If-This-Then-Allow-That (to Phone Home): A Trigger-Based Network Policy Enforcement Framework for Smart Homes

Anthony Tam[1], Furkan Alaca[1], and David Barrera[2]

[1] Queen's University, Kingston, ON, Canada
{anthony.tam,furkan.alaca}@queensu.ca
[2] Carleton University, Ottawa, ON, Canada
david.barrera@carleton.ca

Abstract. The Internet of Things (IoT) has become entrenched in many users' networks due to the utility these Internet-connected objects provide. But this does not mean that users should unconditionally trust IoT devices on their networks. While several approaches exist for restricting network connectivity of IoT devices, these proposals typically identify legitimate traffic, and then permanently allow it to flow to or from the device. In this paper, we argue that this permanent access control can lead to privacy and security violations, and in many cases is not strictly required. We present If-This-Then-Allow-That (IFTAT), a framework that supports security policies that dynamically update network access control rules based on the type of access that is required at any given time. Device or environmental triggers such as motion sensors or mobile phone applications initiate the process of adding firewall exceptions, which are removed either automatically or after another trigger is activated. We describe a proof of concept implementation which shows how IFTAT can restrict the network access of untrusted IoT devices with little impact to the usability of these devices.

1 Introduction

Internet of Things (IoT) devices have been widely exploited by attackers to carry out malicious activities against their users and the Internet at large [1, 7, 11, 14, 21]. The exploitation of IoT devices also threatens user privacy, since IoT devices often access and handle privacy-sensitive data such as audio or presence information. While evidence suggests [19] that users are interested in knowing what data their devices send over the network and why, users are often unaware of this information due to the lack of security and privacy tools that provide it. To mitigate the exploitation of IoT devices, prior research has proposed mechanisms to narrow the scope of allowable IoT device network traffic, e.g., by restricting allowable protocols or allowable source and destination ports and hosts [5, 18, 22]. Many proposed methods employ security policies to allow network traffic that matches pre-defined rules, but few of these methods update the rules in response to contextually-relevant information. This means that a smart doorbell

Version: November 27, 2022. This is the authors' version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record will appear in the conference proceedings of the 15th International Symposium on Foundations & Practice of Security (FPS 2022), published in the Springer Lecture Notes in Computer Science (LNCS) series. Use of this Accepted Version is subject to the publisher's Accepted Manuscript terms of use: https://www.springernature.com/gp/open-research/policies/accepted-manuscript-terms

could upload audio or video to the cloud even while the user is not using the doorbell's mobile app, or that a motion sensor could report motion activity even when an alarm system is disarmed. We are thus motivated to rethink IoT network access control: we aim to provide users with greater control of when their IoT devices communicate over the network, and we do so by providing a means to mediate network traffic based on contextual events that reflect users' real-world usage patterns.

We propose the If-This-Then-Allow-That (IFTAT) framework, which introduces a new time-based dimension to network access control by allowing user-defined policies to update traffic mediation rules in response to trigger events. Trigger events can be generated by sources such as physical sensors, network traffic signatures, or software running on a user's device. User-defined policies specify rules for allowing or denying network traffic to or from a target device (e.g., a security camera, thermostat, doorbell, network-attached storage) in response to the occurrence of a trigger event (e.g., motion detection, light switch turned on, application launched on a user's smartphone). For example, a thermostat might only be allowed to communicate with a cloud service provider while there is a human user in close physical proximity to the thermostat; or a surveillance camera might only be allowed to stream video to the cloud while the owner of the camera has the companion app running in the foreground on their smartphone. IFTAT does not need to learn the network traffic patterns of IoT devices, but IFTAT policies can optionally use such patterns to enable more granular rules for allowing or denying network traffic. Fig. 1 illustrates the timeline of events when executing a policy that applies a traffic mediation rule (e.g., allow a security camera to access the Internet) in response to a trigger event, followed by a change or reversal of that rule (e.g., by denying the security camera access to the Internet) in response to a subsequent trigger event.

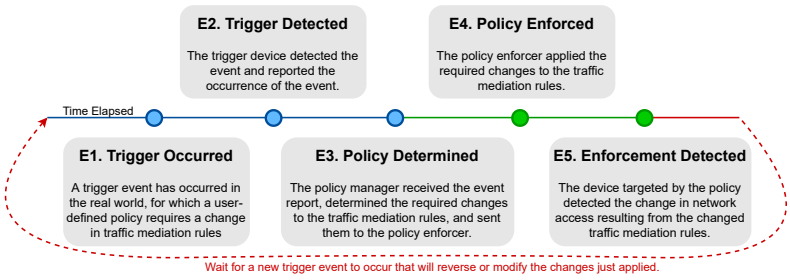

Fig. 1. The timeline of events that occur while detecting a trigger event (blue), applying a change in traffic mediation rules (green), and waiting for a new event to subsequently reverse or modify the previous change in traffic mediation rules (red).

