized example is the recent use of compromised IoT devices to mount massive DDoS attacks from millions of devices — both against journalist Brian Krebs at 620 Gbps [5] and against DNS company Dyn at 1.2 Tbps, which resulted in major sites being inaccessible for several hours [29, 35]. Other potential harms include the compromise of IoT devices — either for use in ransomware attacks [20] or to cause physical harm to homes or residents (e.g., unlocking doors [12], popping lightbulbs [23], or even problems that prevent users from turning on the heat [4]).

An ideal solution could simply ensure the rapid patching of vulnerable devices. Unfortunately, software updates for IoT devices are easier said than implemented, for several reasons:

- *Missing software update capabilities:* Some of IoT devices may not be configured to receive software updates — they may keep forever the software that they are shipped with.
- *Devices outlive software updates:* Some IoT devices — like refrigerators and water heaters, which people buy with the expectation that they will last decades — might receive software updates for only a fraction of that lifetime, if their manufacturers choose to stop maintaining the device's software. In other cases, devices may be produced by startups that, after selling many devices, either go out of business or are purchased by another company, which then chooses to cancel the original product line (as well as maintenance on the already sold products). For example the Revolv device was acquired by Nest and shut down in May 2016 [15]. For their remaining lifetimes, these devices may remain vulnerable to external compromises.
- *Challenging to apply available updates:* Even if a software update is available for some classes of IoT devices, it may be challenging to apply. Some IoT devices require users to actively install updates: specifically, to remember that they have a specific IoT device in the home, notice that it has received an update or recall notice, and then go to that device to initiate a software update. Relying on users may not work — for example, thousands of Foscam-manufactured baby monitors with a remote-control vulnerability remained unpatched because buyers were unaware of Foscam's patch [16]. Even automatic updates, however, may be challenging to apply. Devices might need to restart to update their firmware, causing gaps in availability. Further undesirable consequences could ensue if manufacturers update the firmware on a light switch at night when the user is moving around the house or on a water heater when someone is showering.

Thus, IoT devices will inevitably have security vulnerabilities, and quickly patching those vulnerabilities will not always be straightforward. Therefore, we propose an approach to close the gap: instrumenting the home with a centralized, hub-based security manager that is positioned inside the home's gateway router to intercept all communications and protect vulnerable devices.

This security manager should be extensible, with modules that support varying security goals — for example, filtering traffic to vulnerable devices or strengthening weak device authentication. Such modularization lets us isolate the code for different security goals and defenses into separate components and lets different teams (possibly even from different organizations) develop modules independently. We envision the underlying security manager's business model resembling that of today's desktop anti-malware vendors. Our proposed modules can help improve security for IoT devices at several points after the discovery of a vulnerability: (1) after a vulnerability is discovered but no patch is available, (2) after a patch exists but is not yet applied, (3) during the application of a patch, and (4) in the case of compromise. Though our approach cannot eliminate all challenges at all of these stages, it can significantly raise the bar for security.

Contributions. We demonstrate the value of a hub-based security manager. In doing so, key contributions include:

1) Clearly defining goals for such a security manager across several stages of the device-vulnerability lifecycle.
2) Exploring an extensible hub architecture that supports individual modules for specific security goals.
3) Evaluating our proposals with several case study module implementations built atop a prototype of our security manager design.

Additionally, we provide lessons and recommendations for future hubs and smarthome IoT deployments. These lessons have the potential to inform the directions that industry and government take — for example, the U.S. National Telecommunications and Information Administration (NTIA) is running off a multi-stakeholder effort to improve the security of IoT devices [2], and the Federal Trade Commission (FTC) has announced a competition for IoT device security specifically focused on devices that currently exist in the market [32].

II. Related Work

IoT and Security. First, we look at the wider literature on consumer embedded device security to identify challenges and security goals for IoT. Researchers have identified vulnerabilities in cars [6, 17], medical devices (reviewed in [27]), augmented reality [26], drones [31] and GPS [22].

Denning et al. offer a series of threats and case studies in a 2013 CACM paper [8], and look specifically at household robots in [9]. Fernandes et al. discover multiple privilege escalation vulnerabilities in a common brand of home IoT devices [12], while Yu et al. propose decentralized middleboxes to prevent unapproved communication between IoT devices in the home (for example, if one device has been compromised) [36]. Fernandes et al. present a system that enforces data flow policies on IoT devices [13].

Protecting Vulnerable Devices. Numerous researchers have explored the software update space, but few have focused on IoT devices. The 2006 paper from Bellissimo et al. foreshadowed the importance of updates to the IoT [3]. Kuppusamy et al. note that solutions for PC updates are insufficient for other systems, such as cars [18], and Samuel et al. identify update integrity verification challenges that could be particularly difficult for embedded devices [28]. Researchers have also focused on the usability challenges of applying updates and patches [19, 34], particularly important in this space because many current devices require manual intervention to update.

Wang et al. proposed Shield [33] , which operates between the transport and application layers to detect traffic patterns that indicate the exploitation of a worm. We build on these ideas to address challenges specific to IoT.

III. Context, Goals, and Threat Model

Before discussing our proposed design, we elaborate on our motivation and goals, which we consider to be a core contribution of this paper.

A. Motivation and Context

Hubs and Routers: A Vantage Point for Defense. IoT platforms for smarthomes are rapidly evolving, with numerous industry efforts vying to establish a leadership position. Manufacturers place the controllers for their devices in the cloud, in the home, or in some combination thereof. For example, the 2015 edition of Samsung's SmartThings hub runs most apps in the cloud, but allows some apps to run in the home so the apps still function when the Internet is disconnected [1]. Our solution works with both local and cloud functionality.

