SHIFT: a Security and Home Integration Framework for IoT

Katharina O. E. Müller, Daria Schumm, Elliott Wallace, Bruno Rodrigues, Burkhard Stiller
Communication Systems Group CSG, Department of Informatics IfI, University of Zürich UZH
Binzmühlestrasse 14, CH—8050 Zürich, Switzerland
E-mail: [muller¦schumm¦rodrigues¦stiller]@ifi.uzh.ch, elliott.wallace@uzh.ch

NOMS 2024-2024 IEEE Network Operations and Management Symposium | 979-8-3503-2793-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/NOMS59830.2024.10575751

***Abstract*—With the growing popularity of Internet-of-Things (IoT) and the smart home market, third party data stores and remote cloud environments, security and privacy of personal data is the central problem. Since the existing Privacy Enhancing Technologies (PET) have yet to provide a lightweight solution to safeguard privacy within smart homes, this paper presents the Security and Home Integration Framework for IoT (SHIFT). SHIFT leverages an open-source architecture to integrate existing tools, including home assistants and network monitors, into a unified framework designed to monitor and control smart home devices through security policies. The proof-of-concept prototype for monitoring and controlling smart home device communications using user-defined policies successfully monitors device activity, processes data, and enforces user policies in a live smart home environment. SHIFT gives users control over their devices and data while balancing convenience with privacy and security.**

***Index Terms*—Internet of Things, Smart Home, Security, Sensing, Policy-based Management**

I. Introduction

The growth of the smart home market is driven by technological integration into our daily lives and the convenience of smart devices [1]. This growth can be observed in the segment's increased revenue, with reported rate of revenues in the United States accumulating to 31.5% in a Compound Annual Growth Rate (CAGR) from 2017 until 2022 [2], and global projections pointing to a CAGR of 27.4% from 2022 to 2027 [3]. The main goal of homeowners is to simplify and automate routine tasks, such as regulating temperatures [4], [1], leading to increased data collection and processing, requiring even more resources on already constrained devices. Consequently, to reduce the computational burden and avoid additional devices, many manufacturers opt to send locally collected data (*i.e.*, personal data) to a cloud service for large-scale processing. While this approach simplifies the services offered by the manufacturer and is more convenient for the end-user, it exposes sensitive data to the risk of a potential data breach and unintended uses [5], [1]. Additionally, once the data is shared with an external service, it exposes unsecured devices to the Internet.

Despite continuous improvements by manufacturers, security and privacy problems associated with smart devices are still widely recognized in the literature, such as [1], [4], [6], [5], [7]. Currently, no framework provides a holistic view of the personal data exchange between devices, gateways, and external servers. While manufacturers prefer to integrate their products with automation solutions, such as Apple (HomeKit) [8], Amazon (Alexa Smart Home) [9], and Google (Home) [10], challenges such as personal data handling, malware prevention, data corruption, and service unavailability, persist.

This paper presents the Security and Home Integration Framework for IoT (*SHIFT*), a lightweight system that provides enhanced visibility and control over smart home devices for end-users. Hence, it presents the solution to increase visibility by observing network traffic and Domain Name System (DNS) requests, while the control is achieved through a Policy-Based Management (PBM) architecture by denying or allowing specific network traffic. *SHIFT* improves end-user control over data and is suitable for smaller devices, such as Raspberry Pis, and does not require high computational power. Additionally, this paper provides an open source *SHIFT* prototype [11], [12] (source code and Docker image) for monitoring smart home DNS queries, providing users with insights into device communication and the smart home environment, and enabling enforcement of user-defined domain policies.

This paper is organized as follows. Section II discusses related work that address smart home security and privacy concerns. Section III introduces *SHIFT* prototype, its application scenario, and followed by experiment results in Section IV. An extensive evaluation outlined in Section V and is complemented by conclusions and future work in Section VI.

II. Related Work

As the number of different protocols, technologies, and application scenarios for smart home devices continues to grow, so does the potential attack surface for smart homes [13]. The heterogeneous nature of IoT and smart home devices further weakens the security of such systems. Additionally, the short time-to-market and cost reductions for IoT devices contribute to the weakened hardware and software security [14]. As a result, security and privacy of IoT communications is the central topic of current research.