IFTAT is well-suited for protecting devices that have the characteristic of predictably and consistently requiring network access in response to specific

trigger events that can be detected by other devices on the network. IoT devices in particular fit this general characteristic, since they are generally known to have specialized features that can be implemented by predictable and repetitive actions such as submitting sensor readings to a cloud service at a regular interval [22]. However, our system may also complement traditional firewall rules to reduce the attack surface on more general-purpose devices as well, particularly those that provide services over the network such as network-attached storage.

The contributions of this paper are as follows.

1. We present IFTAT, a framework for time-based network access control derived from user-defined trigger events.
2. We demonstrate the use and effectiveness of IFTAT on mainstream IoT devices and hardware through a proof of concept implementation. We propose and implement an initial set of trigger events.
3. We provide a discussion of how IFTAT defends against two distinct threat models that we define.

The remainder of the paper is structured as follows. Sections 2 and 3 outline the IFTAT security goals and system design, respectively. Section 4 discusses related work. Section 5 discusses and categorizes trigger events that we propose and that we identify from other work. Section 6 presents use cases that we implement with our instantiation of IFTAT using commodity hardware and software. Section 7 discusses security considerations and potential focuses for standardization effort, and Section 8 concludes.

2 Security Goals and Threat Model

IFTAT reduces the attack surface of IoT and other special-purpose devices by reducing the times during which they have unrestricted network or Internet access. This reduced network access is designed to serve as a defense against the following two threat models.

T1: Large-scale opportunistic exploitation. IoT devices are prone to being targeted and exploited by large-scale operations. Large botnets have been created via automated device scanning and exploitation, and leveraged for malicious activities such as distributed denial of service (DDoS) attacks [1, 7]. Residential proxies as a service leverage both volunteer users and compromised IoT devices as proxies to funnnel customers' traffic through residential Internet connections to evade measures such as bot detection or geoblocking [11]. These operations often exploit devices running firmware with unpatched vulnerabilities. Confining the network access of devices to only periods when the access is required would significantly narrow their window of exploitability (and, in the event of device compromise, would reduce the time windows during which they can be leveraged for malicious activities).

T2: Exfiltration of privacy-sensitive data. IoT devices often handle privacy-sensitive data such as audio or presence information. Restricting devices' network connectivity reduces the potential for the exfiltration of privacy-sensitive

data, whether due to exploitation as explained immediately above, device error (e.g., due to misinterpreting the user's intentions [13]), or the collection of tracking and usage data by device manufacturers [8].

3 System Design and Overview

The IFTAT framework specifies four device classes: untrusted devices, trigger devices, policy enforcers, and a policy manager. Fig. 2 illustrates how these devices interact with each other: trigger devices report trigger events to the policy manager, which in turn sends instructions to policy enforcers for how to mediate network traffic to and from untrusted devices. IFTAT can function with multiple trigger devices, untrusted devices, and policy enforcers on the same network.

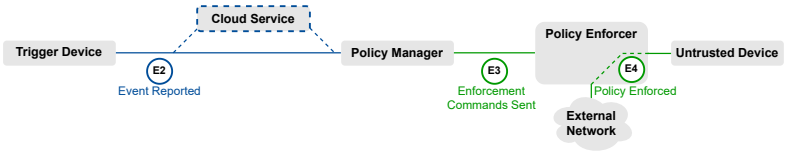

Fig. 2. Sequence of actions performed by each device following the occurrence of a trigger event. Each line represents a network connection, and is labelled with an action performed over that connection during events E2-E4 from Fig. 1. The dashed line at the policy enforcer denotes a network bridge, and the dashed lines leading to the cloud service denote an optional communication path used by some trigger devices.

3.1 Untrusted Devices

An untrusted device is a network-connected device designated by the user to have its network access mediated by the occurrence of user-defined trigger events. Untrusted devices may often be IoT devices, since they have simpler network traffic patterns [22] and often handle privacy-sensitive data, which may motivate users to ensure that such devices remain uncompromised [19]. Examples of such IoT devices may include security cameras, voice assistants, and door locks. General-purpose devices may also be designated as untrusted devices; e.g., a user may wish to restrict network access to their network-attached storage (NAS) device that performs backups of the user's other devices; the user may wish to allow network access to the NAS only in response to trigger events that indicate that a backup will take place (e.g., the user launching a backup application).