For any IoT device without its own Internet connection — i.e., any IoT device without an internal cellular modem (or a connection to a device with a cellular modem) — the device's communications by definition *must* transit through some other Internet-connected device. For some devices, this transit point is not currently called a "hub": for a WiFi security camera or the Amazon Echo, it might be the home's WiFi router. Therefore, to include the maximum number of devices, we propose that our hub security manager be located within a device that has both gateway router and hub capabilities.

Additionally, since our goal is to protect vulnerable devices, we place our security mechanisms within the local network. For example, a cloud-based mechanism might not detect out-of-band attempts to compromise a device or have the vantage point to detect anomalous network activity emanating from the device. Hence, our vision is a centralized device within the home that is connected to all the home IoT devices and provides the network connectivity for those devices.

Existing and New Devices. Our proposal is to take *both* a short-term and a long-term vision on IoT security for vulnerable devices. For our long-term vision, we envision IoT devices designed to work with our proposed system. This vision is feasible if a major industry player adopts our approach. Nevertheless, we acknowledge that legacy devices — those built before our system becomes widely adopted — will still exist on the protected homes' networks. Thus, while our system cannot provide as much protection for legacy devices as it can new, compatible, devices, we still seek to significantly improve security for legacy devices.

B. Goals

Providing Flexibility and Functionality. Our design seeks to be sufficiently flexible to handle a diverse set of security goals. To do so, our security manager design must be extensible to new scenarios, devices, and information about existing devices.

Currently, when a device is vulnerable and not fixable, users can either leave it on, or, if they are aware of and concerned about the vulnerability, remove it from their network. Unfortunately, many home IoT devices do not function correctly without Internet connectivity. We believe that it is possible to offer intermediate options that curtail some

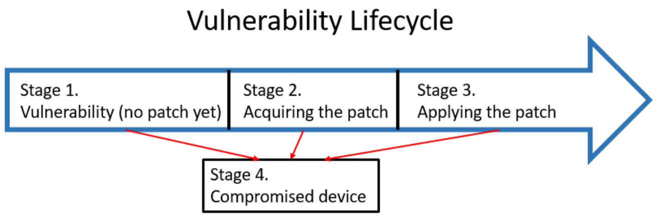

Fig. 1: **The Device-Vulnerability Lifecycle.** Stage 1: vulnerability discovery (no patch yet); stage 2: patch exists but not yet acquired; stage 3: patch application; and stage 4: compromised device.

functionality if necessary to protect the device, but do not completely disable it.

Mitigating Points the Device-Vulnerability Lifecycle. This section explores distinct moments in the lifecycle of a vulnerability for an IoT device and identifies threats and mitigations for these points. We consider the formulation of these security goals to be a contribution of this paper, and we encourage other researchers to extend our solutions to these goals.

Security risks emerge at different stages of the vulnerability lifecycle between vulnerability and patch as shown in Figure 1:

- Stage 1. Vulnerability discovered but no patch exists yet.
- Stage 2. Vulnerability discovered and patch available.
- Stage 3. During the application of a patch.
- Stage 4. If a vulnerable device has been compromised.

Stage 1: Vulnerable and No Patch Yet. IoT devices are particularly vulnerable in this stage, when the world knows about a specific class of vulnerabilities but no patches yet exist. Since some vulnerabilities are never patched, some devices will remain in Stage 1 permanently. In this stage, the goals include: identifying affected devices, filtering attack flows to these devices, and addressing common vulnerabilities. Our analysis of recent CVEs (public vulnerability notifications) in Section VI-A shows that common vulnerabilities not directly addressed by filtering include bad authentication, cross-site request forgery (XSRF), and certificate spoofing. To mitigate these, the hub should:

- *Identify affected devices.* The hub should be able to identify vulnerable devices in the home network. In some cases this might be straightforward, e.g., if there has only ever been one device of a specific type on the market and the presence of that device is very visible on the home network. In other cases this might be challenging, such as when the vulnerability disclosure is be for some dependency of a large number of products — e.g., a version of OpenSSL — and the hub would need to identify all devices in the home that depend on that version of OpenSSL.
- *Filter attack flows to IoT devices.* After a vulnerability is discovered, known attack patterns will emerge. Researchers have previously made use of traffic patterns to detect desktop malware: for example, traffic exploiting a buffer overflow vulnerability might have a characteristic packet-length [33]. The hub should prevent the exploitation of known-vulnerable IoT devices by preventing these flows from reaching the IoT devices.
- *Improve authentication.* An attack pattern that has been prominent in the recent DDoS attacks is the exploitation of hard-coded administrative usernames and passwords [5]. The hub should prevent exploitation of this vulnerability.
- *Mitigate XSRF vulnerabilities.* Cross-site request forgery (XSRF) attacks are difficult to filter because they appear similar to valid commands from an authenticated user. The hub should prevent exploitation of IoT devices known to have XSRF vulnerabilities by adding additional authentication.
- *Secure vulnerable SSL connections.* If a device does not correctly verify SSL connections, an attacker could impersonate the remote server or user that the vulnerable device is talking to. The hub can remove the vulnerability on the external network by intercepting the traffic itself and then making a properly verified connection to the remote server.
- *Inform user.* Vulnerable devices can put many people besides the user at risk, including other people in the home and people and devices across the internet who might be the target of DDoS attacks. However, if the user does not see the effects, they may not realize that one of their devices is compromised — or in danger of becoming compromised. Therefore, the hub should empower the user with information about the state of their network, including the presence of any vulnerable devices.
- *Quarantine affected APIs.* The hub should also have the ability to quarantine vulnerable devices, thereby reducing the likelihood that they will be compromised. There are several possible approaches to quarantining, including quarantining every vulnerable device, quarantining only devices whose vulnerabilities are not sufficiently mitigated by filtering, or only quarantining devices at the request of the user. Ideally, quarantining would not shut off all functionality of the device; for example, if only a small number of APIs are vulnerable, then only blocking those APIs would allow the device to still partially work.

