

Investigating the Latent Security and Privacy Risks in Consumer-oriented  
Software Systems

Kaushal Kafle

Biratnagar, Nepal

Bachelor's Degree in Computer Engineering, Tribhuvan University, 2015

A Dissertation presented to the Graduate Faculty of The College of William and  
Mary in Virginia in Candidacy for the Degree of  
Doctor of Philosophy

Department of Computer Science

The College of William and Mary in Virginia  
August 2024



## APPROVAL PAGE

This Dissertation is submitted in partial fulfillment of  
the requirements for the degree of

Doctor of Philosophy



---

Kaushal Kaffe

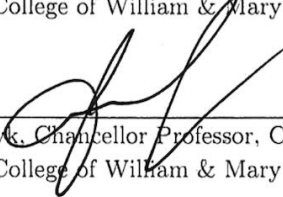
Reviewed by the Committee, August 2024



---

Committee Chair

Adwait Nadkarni, Class of 1953 Associate Professor, Computer Science  
College of William & Mary



---

Denys Poshyvanyk, Chancellor Professor, Computer Science  
College of William & Mary



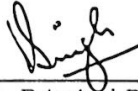
---

Dmitry Evtushkin, Assistant Professor, Computer Science  
College of William & Mary



---

Stephen Herwig, Assistant Professor, Computer Science  
College of William & Mary



---

Kapil Singh, Principal Research Scientist  
IBM Research

## ABSTRACT

Consumer-oriented software systems have become the foundation on which consumer data is collected and transported from the consumers to the data processors. They are complex, with various interconnected, heterogeneous components working together, making their security and privacy analysis challenging, and impact on the user uncertain. In this work, we first explore how security threats can arise in novel context in such systems by performing a security evaluation of data-store based smart home platforms and the overall security risks posed by the design of routines within such platforms. We analyze various components of the smart home such as the platform’s permission enforcement mechanism and the apps/services that connect to the smart home that are used during the creation and execution of routines. We find that 1) platform’s permission enforcement and access control model may be broken and allow for attacker’s to bypass user’s consent to perform privileged tasks, 2) around 20% of apps that connect to smart home platforms may have vulnerable SSL connections, and 3) lateral privilege escalation in smart home platforms is possible with the help of routines, wherein we demonstrate by compromising a smart home camera by escalating our privilege gained with a smart home switch app. Secondly, to develop a practical defense against the threats introduced by the routines, we leverage the unique opportunity provided by the smart home *i.e.*, validating incoming state change requests by comparing with the observations gathered by physical devices connected to the platform, for enhancing integrity in smart home platforms. Using this insight, we propose HomeEndorser, which is a practical framework to provide integrity guarantees to smart home platforms. To do so, HomeEndorser endorses (or rejects) requests by apps or services to modify Abstract Home Objects (AHOs) such as home or fire by enforcing integrity policies based on the current state of devices in the home. By protecting against malicious modifications of AHOs, HomeEndorser is able to prevent arbitrary privilege escalation attacks that were possible by exploiting routines. Finally, to understand how effectively stakeholders convey security and privacy risks to the users, we designed the Polityzer framework to systematically analyze the privacy postures of election campaign websites. Using Polityzer, we find a vast majority of election campaign websites lack a privacy disclosure, and even in cases where privacy policies were provided, they were often incomplete. We also found that campaigns may be inadvertently sharing data with other campaigns through common fundraising platforms, without disclosing such sharing.

## TABLE OF CONTENTS

Acknowledgments	vi
Dedications	vii
List of Tables	viii
List of Figures	x
Chapter 1. Introduction	1
1.1 Thesis Statement	3
1.2 Dissertation Organization	4
Chapter 2. Background and Related Work	5
2.1 Home Automation via Centralized Data Stores	5
2.1.1 General Characteristics	6
2.1.2 Nest Background	8
2.1.3 Hue Background	10
2.2 U.S. Federal Election Campaigns	10
2.3 Related Work	11
Chapter 3. Exploring the Security of Data Store-Based Home Automation	16
3.1 Overview of Analysis in this Chapter	20
3.2 Evaluating Permission Enforcement	21
3.2.1 Generating Permission Maps	21
3.2.2 Analyzing Permission Maps	23