A. Smart Home Security

State-of-the-art smart home security is constantly evolving, and many new technologies are being developed to

improve security and privacy standards. According to [15], [4], many thermostat sensors/actuators and IP cameras (*i.e.*, small sensors) cannot be patched owing to hardware limitations. This is a regular occurrence even as of today, particularly in products of non-reputable mass-market manufacturers [16]. As pointed out by [1], many devices are shipped with default usernames and passwords (*e.g.*, admin/admin, root/12345) that are an easy target for hackers or, due to hardware restrictions, passwords are often of poor entropy, making brute-force assaults conceivable. Moreover, [17] outlines that the lack of experienced administrators, matched with the unique and often complex configuration environment provided by each manufacturer, complicates securing devices for inexperienced users.

B. Data Privacy

Data privacy is a challenge associated with insufficient device security levels pertaining to data confidentiality. In this sense, several surveys [15], [1], [4] unanimously present this aspect as a major challenge, resulting from the manufacturers' various security levels, the devices' low hardware capacity to store and process data locally, the exploitation of upstreamed and anonymized data, and the limited data privacy regulations. Overall, the key is a consistent trade-off between advantages offered by smart home devices and privacy breaches caused by the collection of personal data.

C. Comparable Approaches

App activity control, such as HomePad [18], is one of the privacy preserving approaches. HomePad [18] targets hub-based application activity control and determines how smart home applications access and process sensitive data. The solution compares data flows to user-defined privacy policies and only executes applications that are compliant [18]. However, it requires the implementation of HomePad applications and flow graph modeling, making it unsuitable for layman end-users.

Data minimization strategies offer another approach that lead to the reduction of shared data to preserve privacy [19]. [20] proposes a per-user privacy policy controls enforcement system, unlike per-app privacy policy solutions, thus, reducing system complexity and allowing the establishment of uniform policies across all devices. The solution is implemented on the device and analyses data in plain text. However, it requires an Artificial Intelligence (AI) agent to disable and modify data streams directly at the device level.

Similarly, Peekaboo [21] proposes a hub-centric architecture to minimize outgoing data. Hence, application developers specify data collection behaviors and pre-processing pipelines, before data is sent to external cloud servers, while end-users and external auditors can analyze the data behavior. Likewise, [22] proposes a data-mediator component between devices and a hub that automatically transforms application semantics to data-minimization policies. The solution works on encrypted traffic by taking a man-in-the-middle role and aims to provide more control to users by allowing privacy policy prioritization [22].

[23] developed a prototype focused on limiting non-essential device traffic and blocking non-required destinations without breaking functionality. The system is located at an IP router and establishes firewall rules to automate the process. Similarly, [24] enables smart home device users to gain insights into data being exchanged and to allow users to make informed decisions about their devices. Lastly, [25] proposes a methodology targeting users of wearable smart devices to block unnecessary network communications between a device and the Internet.

In contrast, *SHIFT* is a hub-centric approach that minimizes and blocks data traffic to external parties. Unlike other solutions, SHIFT is lightweight and requires very little technical know-how from the end-user, not requiring a device-by-device implementation. Additionally, *SHIFT* allows for integration with existing platforms, such as Home-Assistant, rather than comprising a stand-alone solution.

III. *SHIFT* Prototype

SHIFT is designed for seamless integration into existing smart home environments. Its functionality relies on a local network DNS communication, making it independent, but adaptable to various smart home platforms, such as Home Assistant and Apple Homekit. At its core, the *SHIFT* comprises a lightweight web application and a Pi-hole instance, serving as the network's primary DNS server. Both components are engineered for hosting on a System-on-Chip (SoC) platform, such as Raspberry Pi, within the local network. The interaction between these elements facilitate the system's key functions, including monitoring and policy enforcement. This design ensures flexibility and compatibility with diverse smart home configurations.

A. Architectural Components

SHIFT is based on the multi-component architecture. Figure 1 shows the prototype environment and relationships between components. It is further described as front- and back-end components.