3.2 Trigger Devices and Trigger Events

A trigger device detects trigger events and reports them to the policy manager. Examples of trigger devices include special-purpose hardware devices such as

motion sensors or light switches. A trigger device may also be a general-purpose device, such as a smartphone running a program that reports user actions, or a network traffic analysis device that reports the occurrence of device activities (e.g., streaming video, updating firmware) on the network [17]. Examples of trigger events are discussed further in Section 5. Depending on the context, a device may be both an untrusted device and a trigger device; for example, a motion sensor may report a trigger event to enable network connectivity for a light switch, and the light switch may in turn report a trigger event to enable network connectivity for a security camera.

Trigger events may have different delays between the events E1, E2, and E3 denoted in Fig. 1. The delay from E1 to E2 is the time elapsed between the event occurring and the detection of the event by the trigger device. For example, passive infrared (PIR) motion sensors typically report motion if it has been detected continuously for a period of time such as two seconds. In contrast, the opening of a mobile application can be detected virtually instantaneously when the user taps on the application icon. The delay between E2 and E3 is the time elapsed between the trigger device reporting an event and the policy manager receiving the report. This delay can be near instantaneous if the trigger device is on the local network and report events to the policy manager over the local network; if the trigger device instead reports events to a cloud service provider, additional delay will be introduced since the policy manager would need to poll the cloud service provider at a regular interval to identify the occurrence of an event. RTX-IFTTT [3] provides a technique to minimize this delay by sniffing outgoing cloud API calls made by IoT devices on the local network, which would eliminate the need for the policy manager to poll the cloud API.

3.3 Policy Enforcer

Policy enforcers mediate network access to and from untrusted devices. A policy enforcer may employ a packet filtering firewall to mediate access based on attributes such as IP address, protocol, and port number. Alternatively, it may employ an application-layer firewall, e.g., to mediate HTTP requests. Each policy enforcer receives instructions from a centralized policy manager for how to enforce network access restrictions. The instructions received will be tailored to the traffic mediation capabilities of the policy enforcer and the set of untrusted devices that are connected to the network through the policy enforcer.

3.4 Policy Manager

The policy manager performs the following key functions:

Policy creation and storage. An interface and syntax is provided for the creation of IFTAT policies. IFTAT policies define how to allow or block network access to or from an untrusted device when a specified trigger event occurs. Network access may be allowed or blocked either in whole or based on specified network packet header attributes or traffic patterns that the policy enforcer is capable of identifying.

Trigger event report collection. An interface is provided for trigger devices to report that a trigger event has occurred. Trigger events can be reported by either local or remote trigger devices. Local trigger devices report events via a local interface such as USB, Bluetooth, Zigbee, or Z-Wave. Remote trigger devices report events over a network interface either directly to the policy manager or to a cloud service (e.g., over HTTPS) that the policy manager can poll to determine when a trigger event occurs.

Untrusted device designation. The policy manager retrieves the list of all devices connected to each policy enforcer, and provides an interface through which the user designates the devices that are untrusted.

Policy translation and distribution. The policy manager must translate IFTAT policies into instructions that can be enforced by the policy enforcer(s). For example, IFTAT policies can be converted into packet filtering rules or Software-Defined Networking (SDN) policies. When a trigger event report is received, the policy manager (i) identifies traffic mediation actions corresponding to any IFTAT policies triggered by the event; and (ii) sends the instructions necessary for executing the actions to the corresponding policy enforcer(s).

4 Related Work

Many systems for mediating IoT traffic have been proposed; e.g., machine learning classifiers can identify and block anomalous or malicious traffic [10, 15]; or devices can be assigned network access policies based on general device categories [4] or specifications of intended network access patterns provided by device manufacturers [9]. Here, we discuss three systems in related work that modify or update their traffic mediation behaviour in response to events observed on the network; i.e., systems that employ what we refer to as trigger devices in IFTAT.

Table 1 summarizes and compares these three systems on the basis of how each of them implements functionality that falls within the responsibility of IFTAT trigger devices, policy enforcers, and policy managers. These systems can be implemented in IFTAT, since the framework allows for the implementation of different types of trigger devices and policy enforcers.

LeakyPick [13] uses a microphone-equipped security device to passively listen for selected "wake words" (e.g., "Alexa" or "Hey Google") to be spoken by the user nearby a smart voice assistant. The security device checks if the voice assistant connects to the Internet without the wake word having been spoken; this is intended to check whether the voice assistant is only active while the user intends it to be. The authors suggest that this technique could also be used to deny network access to the voice assistant unless the wake word is spoken.