3.2.3	Permission Enforcement Findings	$(\mathcal{F}_1 \rightarrow \mathcal{F}_3)$	24
3.3	Evaluating Smart Home Routines		26
3.3.1	Methodology for the Analysis of Routines		27
3.3.2	Smart Home Routine Findings	$(\mathcal{F}_4 \rightarrow \mathcal{F}_5)$	28
3.4	Security Analysis of Nest Apps		30
3.4.1	Application Permission Descriptions		30
3.4.1.1	Analysis Methodology		31
3.4.1.2	Nest App Findings	$(\mathcal{F}_6 \rightarrow \mathcal{F}_9)$	31
3.4.2	Application SSL Use		34
3.5	Lateral Privilege Escalation		35
3.6	Vulnerability Reporting Experience and Current Status		37
3.6.1	SSL Vulnerability in TP Link’s KASA		37
3.6.2	Vulnerable Nest routines, and misinformation in third-party Works with Nest apps		38
3.6.3	Permission Enforcement Vulnerabilities in Hue		40
3.6.3.1	Analyzing the Updated Hue API, from both local and remote apps		41
3.6.3.2	Key Results from Updated Analysis		41
3.7	Feasibility of Analyzing Evolving Smart Home Platforms		42
3.7.1	Platform-independent <i>essential properties</i>		43
3.7.2	Evaluation of 6 Additional Smart Home Platforms		45
3.7.2.1	Nest v1 and v2		46
3.7.2.2	SmartThings Classic and v2		47
3.7.2.3	HomeKit		48
3.7.2.4	Home Assistant		48
3.7.2.5	OpenHAB		49
3.7.3	The challenge for future security research		50

3.8	Lessons from the Study	50
3.9	Chapter Summary	52
Chapter 4.	HomeEndorser: Practical Integrity Validation in the Smart Home	54
4.1	Chapter Motivation	57
4.1.1	API Misuse as an OS Integrity Problem	59
4.1.2	The Need for Integrity Validation	60
4.2	HomeEndorser Design Goals	61
4.3	The HomeEndorser Framework	62
4.3.1	Policy Model	64
4.3.2	Secure and Practical Enforcement	66
4.3.3	Data-driven Policy Template Generation	68
4.4	HomeEndorser Implementation	71
4.5	HomeEndorser Evaluation	72
4.5.1	Feasibility of the Policy Model (HomeEndorser-RQ <sub>1</sub> )	75
4.5.2	Generalizability of the Policy Model (HomeEndorser-RQ <sub>2</sub> )	76
4.5.3	Case Study: Protecting the Security Camera from Privilege Escalation (HomeEndorser-RQ <sub>3</sub> )	76
4.5.4	Intentional AHO Changes and Normal Usage (HomeEndorser-RQ <sub>4</sub> )	78
4.5.5	Runtime Performance (HomeEndorser-RQ <sub>5</sub> )	81
4.5.6	Effort Required to Integrate and Configure HomeEndorser (HomeEndorser-RQ <sub>6</sub> )	82
4.6	Threats to Validity of this Chapter	84
4.7	Chapter Summary	86
Chapter 5.	Polityzer: Understanding the Privacy Practices of Political Campaigns	88

5.1	Motivation	91
5.1.1	Expectations of Governments and Regulators	92
5.1.2	User Expectations and Desire for Privacy	93
5.1.3	Why Should Researchers Care?	94
5.2	Ethical Considerations	95
5.2.1	P <sub>1</sub> : Focus on Privacy, and not Politics	96
5.2.2	P <sub>2</sub> : Limiting Harm to Candidates	96
5.2.3	P <sub>3</sub> : Limiting Harm to Campaign Resources	97
5.2.4	P <sub>4</sub> : Transparency	97
5.3	Overview	98
5.4	Campaign Collector	99
5.4.1	Website Collection Methodology	99
5.4.2	Results of Campaign Website Collection	101
5.5	Analysis of Data Collection (Polityzer-RQ <sub>1</sub> )	103
5.5.1	Methodology	103
5.5.2	Results and Findings	104
5.6	Analysis of Privacy Disclosure (Polityzer-RQ <sub>2</sub> )	107
5.6.1	Methodology	108
5.6.2	Results and Findings	109
5.7	Inter-campaign Sharing Analysis (Polityzer-RQ <sub>3</sub> )	113
5.7.1	Methodology	113
5.7.2	Results and Findings	115
5.8	Security Risk Analysis (Polityzer-RQ <sub>4</sub> )	116
5.9	Use of Privacy Policy Templates	120
5.10	Discussion	121
5.10.1	The Privacy Posture of Campaign Websites	122
5.10.2	Rationale for the Present Privacy Postures	123

5.10.3 Towards Privacy, Transparency, and Accountability in Political Campaign Websites	124
5.11 Threats to Validity	125
5.12 Chapter Summary	127
Chapter 6. Future Work	128
Chapter 7. Conclusion	131