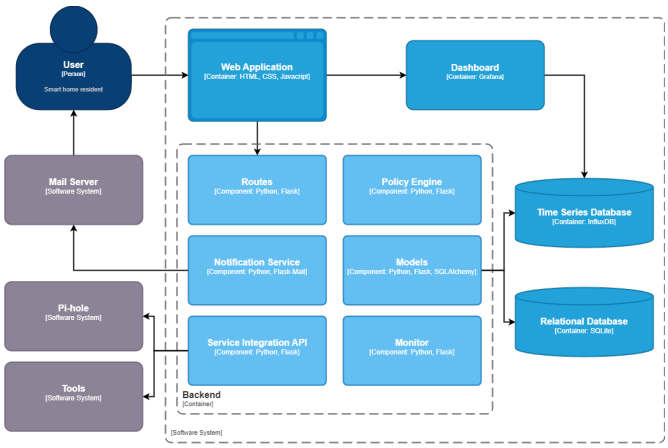

Fig. 1: System Components

Web Application: serves as the user interface for interacting with the system via a web browser.

Dashboard: user interface feature that presents data visualizations and device-network interaction statistics.

Routes: functions as the entry point for the backend, handling incoming HTTP(S) requests from the web application's frontend.

Service Integration API: serves as a consumer for interfacing with various software systems, primarily integrating with the Pi-hole HTTP API within the *SHIFT*'s scope.

Monitor: carries out scheduled tasks to oversee network activity, such as periodically initiating queries to the Pi-hole API for monitoring purposes.

Models: defines the system's data models, encapsulating all database-related functionality.

Policy Engine: evaluates user-defined policies using observed metrics from recurring monitoring.

Notification Service: generates and delivers reports on smart device behavior to users.

Relational Database: stores data concerning users' properties, policies, and devices.

Time Series Database: stores device networking information collected from various data sources in a unified format.

Mail Server: delivers notifications to users via a standard email service.

1) Frontend: Home users access the application through a web-based frontend with an administration interface and a dashboard. Users can register, log in, manage smart home devices, and define device policies in the administration interface, which can be reviewed and edited. The dashboard presents statistics and data visualizations of device network behavior. It is integrated into the web application and relies on Pi-hole's API as the primary data source.

2) Backend: The backend houses the core business logic operating on a lightweight web server. It manages data aggregation, storage, retrieval, incoming frontend requests, integration with third-party services, monitoring, policy evaluation, and user notification generation. *SHIFT* uses a Policy-Based Management (PBM) framework that can help to reduce complexity in observing how devices exchange personal data and enforce security policies (*e.g.*, determining that specific or all devices should not send data externally).

The first step in SHIFT's operation involves configuring user devices, policies, and tools. While most devices offer specific integration APIs via a bridge or at the device, an efficient alternative is to rely on APIs provided by home automation services that already interface with several devices. Additionally, *SHIFT* allows integrating sensing tools to sniff standalone devices, such as smart tags. The second step involves configuring network tools, such as the DNS service and network sniffing, to observe real-time traffic. The third step involves configuring security policies via a web dashboard or smartphone application to define the extent of device traffic control.

Upon configuring a policy, the Policy Engine (PE) will match against policy conflicts and store it in the database. The PE retrieves active policies in the database and matches them with metrics collected in the monitor. The monitor operates in sliding time windows based on data pulled by devices via the network or fetched via API. Upon breach of defined policies, the PE enforces a policy based on actions available by configured tools. In principle, *SHIFT* is intended to observe and block DNS requests based on a DNS proxy service, such as Pi-hole [26].

The Application Server container is the central component of Web application. It leverages a Web server gateway to run the Flask[1] web application and accept HTTP requests. In front of the application server, a Web server container is configured to act as a reverse proxy. It handles incoming client requests from the network by directly serving static content for the web application, forwarding requests to the application server, and returning the application server's response.

The reverse proxy is solely configured to forward requests from the local network. It thereby helps to shield the web app running on the application server, which does not directly expose any ports outside the docker environment. Another key part of the system is the Pi-hole [26] instance running in a separate docker container. The Application Server instance communicates with the Pi-hole container via API on port 80 using the Docker internal network [12]. Pi-hole also accepts DNS requests from the host network on port 53.

The three container instances form the core *SHIFT* system around the web application. In an extended configuration, another container running an instance of the time series database is included. As *SHIFT* currently uses only one data source to monitor device behavior, it is not dependent on this database to store, merge, and aggregate data from different sources.