HomeSnitch [17] classifies IoT device communication into actions (e.g., firmware update, video upload) using a classification algorithm on features extracted from network traffic. A policy language is also proposed, which can allow or deny specific device activities, or use a device activity as a condition to allow or deny other traffic. Since supervised learning is used to train the classifier on a manually-labelled dataset, it is proposed that a service provider would be responsible for

Related work	Trigger device	Policy enforcer	Policy manager
LeakyPick† [13]	**Trigger event:** User-spoken wake word **Detection method:** Monitor ambient sound with microphone	WiFi access point for untrusted devices denies traffic by default and forwards traffic when signalled by policy manager	**Policy syntax:** Rule to permit network access for an untrusted device when a specified trigger is detected **Policy source:** User-specified
HomeSnitch [17]	**Trigger event:** Activity performed by an IoT device **Detection method:** Identify activities using a classifier pre-trained on network traffic signatures	Network gateway that receives OpenFlow rules from policy manager	**Policy syntax:** Rule to permit an untrusted device activity when a specified trigger is detected **Policy source:** User-specified rules; signatures downloaded from 3rd party
SerenIoT [22]	**Trigger event:** Change in an IoT device's network traffic patterns **Detection method:** Identify packet signatures that were not previously observed	WiFi access point for untrusted devices that receives firewall rules from policy manager	**Policy syntax:** Packet signature to define allowable traffic for specified device **Policy source:** Proof-of-work blockchain; new packet signatures are submitted when trigger is detected

Table 1. Comparison of IoT network traffic mediation systems proposed in related work. The systems are compared on the basis of how each system implements functionality that falls within the responsibility of each device class defined in IFTAT. We compare only systems that modify their traffic mediation behaviour in response to detected trigger events. We later propose and implement additional examples of how IFTAT device classes can be instantiated.
† LeakyPick does passive detection, but also suggests the option of active prevention.

providing updated classifier models by collecting and labeling data to periodically re-train the classifier.

SERENIoT [22] uses a public proof-of-work blockchain that can be queried to retrieve the allowable network traffic signatures for a given IoT device. Nodes on the blockchain, called Sentinels, submit summaries of observed device behaviours that get added to the blockchain if the majority of nodes have also observed the same behaviour. New behaviours resulting from firmware updates would thus be observed by the majority of nodes and added to the blockchain, whereas malicious behaviour resulting from device compromise would not. Sentinels allow any newly-connected device on the network to send and receive all traffic for a one-minute period; this is used as a profiling phase to identify the device type if possible and to determine its required network traffic.

5 Trigger Events Identified and Proposed

Table 2 lists examples of trigger events that can be used in IFTAT. We categorize the examples by the source from which the trigger event is derived: (i) physical measurements; (ii) software-based determination that a condition has been satisfied; or (iii) signature- or heuristic-based detection. We list both generic techniques and techniques proposed in academic literature that are suitable for use as trigger events.

deGraaf et al. [2] propose a cryptographic protocol that operates via port knocking to authenticate users prior to allowing application traffic through the firewall. VibLive [23] is a secure continuous liveness detection technique, using a microphone and loudspeaker, to ensure the user is present when giving voice commands. We also propose a technique that, to the best of our knowledge, has not previously been used for access control decisions: detection of when a user opens or closes a specific application on their smartphone. He et al. [6] conducted a survey of techniques for using physical sensors to detect home contexts relevant to security-related decisions, such as user presence, user identity, or home emergencies—these techniques can also be used to define trigger events.

Trigger source	**Trigger event**	**Implementation**
Physical	Motion detected	‡
	Door or window opened or closed	—
	Smoke detected, water detected, etc.	—
Software	Mobile application opened or closed	‡
	User authenticated (e.g., to WiFi network)	—
	Timer expired	‡
Signature	Phrase or word spoken by user	[13]
	Liveness detection	[23]
	Network traffic matched an activity signature	[17, 22]

Table 2. Categorization of trigger events suitable for use in IFTAT.
‡ denotes trigger events implemented in this paper; — denotes trigger events listed as examples but not implemented.

6 System Implementation

We implemented a proof-of-concept of IFTAT on a small test network to demonstrate two use cases as follows.

UC1. A home owner wishes to deny network access to a smart doorbell except for while a person is physically present in front of the device. This reduces the time that the device is allowed outgoing network connections to align with the user's intended usage of the device (e.g., for communicating with a person at their front door). Human presence should be determined without relying on the untrusted device, so we use a separate motion sensor for this task.

UC2. A business owner wishes to deny network access to a security camera except for while an authorized user is using a mobile application to access the camera feed. This prevents the device from being accessible to the Internet while the user does not need to access it. The user's smartphone serves as the trigger device that reports when the user has launched the mobile app.

6.1 Hardware Overview

We use two Raspberry Pi 4 devices[3]: one of them running Home Assistant[4] to function as the policy manager, and the other running OpenWRT[5] to function as the policy enforcer. An Energizer Connect EOD1-1002-2002-SIL Smart Doorbell and a ReoLink RLC-410-5MP security camera are designated as untrusted devices. Two trigger devices are also implemented, an iPhone 12 Pro and a AM312 PIR motion sensor running ESPHome[6] firmware. Fig. 3 depicts the network connectivity between all devices. The policy enforcer has three network interfaces (WAN, LAN, and WLAN) and performs routing, NAT, and packet filtering.