Stage 2: Acquiring the Patch. As with Stage 1, IoT devices are particularly vulnerable at this stage — we assume that adversaries have the knowledge and ability to compromise these devices. Therefore, since a patch is available, we seek to facilitate rapid patching of vulnerable devices. Unfortunately, updating a device might require user intervention, and even if the patch can be applied without user intervention, there maybe risks with updating the device at certain times (e.g., updating lights in the night when there is a risk of a someone falling in the dark or rebooting the smart oven right when it is time to make dinner). In both the manual and automatic cases the software update might be delayed, which leads to the following goals:

- *Alert user of the update's presence, for devices that need user input to update.* Some devices require users to explicitly run some software, or otherwise interact with the device in order to perform an update. For example, many home routers require the user to download the patch and then upload it to the router. The hub must inform the user of the presence and importance of the update, as well as pointing them to instructions for installing the patch.
- *Filter, quarantine, and secure devices.* The presence of a patch will likely cause more adversaries to become aware of and attempt to exploit the vulnerability, so the techniques from Stage 1 apply here as well.

- *Cache patch if it is not a good time to update.* In many cases, updating IoT devices requires a reboot because the updates are to device firmware. Having the device reboot, and become unavailable, might not be convenient at the time the patch becomes available. Instead, the hub should cache patches for devices in the home.

Stage 3: Applying the Patch. Our key goal at this stage is to enable the application of patches when it is safe to do so. For example, it could be problematic to update the clothes dryer when laundry is nearly done in the washer. To safely apply the patch, we have the following goals:

- *User consent for IoT device updates.* Even for updates that will be applied automatically, it is important to notify the user of the forthcoming update, especially on a complicated or continually in-use device [34]. For devices that need user activity to update, this process might be more involved: if the user-update mechanism is not significantly secure, the hub should add an additional level of authentication to ensure the update process itself is not being exploited.
- *Update device at convenient time by using awareness of home state.* Once the user has been notified, automatic updates should be applied at a time when the device is not in use (or not in heavy use in the case of devices that are always active) and is not likely to be needed during the length of the update process (including any possible reboots). The hub should determine when to apply updates based on state of home and the history of use of the device.
- *Deal with update failure.* Sometimes updates fail or cause problems; this phenomenon has already been observed with IoT devices [4]. The hub should notify the user in the case of update failure and attempt to restore device state or retry the update in the case of transient failure.

Stage 4: Device Compromised. Despite our best efforts, devices may be compromised. In that situation, we aim to prevent, as much as possible, the compromised device from harming other devices or the home's security. This is particularly important when the compromised device is configured to pass information to a security-critical device: we do not want the adversary to use a compromised temperature sensor to report a fire and automatically unlock the door. Therefore, for compromised devices we seek to:

- *Detect anomalies.* If a device has been compromised, it will likely start producing traffic that is inconsistent with previous traffic or with information from other devices in the home. The hub should monitor for this and warn the user when a device is behaving irregularly, which might be more likely to inspire the user to take action (quarantine, reboot, update) than a generic warning about a vulnerability in the device.
- *Rate limit traffic from device.* If a device has been compromised but its functionality is critical enough that the user cannot remove it, some effects might still be mitigated. For example, if it attempts to send a high volume of traffic as part of a DDoS attack, the hub could block the traffic.

These examples do not nearly cover all the scenarios possible with a compromised device, but they still significantly increase security from what is available in current systems. Due to the extensibility focus in our hub's design, we believe that future tactics for compromised devices will work easily on top of our existing efforts.

C. Threat Model

The threat model for our design considers both legacy and new devices. In both cases, perfect security does not exist; instead, our goal is to raise the bar. First, the hub itself needs to be secure, receive its own software updates, run modules that are secure and correct, and its manufacturer must be informed of vulnerabilities. Compromise of the hub is outside our threat model. To achieve some of our goals, we assume that the hub can inspect the contents of device traffic, either by design or due to legacy device insecurity [9]. We assume that when devices first connect to the hub, they do not provide false identifying data (e.g. device type or firmware version).

We focus on threats from an adversary who is physically external to the home. However, such adversaries can view or modify all network traffic to and from the home and can use any compromised devices to send traffic inside the home.

IV. Security Manager Design

Figure 2 shows the architecture diagram for our proposed design. In this section, we discuss our design points in terms of both legacy devices (created without knowledge of the security manager) and future devices (aware of our security manager).

A. Intercept Communications. The hub should interpose on all communications between devices and other parties. This includes communications both from other devices inside the home network and from entities outside the network (such as apps running in the cloud).

Legacy Devices. For legacy devices, this requirement means that the hub should sit within the home gateway router, to intercept traffic from devices that communicate via WiFi (such as with many standalone security cameras or the Amazon Echo). Additionally, the hub should have radios for other in-home protocols, such as Z-Wave and Bluetooth.

Future Devices. Devices designed using this model will interact directly with the hub: the hub will have a secure connection to the outside world and to each device, monitoring and influencing the device's communications. However, transiting all data through the hub so it can access cleartext contents of that data raises privacy concerns if the hub and security manager are not manufactured by the same entity as the device. Some traffic content, such as video and audio feeds from security cameras, may not be necessary for security manager functionality and could be encrypted end-to-end from the camera to its cloud provider or remote storage; other, less sensitive data (e.g., control data) could be intercepted by the hub. In general, we identify a trade-off between hiding privacy-sensitive data from the hub and the ability of the hub to protect vulnerable devices via access to the cleartext contents. We encourage other researchers to explore solutions.