IV. Results and Evaluation

This section outlines the results and technical aspects of the prototype implementation, covering the backend and the interactive frontend elements. A *Raspberry Pi 3 Model B Plus Rev 1.3* was used to host the prototype system. Additionally, this section delves into the prototype evaluation, wherein it was deployed in a live smart home environment to assess its functionalities, including monitoring, policy enforcement, email notifications, and device operability under restricting policies. Furthermore, performance data was collected to evaluate its suitability for resource-constrained SoC platforms.

A. Monitoring

Traffic monitoring entails observing DNS requests made by a smart home device via Pi-hole API. To test this capability, a new Netatmo air quality monitor device was added to

[1]https://flask.palletsprojects.com/en/3.0.x/

the *SHIFT* database, and multiple monitoring cycles were run at predefined intervals of 15 minutes. After four cycles, the Reporting Database was examined to determine the performance of the monitoring capability and whether the newly configured device was considered.

TABLE I: Reporting Database After Four Monitoring Jobs

data_source	interval_start	interval_end	total_queries	unique_domains	queries_blocked	evt_create
pi_hole	2023-07-25 12:00	2023-07-25 12:15	1	1	0	2023-07-25 12:15
pi_hole	2023-07-25 12:15	2023-07-25 12:30	1	1	0	2023-07-25 12:30
pi_hole	2023-07-25 12:25	2023-07-25 12:45	1	1	0	2023-07-25 12:45
pi_hole	2023-07-25 12:35	2023-07-25 13:00	2	1	0	2023-07-25 13:00

Each successful monitoring job added a record to the database, as depicted in Table I, indicating it ran four times as intended. The air quality monitor generated five queries during this period, and the DNS requests were successfully recorded. Furthermore, Table II compared the Reporting Database results to the Time Series Database. The data indicates five DNS query records for one client, identified by the air quality monitor's IP address. This is consistent with the information from the Reporting Database table and listed in Table I, with the `_value` column representing the domain that the respective client requested.

TABLE II: Time Series Database After Four Monitoring Jobs

client	query_type	reply_type	_value	_time
192.168.1.161	A	3	netcom.netatmo.net	2023-07-25 12:15
192.168.1.161	A	3	netcom.netatmo.net	2023-07-25 12:25
192.168.1.161	A	3	netcom.netatmo.net	2023-07-25 12:35
192.168.1.161	A	3	netcom.netatmo.net	2023-07-25 12:46
192.168.1.161	A	3	netcom.netatmo.net	2023-07-25 12:56

Furthermore, the Pi-hole's administrator interface was used to validate the retrieved data points to check the query log for the Netatmo air quality monitor. Table III shows the query log for the examined time frame, filtered by the corresponding IP address. Comparing this information with Table I and the data points in Table II confirms that the monitoring job reliably queries Pi-hole's API, stores the data in the Time Series Database, and creates an aggregation in the monitoring Reporting Database.

TABLE III: Pi-hole Query Log

_time	query_type	_value	client	execution
2023-07-25 12:15	A	netcom.netatmo.net	192.168.1.161	OK (cache)
2023-07-25 12:25	A	netcom.netatmo.net	192.168.1.161	OK (answered by 1.1.1.1#53)
2023-07-25 12:35	A	netcom.netatmo.net	192.168.1.161	OK (cache)
2023-07-25 12:46	A	netcom.netatmo.net	192.168.1.161	OK (answered by 1.1.1.1#53)
2023-07-25 12:56	A	netcom.netatmo.net	192.168.1.161	OK (cache)

B. Policy Enforcement

Policy enforcement aims to demonstrate whether *SHIFT* successfully enforced user-defined policies. In this scenario, a new Philips Hue Bridge was added to *SHIFT*, and the default policy was set to of *allow all*, allowing all DNS domains. The subsequent monitoring jobs detected all domains for which the device sends a DNS request. The data was then passed from the Pi-hole DNS server to the Policy Engine (PE) component. Additionally, the PE added a default policy for each domain that lacked an existing one in the new device's Policy Database. At the next policy evaluation cycle, it was verified that the default policy for all domains was automatically created and changed to blocked for associated domains, followed by verification of policies initiated on the Pi-hole instance.