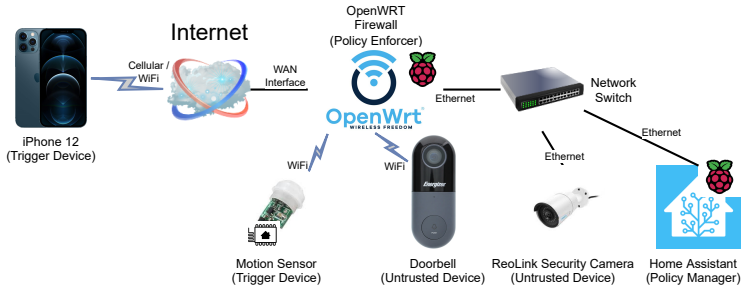

Fig. 3. Connectivity diagram of devices in the proof-of-concept implementation. Each device is labelled with its device class (untrusted device, trigger device, policy enforcer, or policy manager).

Devices are assigned a static IP address to ensure that security policies are applied to the correct untrusted devices. Alternative techniques can be used to identify untrusted devices dynamically, e.g., via device fingerprinting [10, 12, 22].

6.2 Policy Manager

We implement IFTAT policies in YAML using Home Assistant automation rules; for each rule we specify a trigger event and a corresponding traffic mediation action to be taken. Table 3 describes the policies we defined to implement **UC1** and **UC2**.

[3] https://www.raspberrypi.org/
[4] https://www.home-assistant.io/
[5] https://openwrt.org/
[6] https://esphome.io/

Rule	Trigger event	Traffic mediation action
UC1a	Motion detected near doorbell	Allow outbound connections from the doorbell to the Internet
UC1b	Ten minutes elapsed since motion detected near doorbell	Deny all outbound connections from the doorbell to the Internet
UC2a	User launched security camera mobile app	Allow inbound connections from the Internet to the security camera
UC2b	Ten minutes elapsed since security camera mobile app launched	Deny inbound connections from the Internet to the security camera

Table 3. Description of IFTAT policies used to implement proposed use cases **UC1** and **UC2**. The lettered suffixes distinguish between the two policies required for implementing each use case.

The traffic mediation actions are taken by issuing a command over an SSH connection to the policy enforcer. In a production-ready IFTAT implementation, the policy manager would automatically translate each action into a series of commands that the policy enforcer would understand. In our proof-of-concept implementation, we manually create a shell script for each action that enables or disables the iptables rules necessary to execute the action, e.g., allowing or denying network access to the camera. A sample policy in YAML format to implement **UC2** is shown in Fig. 4.

To allow remote trigger devices to report trigger events, we use webhooks on the policy manager. Each webhook is an HTTP URL with an embedded bearer token to ensure that only authorized trigger devices (e.g., the iPhone) can report trigger events. The webhook must be served over HTTPS to ensure that the bearer token cannot be eavesdropped. Trigger devices within the local network utilize Home Assistant's ESPHome integration to monitor the motion sensor's state over the WiFi network.

```
alias: Trigger Security Camera Policy
trigger:
  - platform: webhook
    webhook_id: trigger-security-camera-elS4xo2eEykNlqS0GrlXeCvr
action:
  - service: shell_command.enable_camera
  - delay:
      minutes: 10
  - service: shell_command.disable_camera
mode: restart
```

Fig. 4. A sample security policy to implement **UC2**.

6.3 Trigger Events

We implement three trigger events, which are detected as described below.

Mobile application opened. This event is reported by the iPhone using the Shortcuts app. We create a shortcut that performs the following actions:

1. Send an HTTP POST request to the webhook exposed by the policy manager (using the "Get contents of URL" action).
2. Launch the VLC App (using the "Open App" action).

The first action allows the phone to notify the policy manager that the user intends to interact with the untrusted device (security camera) via the VLC app. This trigger causes the policy manager to modify the active policy on the policy enforcer to allow network access to the untrusted device. The second action launches the application.

Motion present. This event is reported by the motion sensor. Home Assistant presents a binary state (i.e., "on" for motion present or "off" not present) for the motion sensor, which we monitor for changes from the "off" to the "on" state to determine when the event has occurred.

Timer expired. This event was implemented in Home Assistant by configuring a timer to expire 10 minutes after either a **Mobile application opened** or **Motion present** event is detected. Each of these two event types has its own timer, which resets to 10 minutes if a new event of that type is detected before the timer expires.

6.4 Policy Enforcer

Traffic mediation actions are implemented using iptables rules and the OpenWRT UCI system. When blocking network access to a device, we ensure that any active connections are terminated immediately by blocking all ESTABLISHED connections as well. The ReoLink security camera and Energizer Doorbell are configured as untrusted devices and by default will have all network access denied unless the **Mobile application opened** or **Motion present** events are detected.