B. Be Aware of All Home Devices. Since the security manager sees all device network activity, it can keep track of which devices are on the network and keep a log of each device's

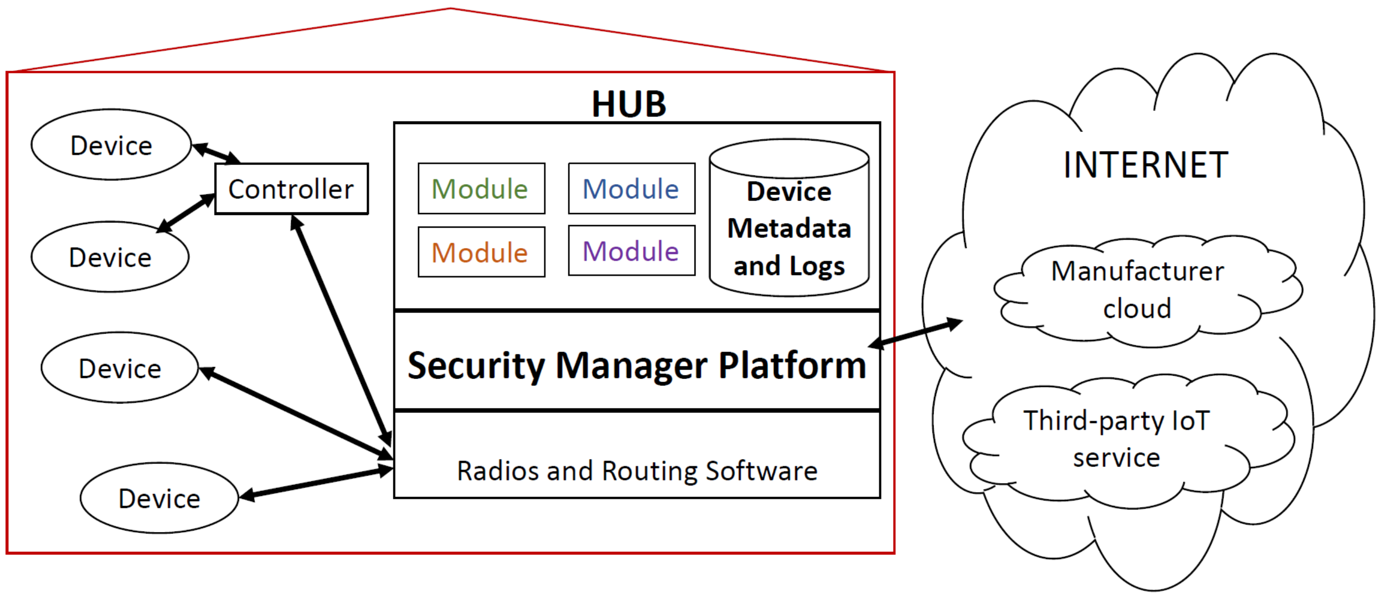

Fig. 2: **Architecture Diagram**. The hub is also the home's gateway router so that device communications transit the hub before leaving the home or reaching other devices, possibly via an intermediate controller. The security manager platform sits atop the communication software, and serves as an extensible platform for modules that perform specific security functionality. To support the modules, the platform stores device metadata and logs of device activity. Security manager features are orthogonal to other manufacturer or third-party features operating in the cloud.

status. By storing this information locally, the security manager can reason about devices — e.g., determining appropriate times to reboot devices for an update or deciding whether any devices have firmware versions affected by a new vulnerability report.

Legacy Devices. Legacy devices may lack APIs available to report firmware version numbers or other metadata. Since the hub can access all device communication, it can fingerprint the device to get that information, even if device traffic is encrypted. Additionally, if the hub can successfully intercept communication contents (because they are in plaintext or use poorly implemented cryptography [9]), it can keep a history of device status: commands to the devices and information from the devices.

Future Devices. Devices should accurately report firmware version and other metadata while they are not compromised. Additionally, the hub can log all communications it monitors.

C. Pre-Fetch and Install Updates. A core trusted feature of the security manager is its ability to assist with updating home devices. The availability and integrity of updates is critical for secure device functionality.

Legacy Devices. Even for devices that cannot be automatically updated via the security manager, the home awareness features could still prove useful. For example, the hub could check the integrity of remotely available updates and notify users at a time when they are home and can install the updates.

Future Devices. For new devices, which accept over-the-air updates pre-fetched by the hub, we can do much more, including downloading updates, checking their integrity, and storing them in the hub. After user notification, we can then install the updates that support automatic installation.

D. Provide Extensibility. Our security goals may be achieved more effectively in pieces, by multiple engineering teams with various types of expertise. For example, anti-malware expertise might help an engineer determine patterns in attack traffic that could serve as the basis for a filter. To leverage a diversity of expertise and handle the breadth of associated problem domains, the security manager is designed to be extensible: engineers can add modules to support specific functionality. Modules register at any or all of four points exposed by the security manager:

- When new devices are added to the system.
- When any command is headed to the device.
- When any reading comes from the device.
- Independently of device communication, at certain times.

To make our proposal concrete, we explore some specific modules below that address different points in the vulnerability space. We then evaluate some example modules in Section VI.

V. Security Manager Modules

We envision modules being built on top of the security manager to address specific classes of vulnerabilities or points in the vulnerability lifecycle. Using separate modules for different security goals lets us confine the trusted computing base for each security goal to the security manager and the relevant module rather to than the system as a whole. We explore this vision by proposing several example modules.