Enforcing user-defined policies requires the PE to update Pi-hole's domain list. These policies were converted into domain list rules, allowlisting for *allow* policies, and blocklisting for *block* policies. All domains were permitted throughout the test, except one, *www.ecdinterface.philips.com*, that was successfully blacklisted in Pi-hole.

This evaluation observed that the policies defined in the prototype application consistently transfer to the Pi-hole system. Pi-hole's DNS sinkhole functionality effectively blocks blocklisted domains, ensuring user-defined policies are enforced. Policy switching to *block* successfully updated the Pi-hole domain list entry to blocklist the domain, blocking it in subsequent requests.

C. Weekly Notification

A weekly notification system was implemented to improve communication visibility for end-users, requiring the configuration of an SMTP connection to the email service responsible for delivering the notifications generated by the Notification Service component. To use notification service, the necessary parameters needs to be specificed, including a mail server address, a port number, and account's username and password. These arguments are then passed to the application through the environment variables. Additionally, the prototype application registers a user with a valid email address and available smart home devices, such as air quality control monitor, a smart TV and a Philips Hue Bridge. The weekly email notification process is activated by a cron job, taking into account the data of each execution and always including the preceding seven days up to the time of execution.

Emails were sent automatically by decreasing the number of worker processes and increasing the request timeout to 180 seconds. The resulting weekly email notification contains a table visualizing the configured smart home devices' behavior over the past week, as seen in Figure 2. The Figure 2 displays top ten domains of the three smart devices visited in the test. The domains are ranked by the number of times the respective device successfully sent a DNS request to resolve the domain.

Additionally, a graph is sent, as depicted in Figure 3, showing the average number of DNS requests made by each device at each hour of the day, increasing the transparency of smart home device behaviors throughout a regular day. The graph indicates that traffic is mostly unrelated to device usage. Although the number of DNS queries increases slightly during the day, it remains relatively constant at night, when devices are not actively used.

D. Smart Device Operability

Device operability aims to measure to what extent the device functionality can be limited by blocking certain

Smart Device	Domain	Visits
Philips Hue Bridge	www.ecdinterface.philips.com	154
Air quality monitor	netcom.netatmo.net	142
Smart TV	fcm.googleapis.com	98
Smart TV	nrdp.prod.cloud.netflix.com	89
Philips Hue Bridge	diag.meethue.com	85
Smart TV	push.prod.netflix.com	41
Smart TV	preapp.prod.partner.netflix.net	29
Philips Hue Bridge	mqtt.2030.ltsapis.goog	25
Philips Hue Bridge	data.meethue.com	17
Smart TV	occ-0-1697-1347.1.nflxso.net	6

Fig. 2: Weekly E-mail Notification Showing the Top 10 Domains

Fig. 3: Weekly E-mail Notification

domains and shielding them from the cloud server. To determine this, several experiments were conducted with the previously added smart home devices from Netatmo and Philips Hue to verify their functionality under policy constraints. In the first step, all policies for both devices were set to *block* the associated domains, as demonstrated by the policy definitions for the Philips Hue Bridge.

After successfully installing blocking policies, testing was conducted to assess data access and device controls for both devices using their Home Assistant integrations and respective mobile applications. It was found that Philips Hue was fully operational with Home Assistant, even without a cloud connection, and operational over Wi-Fi with the Hue app for Android, with no restrictions on functionality as long as the smartphone is on the same network. Hence, this setup's remote access is no longer possible through the cellular network. The Netatmo air quality monitor only calls up a single domain and is not operational with either integration. Due to this limitation, the device had only one policy option, which was set to *block*. Blocking this domain led to blocking all status updates to the cloud server. Consequently, the blocking policy rendered the air quality monitor inoperable. Since, both the Netatmo Home Assistant integration and the manufacturer's Android app, Home Coach, rely on data from the cloud service.

Thus, it was necessary to determine a minimum configuration for the Philips Hue Bridge, achieving a combination of policies that allow as few domains as possible while still enabling remote control via the Hue app. Through experimentation, it was found that only one domain, *www.ecdinterface.philips.com*, was required to enable the remote control functionality. Additionally, the policy for *time1.google.com* was changed to allow time synchronization, thus enabling time-based automation.

Unexpectedly, setting blocking policies for the Philips Hue Bridge resulted in a significant increase in the number of DNS queries that emanated from the device, with the number of queries with all domain blocked setting being up to 20 times higher than in the setting with all domain allowed.