## ACKNOWLEDGMENTS

This work would not be possible without the unwavering support and guidance of many individuals. First and foremost, I am very thankful to my advisor, Dr. Adwait Nadkarni for his tireless mentorship and support during my doctoral studies. As his first PhD student, I was fortunate to be a founding member of the Secure Platforms Lab (SPL) under his leadership, and a witness to some of its biggest accomplishments and challenges. Throughout all of them, his work ethic set a great example on how to pursue research with passion, integrity, and foresight while also creating a lab environment that values not just the outcome but also the process of doing meaningful work. This dissertation would not have been possible without his guidance, and I look forward to carrying with me the lessons I learnt from him as I begin my own academic career. I would also like to thank my committee members, Kapil, Denys, Dmitry, and Stephen for their feedback and support throughout the completion of this work. Thanks also to the CS program administrator Vanessa Godwin, without whom I could not have completed my doctoral studies in time, and who is the most helpful, kind, and understanding person. Thank you also to the rest of the administrative team, Dale Hayes and Jacquelyn Johnson for your help and support!

I would also like to thank various members of the SPL group for their support and friendship. In particular, I would like to thank Sunil, Brian, Amit, Prianka, and Victor for their support in the projects that formulate this dissertation as well as their friendship and camaraderie inside and outside of the lab.

I am eternally thankful to my parents Devendra and Numa, and my sisters Kalpana and Kanchan for their unwavering support throughout my academic journey. To my parents, your perseverance, dedication and the sacrifices you have made for me and my sisters is a constant source of inspiration and motivation for me. To my grandma Tulasha Devi, thank you for your unconditional love all my life. Thanks also to my American family Brian and Maryfay. You have made my years as a doctoral student a joy, and I cherish the familial support and guidance that you have provided me all these years. Chauss!

Most importantly, I would like to thank my wife Kesina. This dissertation and my doctoral studies would not be possible without your love, companionship, advice, and encouragement through both the good times and the hardships. You are the source of my energy and I am thankful for your patience with me during my periods of doubt and insecurity. Words cannot truly capture the appreciation I feel for having you in my life.

I would like to dedicate this dissertation to my parents - Devendra and Numa - who have made endless sacrifices in their lives to get me to this position.

## LIST OF TABLES

3.1	Permission description violations discovered in Works with Nest apps	53
3.2	Feasibility of Analyses $\mathcal{A}_1 \rightarrow \mathcal{A}_5$ on various smart home platforms.	53
4.1	AHOs inferred from Endorsement Attributes	71
4.2	Real and Virtual Devices in Evaluation	74
4.3	Sample policies for endorsing home (“away” $\rightarrow$ “home”)	75
4.4	Performance overhead of HomeEndorser (HE) (in comparison with the unmodified HomeAssistant baseline)	81
4.5	The (minimal) cost of Integrating HomeEndorser with respect to the properties identified in Section 4.5.6	84
4.6	Realistic Scenarios used in evaluation of HomeEndorser	87
5.1	Dataset Overview	103
5.2	Data collected by campaign websites. <b>FEC</b> indicates data required by the FEC, <b>!-FEC</b> indicates data not required by the FEC, <b>FEC*</b> indicates the data not explicitly required by the FEC, but often a part of donation receipts shared with the FEC.	105
5.3	Top 10 most collected data types among campaign websites	106
5.4	Campaigns’ collection of political opinions with PII in the same page	107
5.5	Missing privacy policies in campaign websites	109
5.6	Total privacy policy extracted for analysis per dataset	110
5.7	Data collection without privacy disclosure.	111
5.8	Top 10 Most commonly undisclosed datatypes	112

5.9	Campaign websites that disclose sharing data with other campaigns	112
5.10	Use of Platform <sub>1</sub> or Platform <sub>2</sub> for fundraising.	115
5.11	No. of unsafe campaign websites across datasets	117
5.12	non-US countries where websites are hosted	118
5.13	No. of non-TLS websites	118
5.14	No. of campaign websites with trackers	119
5.15	Extra data mentioned in the privacy policy but not collected in the website.	120

## LIST OF FIGURES

2.1	The general architecture of platforms that leverage centralized data stores. Note that $H$ is the universe of all home state variables, and $V_{\text{device}_i}$ is the universe of all state variables specific to $\text{device}_i$ .	6
2.2	A simplified view of the centralized data store in Nest.	8
3.1	The <i>Keen Home</i> app asks the user to modify the thermostat’s mode, but in reality, this action leads to the <i>entire</i> smart home being set to “home” mode, which affects a number of other devices.	29
3.2	An example from the Nest documentation on OAuth authorization [136] that displays a permission description violation (specifically, VC1) for the <i>Away r/w</i> and <i>Camera + images r/w</i> permissions. The developer’s permission description indicates that the FTL Lights only need to read data store variables, in both cases.	32
4.1	An attack on the security camera through the manipulation of two shared state objects by adversary-controlled integrations.	58
4.2	A conceptual overview of physical home endorsement.	62
4.3	Design components of HomeEndorser’s enforcement	66
4.4	HomeEndorser’s policy specification methodology.	68
4.5	Specification of device-attributes in the SmartThings device handler preamble.	69
4.6	Layout of the physical device placement	75
5.1	Overview of the Polityzer framework	98