A. Modules for Stage 1: Vulnerable (no patch yet)

- *Identify affected devices.* Currently, users may not know the manufacturer or firmware version of their IoT devices; therefore, even if they hear about a vulnerability, it might not be easy to discover if it applies to any of their devices. However, because our security manager is designed to store device type, manufacturer, and version information for each device, it can automatically determine which devices are affected when given a vulnerability report. A module, could example, continuously pull from US-CERT's CVE list [21] and compare new vulnerability notifications against the set of devices in the home.

- *Filter attack flows to IoT devices.* In 2004, Wang et al. introduced network-level Shields (sitting between the transport and application layer) to filter out exploit traffic before devices have been patched. Originally applied to worms [33] and dynamic HTML [25], the Shield idea seems fitting in the IoT space, where updates may move more slowly than attacks, and firewall rules are too restrictive. Since the security manager has accurate information about firmware versions for un-compromised devices, we propose a filtering module that checks for devices with vulnerable firmware versions and pulls new filter rules from a trusted third party (such as the CVE list or other public vulnerability notifications).
- *Improve authentication.* Yu et al. (2015) propose changing default passwords on IoT devices — without having to inform the user — by having a network middlebox translate between the old password entered by the user and a new, secure, password written to the device [36]. With the information in the hub and a list of default usernames and passwords for given device and firmware versions, a module could change these passwords automatically, without any user input necessary.
- *Mitigate XSRF with two-factor authentication.* Cross-site request forgery (XSRF) attacks could make attacker-initiated commands appear to come from an authenticated user. Using the hub's ability to intercept communication and awareness of the current status of multiple devices, we envision a module that delays commands for devices known to have XSRF vulnerabilities until users provides a second factor of authentication to verify their request. For example, the module could use the status of several home sensors to ensure that the user was truly present, or even require the toggling of a specific sensor or button shortly after the command.
- *Secure vulnerable SSL/TLS connections.* Denning et al. discovered that some IoT devices do not verify server certificates when performing a key exchange [9]. Therefore, an adversary could spoof or intercept the connection. However, because our hub is designed to intercept all traffic to and from devices, a module on the hub could spoof the connection proactively: terminating the unvalidated connection before it leaves the home and connecting securely and verifiably with the remote server. This security increase — essentially, man-in-the-middle as a service — is achievable only because our proposed hub runs locally within the home.
- *Inform user.* A hub module could notify the user of vulnerable devices via various means such as flashing lights or push notifications.
- *Quarantine.* If a device has a vulnerability whose exploitation cannot be blocked by the filtering module, it may be necessary to drop all traffic headed for the device or all traffic that uses a particular device feature or API. Since the hub intercepts all traffic, it has the ability to enforce this sort of selective quarantining, rather than the all-or-nothing, connected or disconnected, solution present today.

B. Modules for Stage 2: Acquiring the Patch

- *Alert the user of the update's presence.* Even if users know that one of their devices is vulnerable, they may not know how to obtain an update once one exists. We envision a module that uses the hub's knowledge of device details to cross-check current device version numbers against the manufacuturer's website or a centralized database. This is a particularly important solution for legacy devices that do not support over-the-air updates.
- *Continue the security measures from Stage 1.* The device is still vulnerable at this stage so the modules described above could all still be useful. However, a quarantine might need to be temporarily lifted in order for the device to acquire its update.
- *Pre-fetch patch.* For devices that accept updates locally (either because they lack over-the-air update capabilities or were designed with our security manager in mind), a module on the hub can check for updates and pre-fetch patches. Pre-fetching might be particularly valuable if, for example, a battery-powered device is offline when the patch becomes available, or if there are multiple, identical, power-limited devices (e.g., many lightbulbs) that need simultaneous updating.

C. Modules for Stage 3: Applying the Patch

- *User consent for IoT device updates.* Beyond the information and notification discussed in the previous stages, an update module could also solicit a second factor of authentication from the user — similar to the XSRF mitigation described above — for update procedures that are less secure.
- *Home state aware update timing.* For updates that can be applied automatically and have been pre-fetched to the hub, we propose a module that applies the hub's collected history of device status and usage to determine a safe, convenient time to install the update (and reboot the device if necessary). This determination could be based on simple heuristics, such as time-of-day usage patterns, or on more advanced machine learning techniques.
- *Dealing with update failure.* If an automatic or user-activated update fails, the hub can inform the user and retry the update. It can also use its backups of device state to refresh information on the device if needed.

D. Modules for Stage 4: Device Compromised

- *Detect anomalies.* The first challenge in dealing with a compromised device is detecting its compromised state. To this end, we envision an anomaly detection module that can make use of both device state history and information from other home devices (particularly sensors) to detect unlikely information emanating from the device. Detection could be based on simple heuristics or machine learning. This requires that the hub store baseline information about the device prior to its compromise.

Vulnerability type	Num.	Fixable w/ Hub?	Main Modules Used
Filterable	19	16-19 (84-100%)	Filtering
Authentication	5	4-5 (80-100%)	Several (see description)
XSRF	4	4 (100%)	Two-factor authentication
Certificate spoof	3	3 (100%)	Several (see description)
Denial of service	1	1 (100%)	Quarantine
Other/unknown	4	0-1 (0-25%)	Unknown
Total	36	28-33 (78-92%)	

TABLE I: **Analysis of recent IoT related CVEs.** About half of the CVEs could be mitigated by simple filtering. Additionally, for all categories within the scope of our threat model, the security manager could perform some mitigation.