E. Performance Evaluation

A performance evaluation was carried out to measure the resource utilization of each container running on the platform, the `docker stats` command was executed every three minutes. This was executed continuously for twelve hours under normal operational conditions to establish a performance baseline. During this initial evaluation, the following containers remained active throughout the test: the prototype application, the Time Series Database, the reverse proxy server, and the Home Assistant instance. The gathered data then underwent processing, including data cleaning and removing outliers using the z-score method. Subsequently, the dataset was visualized.

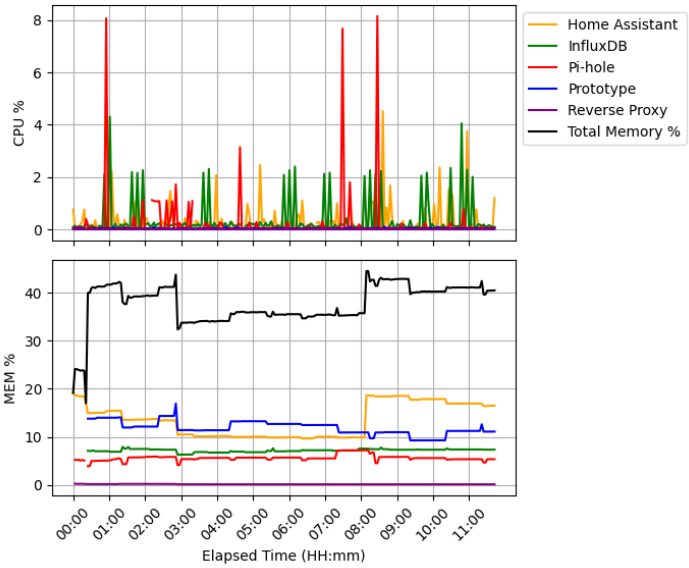

Fig. 4: Relative CPU and Memory Usage of Docker Containers

Figure 4 shows each containerized application's relative CPU usage. Indicating that, apart from a few spikes in the

course of the Pi-hole instance, the CPU load is in a relatively low range. Based on this observation, the Raspberry Pi's processing power seems sufficient to run the prototype application in this configuration. Furthermore, no performance problems were noticed while using the application.

Additionally, Figure 4 depicts the memory utilization of the docker containers during the baseline measurement. In contrast to the values observed for CPU usage, the total memory loads on the system were high. The largest contributors to these values were the Home Assistant instance and the prototype container. A constant load between 10-15% caused by the prototype application is rather at the high end for a lightweight application. However, there were also no major fluctuations in the course of the measurements, indicating a certain level of stability.

Furthermore, during the experiments on the productive Raspberry Pi system, it was observed that the available random access memory (RAM) on the platform was not always sufficient. This led to several events, resulting in an unresponsive system requiring a restart. To counteract this, a swap file with a higher capacity was configured. This significantly reduced the load on the RAM. Although access to the virtual memory of the swap file is much slower, no major performance losses were observed during testing.

V. Discussion

The first test case validates the prototype's ability to monitor DNS requests of smart home devices. Showing that the prototype allows users to add new smart home devices via the user interface. Furthermore, the results reveal that monitoring jobs, which receive data about DNS queries of the devices via the Pi-hole API, are automatically executed at regular intervals. Finally, the collected data points are reliably stored in a Time Series Database. Since this prototype utilized Pi-hole as the only data source for the network behavior, using a Time Series Database is not strictly necessary. However, using a Time Series Database is desirable in a more comprehensive system with multiple data sources. It facilitates the consolidation of the collected data in a uniform format and stores it for later processing (*e.g.*, to build reports or create data visualizations). In the presented case, one can query the Pi-hole API at any given time for this data, emhasising that saving the data twice is redundant. Nevertheless, the feature was implemented as a proof-of-concept and can be activated via configuration.

The goal of the second experiment, as described in the enforcement scenario, was to assess the system's capabilities to set and enforce simple *allow* and *block* policies. The test results confirm that the prototype application meets the specified requirements. The documentation validates that policies are automatically created for newly detected domains with the configured default values of the respective smart home devices. Furthermore, it was proven that users can easily change policies created via the user interface to influence behavior, and that the changes are successfully propogated to the Pi-hole system.