# Chapter 1

## Introduction

In the last decade, there has been a rapid rise in consumer-oriented software systems that are focused on directly interacting with consumers to collect and process their data in order to provide different services. Such systems can be very diverse in nature, including systems like the Internet-of-Things (IoT) [100], mobile apps [117] and even online election campaigns [122]. Consumer-oriented software systems have grown in prominence over the last decade and have become the cornerstone through which consumer data is collected and transported from the consumers to the stakeholders (e.g., usage behavior collected through mobile apps) [145].

In addition to the diversity in terms of functionality and architecture, such software systems are also composed of different subcomponents including the use of cloud APIs, storage, third-party apps and integrations, third-party tools and other dependencies. For instance, consider the smart home which is a popular subclass of the IoT domain. It consists of not just many kinds of IoT devices, but also of several smart home platforms, thousands of apps that integrate into the platforms and control the IoT devices and routines (or automations) created by the users. Due to this layer of interdependencies, it is very

---

**IEEE Copyright Note:** In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of the College of William and Mary's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

challenging for the consumers to assess the risks in security and privacy when using these software systems.

This dynamic nature of consumer-oriented software systems creates unique challenges in order to analyze the inherent security and privacy risks. For example, security threats that are well-known in traditional operating systems can manifest in these systems in novel ways and context. Similarly, any privacy analysis of such systems needs to consider the conflicts that could potentially arise due to the different privacy requirements of different components operating within the same system. For instance, online election campaigns and their fundraising platforms may have entirely different policies in dealing with the consumer data they collect.

In order to truly assess the security and privacy implications of using these systems, we need to first address the following inherent challenges:

1. *Diverse systems built on inter-connected components ( $\mathcal{C}_1$ ):* With several heterogeneous, interconnected components working together, the systems can also be increasingly complex to understand given that each component may have its own functionalities, design and interactions with other components. In order to gain a practical understanding of the security and privacy challenges, we need to analyze such systems holistically. In particular, it is critical to understand how security threats can arise due to the interplay between the components.
2. *Developing practical solutions ( $\mathcal{C}_2$ ):* Any solution or framework designed to identify or prevent attacks in such software systems needs to account for all the inter-connected subcomponents to be practical. It is also vital to leverage the abstractions provided by the systems themselves so that the solution is adaptable.
3. *Identifying the privacy requirements ( $\mathcal{C}_3$ ):* Given their diversity, the privacy requirements that underpin consumer-oriented software can also vary a lot from one stakeholder to the next. For example, the IoT vendors that sell and operate IoT devices fall under the for-profit domain, and are thus governed by data privacy regulations such as the GDPR [73] and CPRA [4]. However, the online election campaigns in the U.S. fall under the non-

profit domain, and are thus not governed by any data privacy regulations. Hence, the privacy analysis of such systems has to first consider the privacy requirements they are operating under.

## 1.1 Thesis Statement

The inter-dependent components in consumer-oriented software systems pose significant challenges in analysis as we outline above. For any evaluation of such systems to truly reflect the real-world risks posed to consumers who use them, several factors need to be considered.

For instance, consider the case of smart homes which are increasing in popularity among consumers every year [198]. A typical smart home system that is deployed by a consumer in their home consists not just of smart devices from multiple vendors, but also operates under a smart home platform (that we detail in Chapter 2), and can be managed by multiple mobile apps provided by different stakeholders (*e.g.*, platform app, device vendor app, or other third-party apps from the marketplace). Thus, tackling  $\mathcal{C}_1 \rightarrow \mathcal{C}_3$  is not trivial and requires a careful assessment of the system architecture, domain-specific considerations (*e.g.*, role of devices and automations), and the role that key stakeholders operating within the ecosystem play in minimizing the security and privacy risks.

Therefore, the thesis statement of this work is the following:

A practical security and privacy evaluation of complex, consumer-oriented, software systems, must consider the risks arising due to their inter-dependent components.

In proving this thesis, we seek answers to the following research questions in this dissertation.

- **RQ<sub>1</sub>: How do threats arise due to the inter-dependent components?** The emphasis of the analysis presented in this dissertation will be on identifying security and privacy threats that can manifest due to the inter-play among system components.