- *Rate limit traffic.* A particular behavior of compromised devices may be sending a high volume of traffic to an unusual end point as part of participation in a DDoS attack. A rate-limiting module can track traffic frequency from each device and block devices that send too quickly. This module's behavior is possible only because the hub runs locally: if data from the device needed to reach the cloud in order to enforce this security property, its benefit would be significantly reduced. Rate limiting is also useful in the other direction (traffic to a device) for safety purpose (e.g., to avoid flickering a light switch [23] or denying service to an alarm system).

Stepping back, these modules offer an outline of the possibilities that a hub-based security manager might offer. Due to the extensibility of the design, all of these features are feasible insofar as they are needed or supported by the end devices. This provides confidence that additional security modules designed by other researchers in the future will also work easily with our security manager's design.

VI. Evaluation

We seek to assess whether our design applies to real vulnerabilities (Section VI-A) and also whether it supports the easy development of modules to address different points in the vulnerability lifecycle (Section VI-B).

A. Common Vulnerability (CVE) Analysis

Table I shows an analysis of Common Vulnerabilty and Exposure (CVE) notifications from 2012-2016 based on a search using keywords related to IoT. This sample illustrates the types of vulnerabilities found in IoT devices.

About half of the CVEs considered could be mitigated by filtering; in nearly all of those cases, the hub would be able to do so. These CVEs included cross-site scripting attacks, directory traversal attacks, buffer overflow, SQL injection, and crafted URLs or other specialized behavior. In the remaining three cases, the author of the filtering module would need more information than was available in the CVE.

For the other in-scope vulnerability types, the security manager design is similarly successful. The denial of service attack requires quarantining the device and notifying the user. Cross-site request forgery vulnerabilities could be addressed via the addition of a second-factor device. Since the security manager is aware of multiple devices in the home, it could delay the request to the vulnerable device until it receives a signal from the second-factor device. Most complicated to mitigate would be certificate spoofing — because the specific vulnerabilities deal with the update process — and bad authentication — because the mitigation (filtering, fixing the authentication, quarantining, and notifying the user) depends heavily on the vulnerability.

Finally, 4 CVEs were out of scope, with situations like compromise before the device could connect to the hub, which is outside our threat model, or with insufficient information about the vulnerability in the CVE.

B. Case Study Modules

In Section V, we introduced a number of modules that address different points in the IoT vulnerability lifecycle from Section III-B. Here, we assess the ease and feasibility of prototyping example modules on our framework.

To prototype these modules, and to focus our work on exploring the security manger itself, we do not work with real devices but instead emulate them. We use Microsoft Research's HomeOS [11] to capture both data from real Z-Wave devices (the versions of Aeon Labs' controller, multisensor, switch, and door-window sensor compatible with the driver in HomeOS[1]) and commands from HomeOS apps to these devices. Then, we replay that data from our emulated devices and spoof an attacker or compromised device. This approach provides the greatest flexibility to evaluate our system under both real and synthesized adversarial conditions, and it particularly lets us experiment with vulnerabilities that are not present in the specific versions of the devices we use. We prototype a security manager platform that intercepts communications, stores histories of device status, and offers the extensibility API points that we proposed in Section IV. Finally, we built the modules discussed below on top of the prototype security manager platform.

1) Filtering Traffic: Our first prototype evaluation addressed filtering for known vulnerabilities not yet patched on the device. Our implementation was inspired by the Shield work [33] and a CVE for the Foscam security camera describing a directory escape vulnerability that allowed the `passwd` file to be exfiltrated to a remote attacker [7]. To mitigate this vulnerability, we intercepted traffic bound to the device, used stored information about the device's firmware version, and then filtered all commands to devices with vulnerable firmware versions for the string `".."`. Although this filter is extremely simple, the structure of the module allowed the addition of additional filters based on device metadata (manufacturer, device type, version number), with only a few lines of code per filter. Although creating such filters currently requires manually examining the CVE disclosure page and determining a patch, future work might automate this function.

2) Patching at a Convenient Time: We focused our patching module prototype on the challenge of determining appropriate times to patch. This made use of the whole-home awareness of the security manager, checking the history of use patterns of each device, so that, for example, the switch is not out of service when the homeowner is likely to use the lights. Additionally, since updates might require the system to be rebooted, the switch must be in the off position for the update to be the least disruptive. This module used historical

[1]Devices can be found at https://labofthings.codeplex.com/wikipage?title=Lab\%20of\%20Things\%20Devices.

information about the device under consideration and could be expanded to use information from multiple devices; for example, high luminosity readings on an adjacent multisensor might indicate less need for the lights. The module ran asynchronously, independent of device or app input after being started when the first device was connected.

3) Rate Limiting: An unusually high bandwidth of traffic from a device could indicate a compromise and participation in a DDOS attack, while frequently flipping commands to a device could be bad for the device and a sign of an incorrectly configured app. We prototyped a small rate limiting module. This required minimal state (the time of the last device interaction), and the same (less than 10 lines of code) check could be used for traffic in either direction by registering the module on both entry-points.

Summary of Evaluation. Our evaluation confirmed the validity of our proposed direction: leveraging an extensible, hub-based security manager to improve the security of a home's IoT ecosystem. The CVE analysis showed that the modules we envisioned could address common vulnerabilities in IoT devices; the prototype module evaluation confirmed that the extensibility API in our proposed design facilitates module implementation.