While it can be argued that this prototype offers an alternative interface for Pi-hole interaction, the extent to which this is true should be considered. The prototype's functionalities undeniably rely on Pi-hole, which provides a broader range of features and configurations, rendering it a potent tool. Nonetheless, Pi-hole is not known for its simplicity and is better suited for tech-savvy users. Conversely, the prototype targets typical smart homeowners aiming for increased transparency and privacy control. Moreover, it's designed to facilitate the integration of additional mechanisms for enforcing privacy policies beyond Pi-hole. In contrast to Pi-hole, the prototype automatically generates reports about the behavior of smart home devices, which are sent to the user via email. The results affirm that the prototype can generate a weekly report and send it to the user as an email notification. Testing this scenario revealed some pitfalls that could subsequently be resolved. Nevertheless, these difficulties highlight the challenges of implementing a solution for a platform with limited resources.

In addition to the integrated dashboard in the web UI, weekly email notifications provide an important source of information for users about the activities of their smart home devices. Therefore, they play a key role in achieving one of the main goals of this paper. The statistics and visualizations contained in the notifications make the behavior of the devices in the smart home observable and, thus, create increased transparency regarding the impact on privacy.

VI. Summary and Future Work

This paper presents SHIFT as a prototype to enhance user privacy in smart home environments [12], [11]. Considering the increase in the number of connected smart devices in the home and the constant communication of these devices with cloud services, there is a pressing need to look at how these devices and services affect the privacy and security of users. In this regard, SHIFT provides monitoring smart home device network traffic via DNS queries, processing collected data to make device communication transparent to users, and allowing users to set policies for device access. Integration with Pi-hole enables regular monitoring of DNS requests from smart home devices and enforces user-defined domain-blocking policies. However, it is important to note that this approach is limited to traditional DNS queries and does not cover DNS over HTTPS (DoH) requests.

To increase applicability, further research, development, and testing are needed to add a diverse range of smart home devices and examine the complexities of such setups. Additionally, the monitoring and policy enforcement mechanisms can be expanded through ARP spoofing and deep packet inspection, or integrating existing tools with such capabilities through APIs, could provide a more comprehensive view of smart home communications.

REFERENCES

[1] B. L. R. Stojkoska and K. V. Trivodaliev, "A Review of Internet of Things for Smart Home: Challenges and Solutions," *Journal of Cleaner Production*, Vol. 140, pp. 1454–1464, 2017.

[2] Jeremiah Lasquety-Reyes, "Smart Home Revenue Forecast in the World Until 2025," URL: https://www.statista.com/forecasts/887554/revenue-in-the-smart-home-market-in-the-world, Jan. 2023, Accessed: 2023-01-05.

[3] Grand View Research, "Smart Home Market Size, Share & Trends Analysis Report," URL: https://www.grandviewresearch.com/industry-analysis/smart-homes-industry, Jan. 2023, Accessed: 2023-01-05.

[4] N. Komninos, E. Philippou, and A. Pitsillides, "Survey in Smart Grid and Smart Home Security: Issues, Challenges and Countermeasures," *IEEE Communications Surveys & Tutorials*, Vol. 16, No. 4, pp. 1933–1954, 2014.

[5] M. Tabassum, T. Kosinski, and H. R. Lipford, "" I don't own the data": End User Perceptions of Smart Home Device Data Practices and Risks," *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*, 2019, pp. 435–450.

[6] S. Zheng, N. Apthorpe, M. Chetty, and N. Feamster, "User Perceptions of Smart Home IoT Privacy," *Proceedings of the ACM on Human-computer Interaction*, Vol. 2, No. CSCW, pp. 1–20, 2018.

[7] D. Geneiatakis, I. Kounelis, R. Neisse, I. Nai-Fovino, G. Steri, and G. Baldini, "Security and Privacy Issues for an IoT based Smart Home," *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2017, pp. 1292–1297.

[8] Apple Inc., "Developing Apps and Accessories for the Home," URL: https://developer.apple.com/apple-home/, Jan. 2023, Accessed: 2023-01-05.

[9] Amazon.com Inc., "Understand Smart Home Security Skills," URL: https://developer.amazon.com/en-US/docs/alexa/device-apis/overview-smart-home-security.html, Jan. 2023, Accessed: 2023-01-05.