- **RQ<sub>2</sub>**: **How do we develop practical defenses against these threats?** To tackle the threats, the solutions need to be grounded on practical design decisions, with particular focus on making use of domain-specific abstractions.
- **RQ<sub>3</sub>**: **How effectively do stakeholders convey the risks to the users?** Finally, once we characterize the threats, the dissertation will also focus on the role of stakeholders in conveying the risks to the users, and analyzing the effectiveness of their methods.

## 1.2 Dissertation Organization

The rest of the dissertation is organized as follows. Chapter 2 provides background and discuss related work of this dissertation. In Chapter 3, we first explore the architecture of 2 popular smart home platforms, Google Nest and Philips Hue, with respect to how integrations and routines function within them. Next, we analyze and identify problems in their access control mechanisms as well as in their integrations. Finally, we carry out an end-to-end privilege escalation attack in the smart home by exploiting vulnerable integrations and routines (**RQ<sub>1</sub>**). In Chapter 4, we propose and implement the HomeEndorser framework, which is a framework that enables platforms to supplement their defenses *by providing an integrity guarantee mechanism using information drawn from devices in the home*. HomeEndorser endorses legitimate requests made by integrations by validating the requested changes with the device observations gathered at runtime while rejecting requests that are inconsistent with the device observations (**RQ<sub>2</sub>**). Finally, in chapter 5, we develop the framework Polityzer, using which we perform a systematic, large-scale analysis of the privacy posture of U.S. election campaign websites. In particular, we identify the types of data that campaign websites collect and identify several findings regarding the systemic problems associated with their privacy disclosures (**RQ<sub>3</sub>**).

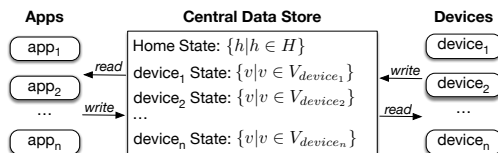
## Chapter 2

# Background and Related Work

In Section 2.1 of this chapter, we first describe the general architecture and characteristics of the smart home platforms that we study in Chapters 3 and 4. We also give a brief background of 2 smart home platforms in particular, Google Nest and Philips Hue, whose security analysis is discussed in detail in Chapter 3. Similarly, in Section 2.2, we describe the structure of the federal election campaigns in the United States that are the subject of our privacy analysis in Chapter 5. Finally, in Section 2.3, we discuss the related works of this dissertation.

### 2.1 Home Automation via Centralized Data Stores

This section describes the general characteristics of data store-based platforms, *i.e.*, smart home platforms that use a *centralized data store* to facilitate routines. We provide the background on two such platforms, namely (1) Google’s “Works with Nest” [143] platform (henceforth called “Nest”) and (2) the Philips Hue lighting system [159] (henceforth called “Hue”), which serve as the targets of our security evaluation. The Android apps for both of the systems have over a million downloads on Google Play [82, 83], indicating significant adoption, and far-reaching impact of our analysis.



**Figure 2.1:** The general architecture of platforms that leverage centralized data stores. Note that  $H$  is the universe of all home state variables, and  $V_{device_i}$  is the universe of all state variables specific to  $device_i$ .

### 2.1.1 General Characteristics

Figure 2.1 shows the general architecture of DSB platforms, consisting of 3 main components: *apps*, *devices*, and the *centralized data store*, which generally communicate over the Internet. Additionally, a physical hub that facilitates local communication via protocols such as Zigbee or Z-wave may be present (*e.g.*, the Hue Bridge). The apps may either be Web services hosted on the cloud, or mobile apps communicating via Web services. At this juncture, we generalize apps as third-party software interacting with the data store, and provide the platform-specific descriptions later.

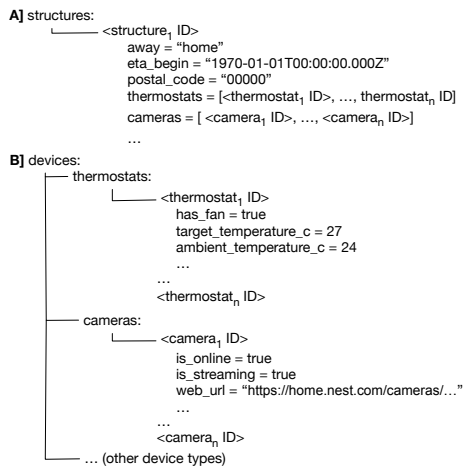
The centralized data store facilitates communication among apps and devices via state variables. The data store exposes two types of state variables: (1) *Home* state variables that reflect the general state of the entire smart home (*e.g.*, if the user is at *home/away*, the *devices attached* to the home, the *postal code*), and (2) *Device-specific* state variables that reflect the attributes specific to particular devices (*e.g.*, if the Camera is *streaming*, the *target temperature* of the thermostat).