Mediating traffic between devices on the same LAN requires using different LAN segments (e.g., using separate VLANs) or using Software Defined Networking as in HomeSnitch [17]. In the absence of such mechanisms, devices within the same LAN segment can communicate with each other without restriction.

6.5 Performance Evaluation

The time delays between the events in Fig. 1 have practical implications for creating IFTAT policies, since trigger events should be detected and the resulting traffic mediation action should take effect before the untrusted device requires network access. To evaluate the practicality of the policies we implemented, we collected the following timestamps to compute the aforementioned delays:

1. When the trigger event is detected by the trigger device
2. When the trigger event report is received by the policy manager
3. When the traffic mediation instructions are received by the policy enforcer
4. When the change in network access is detected by the untrusted device

All devices were synchronized with the same NTP server to ensure consistent timestamps, and all delay calculations were averaged across ten runs. Fig. 5 provides a timeline of events that includes the delays computed from the above timestamps that were collected for the enforcement of **UC1a**. We draw comparisons below with delays observed for the other policies.

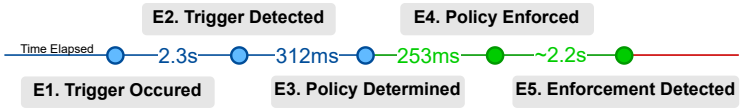

Fig. 5. A timeline for enforcing security policies when a trigger is received for **UC1a**.

E1-E2. For **UC1a**, after motion occurs, there is a small delay before it is detected by the motion sensor; this delay is sensor-dependent. As per the AM312 datasheet, our motion sensor has an activation delay of 2.3 seconds. For the remaining three use cases (**UC1b**, **UC2a**, **UC2b**), this delay is negligible.

E2-E3. For **UC1a** and **UC2a**, this delay was ∼300ms. This is the time taken for Home Assistant to receive an external trigger and process the policy (e.g., see Fig. 4) to determine the command to send to the policy enforcer. For **UC1b** and **UC2b**, this delay is negligible since the timer trigger is implemented directly on the policy manager.

E3-E4. All four policies from Table 3 consistently resulted in a delay of ∼250ms; this is the time taken to establish an SSH connection to the policy enforcer and execute the shell script to enforce the policy.

E4-E5. For **UC1a**, a delay of ∼2.2s was measured before the Energizer mobile app (which tracks the doorbell status through a cloud backend) would identify the doorbell as online. In contrast, for **UC2a**, an RTSP connection could immediately be opened from the mobile app to the camera upon the policy being enforced, since the camera is accessed via a direct connection (i.e., the delay was negligible). For both **UC1b** and **UC2b**, a cloud provider may cause additional delay to identify the device as offline after failing to receive several consecutive heartbeat messages. For **UC1b**, connectivity to the doorbell resulted in immediate disconnection from the video stream, but the doorbell was not reported offline by the mobile app for ∼2.5s. In contrast, for **UC2b**, the camera was immediately identified as offline by the mobile application.

Finally, we investigate the impact of the firewall rule table being updated each time a trigger event report is received. To test whether network performance is impacted, we used iperf3[7] to send a fixed number of UDP packets at a rate of 75Mbps to the policy manager from outside the local network (i.e., through the WAN interface of the policy enforcer, which is the LAN-to-WAN gateway in our implementation). In a control test with no rule updates, the average measured round-trip delay and jitter between the policy manager and the external test

[7] https://iperf.fr/

device was 0.587ms and 0.1106ms, respectively; no packet loss occurred and the TCP state table was preserved. To measure the impact of rule updates, we run a shell script that updates the firewall rules by successively adding and removing the rules for **UC1** in a loop; we measured that the script executed 40 iterations of the loop per second, with each iteration taking ∼25ms to complete. We repeated the iperf3 test while the aforementioned shell script was running, and observed an average round-trip delay and jitter of 0.589ms and 0.1134ms, respectively; no packet loss occurred and the TCP state table was preserved. We thus conclude that even under an unrealistically high rate of firewall rule updates as described above, the impact on network performance is negligible.

7 Discussion

Herein we discuss how IFTAT can strengthen a network's security and we discuss avenues for standardization that would support the security objectives of IFTAT.

7.1 Security Considerations

We discuss how IFTAT can combat the spread and operation of IoT malware (refer to **T1**) and the exfiltration of privacy-sensitive data (refer to **T2**). We also discuss security considerations relating to the implementation of trigger devices.