[10] Google, "A Helpful Home is a Private Home," URL: https://safety.google/nest/, Jan. 2023, Accessed: 2023-01-05.

[11] Elliott Wallace, "shipp - Smart Home Integrated Privacy Protection," Accessed: 2023-08-09. [Online]: https://github.com/elduwa/shipp

[12] ——, "shipp - Dockerhub," Accessed: 2023-08-09. [Online]: https://hub.docker.com/r/elliottwallace/shipp

[13] B. Hammi, S. Zeadally, R. Khatoun, and J. Nebhen, "Survey on smart homes: Vulnerabilities, risks, and countermeasures," *Computers & Security*, Vol. 117, p. 102677, 2022. [Online]: https://www.sciencedirect.com/science/article/pii/S016740482200075X

[14] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating critical security issues of the iot world: Present and future challenges," *IEEE Internet of Things Journal*, Vol. 5, No. 4, pp. 2483–2495, 2018.

[15] C. Lee, L. Zappaterra, K. Choi, and H.-A. Choi, "Securing Smart Home: Technologies, Security Challenges, and Security Requirements," *2014 IEEE Conference on Communications and Network Security*. IEEE, 2014, pp. 67–72.

[16] H. Touqeer, S. Zaman, R. Amin, M. Hussain, F. Al-Turjman, and M. Bilal, "Smart Home Security: Challenges, Issues and Solutions at Different IoT Layers," *The Journal of Supercomputing*, Vol. 77, No. 12, pp. 14 053–14 089, 2021.

[17] J. Dahmen, D. J. Cook, X. Wang, and W. Honglei, "Smart secure homes: a Survey of Smart Home Technologies that Sense, Assess, and Respond to Security Threats," *Journal of Reliable Intelligent Environments*, Vol. 3, No. 2, pp. 83–98, 2017.

[18] I. Zavalyshyn, N. O. Duarte, and N. Santos, "Homepad: A privacy-aware smart hub for home environments," *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018, pp. 58–73.

[19] I. Zavalyshyn, A. Legay, A. Rath, and E. Rivière, "Sok: Privacy-enhancing smart home hubs," *Proceedings on Privacy Enhancing Technologies*, Vol. 4, pp. 24–43, 2022.

[20] N. Klingensmith, Y. Kim, and S. Banerjee, "A hypervisor-based privacy agent for mobile and iot systems," *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '19. New York, NY, USA, Association for Computing Machinery, 2019, p. 21–26. [Online]: https://doi.org/10.1145/3301293.3302356

[21] H. Jin, G. Liu, D. Hwang, S. Kumar, Y. Agarwal, and J. I. Hong, "Peekaboo: A hub-based approach to enable transparency in data processing within smart homes," *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 303–320.

[22] H. Chi, Q. Zeng, X. Du, and L. Luo, "Pfirewall: Semantics-aware customizable data flow control for smart home privacy protection," *CoRR*, Vol. abs/2101.10522, 2021. [Online]: https://arxiv.org/abs/2101.10522

[23] A. M. Mandalari, D. J. Dubois, R. Kolcun, M. T. Paracha, H. Haddadi, and D. Choffnes, "Blocking without breaking: Identification and mitigation of non-essential iot traffic," *Proceedings on Privacy Enhancing Technologies*, Vol. 2021, No. 4, pp. 369–388, 2021.

[24] F. Klement, H. C. Pöhls, and K. Spielvogel, "Towards privacy-preserving local monitoring and evaluation of network traffic from iot devices and corresponding mobile phone applications," *2020 Global Internet of Things Summit (GIoTS)*. IEEE, 2020, pp. 1–6.

[25] A. Kazlouski, T. Marchioro, and E. Markatos, "I just wanted to track my steps! blocking unwanted traffic of fitbit devices," *Proceedings of the 12th International Conference on the Internet of Things*, ser. IoT '22. New York, NY, USA, Association for Computing Machinery, 2023, p. 96–103. [Online]: https://doi.org/10.1145/3567445.3567457

[26] Pi-hole, "Pi-hole Network-wide Ad Blocking," URL: https://pi-hole.net/, Jan. 2023, Accessed: 2023-01-13.