Apps and devices communicate by reading from or writing to the state variables in the data store. This model allows expressive communication, from simple state updates to indirect trigger-action routines. Consider this simple state update: the user may change the temperature of the thermostat from an app, which in turn *writes* the change to the *target temperature* variable in the data store. The thermostat device receives an update from the data store (*i.e.*, *reads* the *target temperature* state variable), and changes its target temperature accordingly. Further, as stated previously, expressive routines may

also be implemented using the data store. For instance, the thermostat may change to its “economy” mode when the home’s state changes to *away*, *i.e.*, the thermostat’s app may detect that the user has left the smart home (*e.g.*, using Geofencing), and *write* to the home state variable *away*. The thermostat may then *read* this change, and switch to its economy mode.

A salient characteristic of DSB platforms is that they lean towards seamless home automation by automatically interacting with devices and executing complex routines via the data store. However, even among DSB platforms, there are some key differences, which motivates our targeted analysis of the Nest and Hue platforms and their apps, as we discuss below.

We observe that while both Nest and SmartThings execute routines, there is a key difference in how routines are managed in these platforms. SmartThings allows users to create and manage routines from the SmartThings app itself, thereby providing users with a general view of all the routines executing in the home [194]. In contrast, Nest routines are generally implemented as *decentralized* third-party integrations. Third-party products that facilitate routines provide the user with the ability to view and manage them. As a result, the Nest platform does not provide the user with a *centralized view* of the routines that are in place. Due to this lack of user control, Nest smart homes may face unique security risks and challenges, which motivates this security analysis. Similarly, we observe that the Philips Hue platform may be another interesting variant of DSB platforms. That is, Hue integrates *homogeneous* devices related to lighting such as lamps and bulbs, unlike Nest and SmartThings that integrate heterogeneous devices, and represents a drastically simpler (and hence unique) variant of home automation platforms that use centralized data stores. As a result, the analysis of Hue’s attack surface has potential to draw attention to other similar, homogeneous platforms, which is especially important considering the fragmentation in the smart home product ecosystem [47]. To our knowledge, this work is the first to analyze this relatively new class of smart home platforms, and specifically, Nest and Hue.



**Figure 2.2:** A simplified view of the centralized data store in Nest.

### 2.1.2 Nest Background

The *Works with Nest* platform integrates a heterogeneous set of devices, including devices from Nest (*e.g.*, Nest thermostat, Nest Cam, Nest Protect) as well as from other brands (*e.g.*, Wemo and Kasa switches, Google Home, MyQ Chamberlain garage door opener) [143]. This section describes the key characteristics of Nest, *i.e.*, its data store, its access control model, and routines.

**Data store composition:** Figure 2.2 shows a simplified, conceptual view of the centralized data store in Nest. Note that the figure shows a small fraction of the true data store, *i.e.*, only enough to facilitate understanding. Nest implements the data store as a JSON-format document divided into two main top-level sections: *structures* and *devices*. A *structure* represents an entire smart home environment such as a user’s home or office, and is defined by various state variables that are global across the smart home (*e.g.*, *Away* to indicate the presence or absence of the user in the structure and the *postal\_code* to indicate the home’s physical location). The devices are subdivided into device types (*e.g.*, thermostats, cameras, smoke detectors), and there can be many devices of a certain type, as shown in Figure 2.2. Each device stores its state in variables that are relevant to its type; *e.g.*, a thermostat has state variables for *humidity*, and *target\_temperature\_c*, whereas a

camera has the variables *is\_online* and *is\_streaming*. Aside from these type-specific variables, devices also have certain variables in common; *e.g.*, the alphanumeric *device ID*, the *structure ID* of the structure in which the device is installed, the device’s user-assigned *name*, and *battery\_health*.

**Access Control in Nest:** Nest treats third-party apps, Web services, and devices that want to integrate with a Nest-based smart home as “products”. Each Nest user account has a specific data store assigned to it and any product that requests access to the user’s data store needs to be first authorized by the user using OAuth 2.0. Nest defines read or read/write permissions for each of the variables in the data store. Some variables, *e.g.*, the list of all thermostats in the structure, are always *read-only*. A product that wants to register with Nest must first declare the permissions that it needs (*e.g.*, *thermostat read*, *thermostat read/write*) in the Nest developer console. When connecting a product to Nest, during the OAuth authorization phase, the user is shown the permissions requested by the product. Once the user grants the permissions, a revocable access token is generated specific to the product, the set of permissions requested, and the particular smart home to which the product is connected. This token is used for subsequent interactions with the data store.