VII. Discussion

As mentioned in the threat model (Section III-C), the security of the hub itself is critical if it is going to successfully secure other devices in the home. As a single point of failure, the hub may offer a more promising target for attack, but its manufacturer will have significantly more resources to devote to security than individual device makers since security is inherent to the hub's function. Additionally, trade-offs must be made in the security of individual modules: for example, locking down remote updates may make it difficult for users to add third-party software to their devices or keep it up-to-date, perhaps stifling innovation; however, allowing updates without full integrity checks could allow malicious actors to compromise home devices. Another trade-off comes in modules such as anomaly detection that collect information about many devices over time, which may have privacy implications. We encourage future research to explore these challenges.

As the central security manager design is adopted and individual devices designed to make use of its presence, there are opportunities to reduce the vulnerability surface. For example, many CVEs for WiFi-connected IoT devices come from running a local webserver that hosts device configuration pages. A device designed with a hub-based security manager in mind can outsource its configuration to an app running locally on the hub, as suggested in the HomeOS work [10,11] and appearing in the 2015 incarnation of Samsung's SmartThings [1]. Finally, although we focused on the smarthome environment, we imagine that our design and lessons could serve in other contexts. For example, some IoT devices are mobile, such as cars, and may have additional challenges [18]. IoT devices are also widely used in enterprise or industrial settings, which may have significantly more devices than a home. We believe our ideas carry over and we encourage other researchers to further explore challenges specific to these settings.

VIII. Conclusion

Home IoT devices — and security vulnerabilities on home IoT devices — now pervasive, pose vulnerabilities as substantial as their benefits. Quickly patching IoT devices to remove vulnerabilities is not always straightforward due to the longevity of some home appliances, the lack of software update capabilities, and the challenges of predicting a convenient time to update. This paper proposes an approach to reduce the harm from unpatched vulnerabilities: a security manager on top of a centralized IoT hub. This manager would be located inside the home's gateway router so that it can intercept communications to and from devices and have the vantage point to detect anomalous network activity from specific devices. Our proposed manager is aware of all IoT devices in the home and their usage patterns, and offers an easy-to-use extensibility API for the creation of modules that address specific classes of vulnerabilities or points in the vulnerability lifecycle.

Our analysis shows that this direction is viable: a few modules built on top of the security manager platform could address many recent home IoT vulnerabilities, and expanding the modules to address new devices or vulnerabilities would be straightforward. In proposing this security manager design, we hope to stimulate research and encourage further work in this area by academia, industry, and regulators.

Acknowledgements. We thank Karl Koscher for technical assistance with IoT hardware, and Camille Cobb, Pedro Fonseca, Gennie Gebhart, Kiron Lebeck, Niel Lebeck, Ada (Adam) Lerner, Amrita Mazumdar, Ellis Michael, Lucy Simko, Ian Smith, Adriana Szekeres, Alex Takakuwa, Paul Vines, Irene Zhang, and Eric Zeng for feedback on earlier drafts. This work was supported in part under NSF Grant No. CNS-1565252, an NSF Graduate Research Fellowship under Grant No. DGE-1256082, and the Short-Dooley Professorship.

References

[1] "Samsung SmartThings Hub FAQ — SmartThings Developer Documentation," Sep. 2015. [Online]. Available: http://docs.smartthings.com/en/latest/sept-2015-faq.html

[2] "Notice of Multistakeholder Process on Internet of Things Security Upgradability and Patching Open Meeting | NTIA," 2016. [Online]. Available: https://web.archive.org/web/20161028003346/https://www.ntia.doc.gov/federal-register-notice/2016/10192016-meeting-notice-msp-iot-security-upgradability-patching

[3] A. Bellissimo, J. Burgess, and K. Fu, "Secure software updates: Disappointments and new challenges." in *HotSec*, 2006.

[4] N. Bilton, "Nest Thermostat Glitch Leaves Users in the Cold," *The New York Times*, Jan. 2016. [Online]. Available: http://www.nytimes.com/2016/01/14/fashion/nest-thermostat-glitch-battery-dies-software-freeze.html

[5] Brian Krebs, "Hacked Cameras, DVRs Powered Todays Massive Internet Outage," Krebs on Security, Oct. 2016. [Online]. Available: https://krebsonsecurity.com/2016/10/hacked-cameras-dvrs-powered-todays-massive-internet-outage/

[6] S. Checkoway *et al.*, "Comprehensive experimental analyses of automotive attack surfaces." in *USENIX Security Symposium*, 2011.

[7] CVE, "CVE-2013-2560," 2013. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-2560

[8] T. Denning, T. Kohno, and H. M. Levy, "Computer security and the modern home," *CACM*, vol. 56, no. 1, p. 94, 2013.

[9] T. Denning, C. Matuszek, K. Koscher, J. R. Smith, and T. Kohno, "A spotlight on security and privacy risks with future household robots: Attacks and lessons," in *UbiComp '09*, 2009.

[10] C. Dixon, R. Mahajan, S. Agarwal, A. J. Brush, B. Lee, S. Saroiu, and V. Bahl, "The home needs an operating system (and an app store)," *Hotnets '10*, 2010.

[11] C. Dixon, R. Mahajan, S. Agarwal, A. Brush, B. Lee, S. Saroiu, and P. Bahl, "An Operating System for the Home," *NSDI*, 2012.

[12] E. Fernandes, J. Jung, and A. Prakash, "Security Analysis of Emerging Smart Home Applications," in *Proceedings of the 37th IEEE Symposium on Security and Privacy*, 2016.

[13] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash, "Flowfence: Practical data protection for emerging iot application frameworks," in *USENIX Security Symposium*, 2016.

[14] Gartner, "Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent From 2015," Nov. 2015. [Online]. Available: http://www.gartner.com/newsroom/id/3165317

[15] A. Hern, "Revolv devices bricked as Google's Nest shuts down smart home company," *The Guardian*, Apr. 2016.