Protecting externally-exposed devices against compromise. This is the primary threat targeted by **UC2** with the ReoLink security camera. In this use case, IFTAT ensures that the device is only externally accessible when required by the user. Externally accessible devices are regularly targeted by botnets via IP scanning, causing any online and vulnerable devices to be infected by malware [1]. IFTAT reduces the likelihood of infection, as the device can only be scanned for a short period of time following a valid trigger event.

While home Internet gateway devices typically block incoming connections by default to all devices on the home network, they provide user interfaces to open ports to target devices. IFTAT offers the alternative of only openings ports on a temporary basis in response to trigger events that reflect a legitimate user's attempt to access the target devices. Moreover, IFTAT can leverage the following additional measures to further enhance the security of **UC2**:

i. The inclusion of an IP address in an allow list, ensuring only the mobile device which performed the trigger is able to access the camera, this reduces the risk to the levels of protection applied to the web hook bearer token.
ii. Incoming traffic to the policy enforcer targeting the untrusted device could be collected while the untrusted device has been denied network access. Using this data, signatures can be created representing traffic patterns sent to the untrusted device in the absence of trigger events. This signature can then be used to block potentially malicious traffic when the device is later allowed network access.

Preventing leakage of sensitive information. This is the primary threat targeted by **UC1**. IFTAT restricts the device's outgoing network communication to only the time periods that align with the user's intended usage of the device. This reduces unnecessary opportunities for leaking sensitive information to the device manufacturer or third-party trackers [8], and prevents the device from performing other outbound malicious activities [1, 11]. However, devices which are reliant the manufacturer's cloud service may react differently to being connected and disconnected from the Internet [16], e.g., by caching events locally and sending them to the cloud when connectivity is restored. Techniques employed by OConnor et al. [16, 17] may be used to determine how devices behave when they lose connectivity, and this can inform the creation of IFTAT policies.

Importance of countermeasures against trigger device spoofing. Should a trigger device be spoofed or compromised, it may be possible for an attacker to artificially signal the occurrence of a trigger event to allow network access to a target untrusted device. Thus, it is critical to secure the communication channel between the trigger devices and the policy manager. This can be achieved by reporting trigger events to the policy manager over an encrypted channel, e.g., a TLS connection, or over a channel that is inaccessible to untrusted devices, e.g., Zigbee or USB. This limits the attack surface that could be used by an attacker to compromise a trigger device and use it to allow network traffic to an untrusted device at will. However, the security of these channels may not be perfect, and the risk of trigger device spoofing remains present in Zigbee devices as well [20], especially if an attacker has physical access to the environment.

7.2 Standardizing IFTAT

Manufacturer Usage Descriptions (MUD) [9] provides a policy language that device manufacturers can use to define a profile of the network access (e.g., protocols, port numbers, destination IP addresses or hostnames) that the device requires. A device's MUD profile can then be used to restrict its network access and reduce its attack surface. Since MUD profiles are provided by the device manufacturer, they would be expected to be more accurate than network traffic profiles that are learned via traffic analysis as is done in aaa related work [17, 22] discussed in Section 4. The primary obstacle in the use of MUD profiles is its limited adoption thus far by device manufacturers. Should MUD be more widely adopted, IFTAT can use them to enforce more granular policies: when a trigger event occurs, an untrusted device could be allowed only the network access as defined by its MUD policy (instead of allowing unrestricted access), thereby minimizing the attack surface of the device.

MUD could also be extended to support the concept of trigger devices. For example, a device's MUD profile could specify if certain types of network access is required in response to external trigger events. For example, the MUD may define the network access required by the device when a **New firmware published** or **Motion present** event occurs. The former can be detected by polling

the manufacturer's website to monitor for announcements that a new firmware version has been released. The latter can be detected by allowing the user to select a motion sensor device on the network to use as the trigger device.

8 Conclusion

IoT users deserve more control over the devices they own. To gain this control, users must currently manage complex networking setups or manually add/remove firewall rules. Due to the dynamic nature of many IoT devices, this network management requires constant supervision and adjustment to not interfere with device functionality.

This paper presents IFTAT, a framework that gives users simple, granular, time-restricted control over the network connectivity of untrusted devices on their networks. Narrowing this connectivity time window substantially reduces the amount of information that can be leaked, as well as the exposure window during which devices are vulnerable to attack or misuse. As demonstrated, IFTAT can be deployed using existing IoT devices, hubs, and network infrastructure to create and manage policies, requiring no costly new equipment or backend cloud services. We hope that IFTAT and future time-based access control systems based on our framework offer users peace of mind when bringing new, potentially untrusted devices into their networks.

Acknowledgements. The second and third authors acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC) through the Discovery Grant program.