**Accessing the Nest data store:** Devices and applications that are connected to a particular smart home (*i.e.*, the user’s Nest account) can update data store variables to which they have access, and also subscribe to the changes to the state of the data store. Nest uses the REST approach for these update communications, as well as for apps/devices to modify the data store. The REST endpoints can be accessed through HTTPS by any registered Nest products.

**Routines in Nest:** In Nest, the user cannot create or view routines in a centralized interface (*i.e.*, unlike SmartThings). Instead, apps may provide routines as opt-in features. For example, the Nest smoke alarm’s *smoke\_alarm\_state* variable has three possible values, “ok”, “warning”, and “emergency”. When this variable is changed to “warning”, other smart

home products (e.g., Somfy Protect [224]) can be configured to trigger and warn the user.

### 2.1.3 Hue Background

Unlike Nest, which is a platform for heterogeneous devices, Philips Hue deals exclusively with lighting devices such as lamps and bulbs. As a result, the centralized data store of Philips Hue supports much simpler routines. Hue implements its data store as a JSON document with sections related to (1) physical lighting devices, (2) semantic groups of these devices, and (3) global config variables (such as *whitelisted apps* and the *linkbutton*). To connect a third-party management app to a user's existing Hue system, the app identifies a Hue bridge connected to the local network, and requires the user to press a physical button on the bridge. Once this action is completed by the user, the app receives a *username* token that is stored in the *whitelisted* section of the Hue data store. Whitelisted apps can then read and modify data store variables as dictated by Hue's access control policy, which grants all authorized apps the same access regardless of their purported functionality.

## 2.2 U.S. Federal Election Campaigns

Federal elections in the U.S. can be divided into two categories (termed 'regular elections' from hereon): i) the *Presidential election* that elects the President, and ii) the *Congressional elections* that elect the members of the Congress. The presidential election occurs every four years while the congressional elections occur every two years. Further, the US Congress is divided into two branches – the House of Representatives (termed simply 'House' from hereon), which consist of 435 voting members and six delegates, and the Senate, which consist of 100 members. The members of the House and Senate serve terms of two and six years respectively. Hence, all 435 members (and 6 delegates) of the House get elected every two years while only a third members of the Senate get elected every two years, as was the case in the 2020 election where 32 Senate seats and 441 House seats (including six delegates) were contested.

Outside of the regular elections that occur every two years, *special elections* are held when there are vacancies for any Congressional seats before the regular elections. There were 11 special elections in 2020: eight for House seats and three for Senate seats[20]. Furthermore, all the regular elections to the House, the Senate and the Presidency are preceded by *primary* elections, where the political parties select a candidate that advances to the regular elections [125]. The analysis in Chapter 5 incorporates the election campaigns of all candidates (including those eventually elected) for both the Presidential and Congressional elections of 2020, including special elections and primaries.

Finally, all candidates who raise/spend over \$5000 must register their campaigns and file financial reports with the Federal Election Commission (FEC) [63]. The FEC discloses this information in a searchable database, including the state and district that a candidate is registered in as well as the amount of money raised and spent [61]. In this work, we treat the FEC database as the ground truth regarding the candidates participating in the election.

## 2.3 Related Work

We now discuss prior work that are related to different parts of this dissertation work.

**SSL analysis in apps:** Smart home platforms are an extension of the new modern OS paradigm, the security problems in smart home platforms are similar to prior modern OSes (*e.g.*, application over-privilege, incorrect enforcement). As a result, some of the same techniques may be applied in detecting such problems, *e.g.*, our work uses automated testing to derive permission maps and compares the maps to the platform documentation, in a manner similar to Felt et al.’s seminal evaluation of Android permission enforcement [64]. We also leverage lessons from prior work on SSL misuse [60, 152, 172, 196] to perform the SSL Analysis (Section 3.4.2) and the MiTM exploit (Section 3.5). The lack of transitivity in access control that we observe is similar to prior observations on Android [65, 35, 132, 131]; however, the implications are different in the smart home. The novelty of our work is

rooted in using lessons learned from prior research in modern OS and application security to identify problems in popular but under-evaluated platforms such as Nest and Hue, and moreover, in demonstrating the potential misuse of home automation *routines* for performing lateral privilege escalation.

**Security evaluation of Smart home platforms:** While prior work analyzes IoT apps to study the *potential* for adversarial misuse [51], our work is the first to demonstrate an end-to-end lateral privilege escalation attack involving routines (Section 3.5). This focus on adversarial misuse and a demonstrated end-to-end attack distinguishes our work from closely-related work in smart home security, such as the security evaluation of the SmartThings platform and its apps by Fernandez et al. [66], and systems such as IoTSAN [144] and the Soteria [33] that detect the side-effects of the concurrent execution of Samsung’s SmartApps. Aside from our 11 novel findings ( $\mathcal{F}_1 \rightarrow \mathcal{F}_{11}$ ), the value of our work is in its holistic evaluation of home automation security, *i.e.*, as we study the permission text artifacts, product review-based defenses, and the detrimental impact of platform evolution on the feasibility of analysis.