[16] K. Hill, "'Baby Monitor Hack' Could Happen To 40,000 Other Foscam Users," Aug. 2013. [Online]. Available: http://www.forbes.com/sites/kashmirhill/2013/08/27/baby-monitor-hack-could-happen-to-40000-other-foscam-users/

[17] K. Koscher *et al.*, "Experimental security analysis of a modern automobile," in *2010 IEEE Symposium on Security and Privacy*, 2010.

[18] T. Kuppusamy, A. Brown, S. Awwad, D. McCoy, R. Bielawski, C. Mott, S. Lauzon, A. Weimerskirch, and J. Cappos, "Securing software updates for automobiles," in *To appear at escar EU 2016*, 2016.

[19] A. Mathur, J. Engel, S. Sobti, V. Chang, and M. Chetty, ""They Keep Coming Back Like Zombies": Improving Software Updating Interfaces," in *SOUPS 2016*, 2016.

[20] A. McLean, "IoT malware and ransomware attacks on the incline: Intel Security," Sep. 2015. [Online]. Available: http://www.zdnet.com/article/iot-malware-and-ransomware-attacks-on-the-incline-intel-security/

[21] MITRE, "CVE - Common Vulnerabilities and Exposures (CVE)." [Online]. Available: https://cve.mitre.org/

[22] T. Nighswander, B. Ledvina, J. Diamond, R. Brumley, and D. Brumley, "GPS Software Attacks," in *ACM CCS '12*, 2012.

[23] T. Oluwafemi, T. Kohno, S. Gupta, and S. Patel, "Experimental Security Analyses of Non-Networked Compact Fluorescent Lamps: A Case Study of Home Automation Security," *LASER 2013*, 2013.

[24] RealKnol, "The Home Automation Market by the Numbers," Jun. 2015. [Online]. Available: https://blog.remotely.com/2015/06/20/the-home-automation-market-by-the-numbers/

[25] C. Reis, J. Dunagan, H. J. Wang, O. Dubrovsky, and S. Esmeir, "Browsershield: Vulnerability-driven filtering of dynamic html," *ACM Trans. Web*, vol. 1, no. 3, Sep. 2007.

[26] F. Roesner, T. Kohno, and D. Molnar, "Security and privacy for augmented reality systems," *CACM*, vol. 57, no. 4, pp. 88–96, 2014.

[27] M. Rushanan, D. Foo Kune, C. M. Swanson, and A. D. Rubin, "SoK: Security and Privacy in Implantable Medical Devices and Body Area Networks," in *Proceedings of the 35th Annual IEEE Symposium on Security and Privacy*, May 2014.

[28] J. Samuel, N. Mathewson, J. Cappos, and R. Dingledine, "Survivable key compromise in software update systems," in *ACM CCS '10*, 2010.

[29] Scott Hilton, "Dyn Analysis Summary of Friday October 21 Attack," Oct. 2016. [Online]. Available: http://hub.dyn.com/dyn-blog/dyn-statement-on-10-21-2016-ddos-attack

[30] A. K. Simpson, F. Roesner, and T. Kohno, "Securing vulnerable home iot devices with an in-hub security manager," *The First International Workshop on Pervasive Smart Living Spaces (PerLS 2017)—in conjunction with IEEE PerCom 2017*, 2017.

[31] Y. Son, H. Shin, D. Kim, Y. Park, J. Noh, K. Choi, J. Choi, and Y. Kim, "Rocking drones with intentional sound noise on gyroscopic sensors," in *USENIX Security Symposium*, 2015, pp. 881–896.

[32] United States Federal Trace Commission, "IoT Home Inspector Challenge," 2017. [Online]. Available: https://www.ftc.gov/iot-home-inspector-challenge

[33] H. J. Wang, C. Guo, D. R. Simon, and A. Zugenmaier, "Shield: Vulnerability-Driven Network Filters for Preventing Known Vulnerability Exploits," in *SIGCOMM*, 2004.

[34] R. Wash, E. Rader, K. Vaniea, and M. Rizor, "Out of the Loop: How Automated Software Updates Cause Unintended Security Consequences," *SOUPS '14*, 2014.

[35] N. Woolf, "DDoS attack that disrupted internet was largest of its kind in history, experts say," *The Guardian*, Oct. 2016.

[36] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu, "Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the Internet-of-Things," in *Hotnets*, 2015.

APPENDIX

We provide here the list of CVEs that we analyzed in our evaluation (Section VI-A). They were collected by searching the CERT vulnerability list [21] for keywords related to IoT and Home Automation from the years 2012-2016 and selecting any that corresponded to IoT devices (as opposed to routers, software, servers, or other vulnerable categories). The vulnerabilities are sorted from most to least recent.

```
CVE-2016-0866
CVE-2016-0865
CVE-2016-0864
CVE-2016-0863
CVE-2015-0739
CVE-2014-9517
CVE-2014-9238
CVE-2014-9234
CVE-2014-2362
CVE-2014-2361
CVE-2014-2360
CVE-2014-1911
CVE-2014-1849
CVE-2013-6952
CVE-2013-6951
CVE-2013-6950
CVE-2013-6949
CVE-2013-6948
CVE-2013-5321
CVE-2013-5215
CVE-2013-3962
CVE-2013-3690
CVE-2013-3689
CVE-2013-3687
CVE-2013-3686
CVE-2013-3541
CVE-2013-3540
CVE-2013-3539
CVE-2013-3417
CVE-2013-2560
CVE-2013-1219
CVE-2012-4621
CVE-2012-4046
CVE-2012-3002
CVE-2012-1288
CVE-2012-0284
```