References

1. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., Zhou, Y.: Understanding the Mirai botnet. In: USENIX Security Symposium (Aug 2017)
2. deGraaf, R., Aycock, J., Jacobson, M.: Improved port knocking with strong authentication. In: Annual Computer Security Applications Conf. (2005)
3. Dong, K., Zhang, Y., Zhao, Y., Li, D., Ling, Z., Wu, W., Zhu, X.: Real-time execution of trigger-action connection for home Internet-of-Things. In: IEEE INFOCOM 2022 - IEEE Conf. on Computer Communications (2022)
4. Goutam, S., Enck, W., Reaves, B.: Hestia: simple least privilege network policies for smart homes. In: ACM WiSec (2019)
5. Hamza, A., Ranathunga, D., Gharakheili, H.H., Roughan, M., Sivaraman, V.: Clear as MUD: Generating, validating and applying IoT behavioral profiles. In: ACM SIGCOMM Workshop on IoT Security and Privacy (2018)
6. He, W., Zhao, V., Morkved, O., Siddiqui, S., Fernandes, E., Hester, J., Ur, B.: SoK: Context sensing for access control in the adversarial home IoT. In: IEEE EuroS&P (2021)

7. Herwig, S., Harvey, K., Hughey, G., Roberts, R., Levin, D.: Measurement and analysis of Hajime, a peer-to-peer IoT botnet. In: NDSS Symposium (2019)
8. Huang, D.Y., Apthorpe, N., Li, F., Acar, G., Feamster, N.: IoT inspector: Crowdsourcing labeled network traffic from smart home devices at scale. Proc. of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies **4**(2), 1–21 (2020)
9. Lear, E., Droms, R., Romascanu, D.: Manufacturer usage description specification. RFC 8520 (Mar 2019)
10. Marchal, S., Miettinen, M., Nguyen, T.D., Sadeghi, A., Asokan, N.: AuDI: Toward autonomous IoT device-type identification using periodic communication. IEEE Journal on Selected Areas in Communications **37**(6), 1402–1412 (2019)
11. Mi, X., Feng, X., Liao, X., Liu, B., Wang, X., Qian, F., Li, Z., Alrwais, S., Sun, L., Liu, Y.: Resident evil: Understanding residential IP proxy as a dark service. In: IEEE Symposium on Security and Privacy (2019)
12. Miettinen, M., Marchal, S., Hafeez, I., Asokan, N., Sadeghi, A.R., Tarkoma, S.: IoT sentinel: Automated device-type identification for security enforcement in IoT. In: Conf. on Distributed Computing Systems (ICDCS) (2017)
13. Mitev, R., Pazii, A., Miettinen, M., Enck, W., Sadeghi, A.R.: LeakyPick: IoT audio spy detector. In: Annual Computer Security Applications Conf. (2020)
14. Newman, L.H.: An elaborate hack shows how much damage IoT bugs can do. WIRED (Apr 2018), https://www.wired.com/story/elaborate-hack-shows-damage-iot-bugs-can-do/
15. Nguyen, T.D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N., Sadeghi, A.: DÏoT: A federated self-learning anomaly detection system for IoT. In: IEEE International Conf. on Distributed Computing Systems. pp. 756–767 (2019)
16. OConnor, T., Enck, W., Reaves, B.: Blinded and confused: uncovering systemic flaws in device telemetry for smart-home internet of things. In: Conf. on Security and Privacy in Wireless and Mobile Networks. pp. 140–150 (2019)
17. OConnor, T., Mohamed, R., Miettinen, M., Enck, W., Reaves, B., Sadeghi, A.R.: HomeSnitch: Behavior transparency and control for smart home IoT devices. In: ACM WiSec (2019)
18. Ruiz, C., Pan, S., Bannis, A., Chang, M.P., Noh, H.Y., Zhang, P.: IDIoT: Towards ubiquitous identification of IoT devices through visual and inertial orientation matching during human activity. In: IEEE/ACM International Conf. on Internet-of-Things Design and Implementation (2020)
19. Seymour, W., Kraemer, M.J., Binns, R., Van Kleek, M.: Informing the design of privacy-empowering tools for the connected home. In: ACM Conf. on Human Factors in Computing Systems (2020)
20. Talakala, G.H., Bapat, J.: Detecting spoofing attacks in Zigbee using device fingerprinting. In: IEEE Annual Consumer Communications Networking Conf. (2021)
21. The Associated Press: Your smart fridge could be mining bitcoins for criminals. CBC News (Jun 2018), https://www.cbc.ca/news/science/bitcoin-hacking-smart-devices-1.4728222
22. Thomasset, C., Barrera, D.: SERENIoT: Distributed network security policy management and enforcement for smart homes. In: Annual Computer Security Applications Conf. (2020)
23. Zhang, L., Tan, S., Wang, Z., Ren, Y., Wang, Z., Yang, J.: VibLive: A continuous liveness detection for secure voice user interface in IoT environment. In: Annual Computer Security Applications Conf. (2020)