In a similar vein as this work, prior work by Surbatovich et al. [201] has analyzed the security and privacy risks associated with IFTTT recipes, which are trigger-action programs similar to routines. The key difference is that Surbatovich et al. examines the safety of individual recipes, while our work explores routines that may be safe on their own (*e.g.*, when *home*, turn off the Nest Cam), but which may be used as gadgets by attackers to attack a high-integrity device from a low-integrity device. Our holistic analysis is complementary to such per-routine analysis, as well as per-device security analysis performed in prior work, such as Sukhvir et al.’s attack on the communication and authentication protocols in Hue and Wemo [147], or Sivaraman et al.’s attack on the home’s firewall using a malicious device on the network [187]).

**Preventing API misuse in platforms:** Much prior work in smart home security has focused on reducing the misuse of sensitive APIs, through improvements to the platform per-

mission model (*e.g.*, risk-based permissions [171], functionality-based enforcement [119]), providing users with context of an API access (*e.g.*, ContexIoT [110]), enforcing least privilege using context from the app’s description (*e.g.*, SmartAuth [212]), and, when all else fails, forensic analysis by integrating fine-grained provenance tracking into apps (*e.g.*, ProvThings [217]). At first glance, the problem that we describe in Chapter 4 may also seem to be one of over-privilege or API misuse; however, the approach of removing privileges (or completely preventing third-parties from accessing Abstract Home Objects (AHOs) (Chapter 4) would be impractical because several third-party integrations may truly need access to AHOs, and denying them access may come at prohibitive usability costs. Instead, our solution in Chapter 4, HomeEndorser, presents a more direct solution to the lack of integrity for AHOs in the smart home. Hence, we assert that addressing this problem as API misuse by retrofitting existing defenses would be both ineffective and impractical.

**Centralized smart home state modifications:** An alternative to securing shared states, as opposed to our solution in Chapter 4 is centralizing them and only allowing trusted third parties to modify them, as Schuster et al. explore using “environmental situation oracles” [181] or ESOs. The ESO model aims to provide *privacy* (and not integrity), *i.e.*, to prevent several parties from persistently accessing the user’s private information (*e.g.*, location) to generate a shared state (*e.g.*, home), by only allowing one dedicated trusted app per shared state. The key difference between ESOs and our proposed solution is that our approach does not require a trusted party to take upon itself the responsibility of accurately generating the value for a shared state, which may introduce compatibility issues by requiring a separate trusted app for every AHO to be approved by various stakeholders such as users, platform vendors, and device vendors. Instead, our solution applies an endorsement policy that *sanity checks* a proposed AHO change based on consistency with expected device states.

Finally, a related problem of endorsing operations has also been explored in Android.

Android devices contain many sensors (e.g., camera and microphone) that apps may use if authorized. Android depends on users to authorize the sensors that apps may use, but a malicious app may access a sensor at will once authorized [183]. To address this problem, researchers have explored methods to endorse authorized sensor operations by comparing the context of the sensor operation request to authorized contexts. For instance, user-driven [176, 175] access control requires that applications use system-defined GUI gadgets associated with particular operations to endorse the sensor operation associated with a user input event unambiguously. The AWARE [157] and EnTrust [156] systems permit applications to choose their own GUI gadgets, but restrict applications to employ the same GUI configuration (e.g, layout) to endorse a sensor operation. For IoT systems, rather than GUI contexts, we find that state changes are associated with physical events, enabling them to be endorsed based on properties of those physical events.

**Election data and security analysis:** The Internet Society has performed data protection, privacy, and security analysis of presidential campaigns since 2016 [106]. The 2020 audit was limited to 20 presidential campaigns, whereas our work covers a 2060 senate, house, and presidential campaigns, and provides specific analysis of collected data-types relative to privacy policies. Further, our email study builds upon Podob et al. [163]’s work, which found evidence of sharing among campaigns and PACs. In addition to what Podob et al. study, we also explore contradictions between a campaign’s sharing practice and their privacy policy, and show the *types* of data being shared and the issues in disclosure practices.

Consolvo et al. [44] conducted interviews with campaign personnels to understand their security practices and perceptions about the use of digital assets, finding vulnerabilities and risk due to this use. To safeguard against such vulnerabilities, various organizations have outlined recommendations for campaigns [68, 150, 104, 67]. Our work instead focuses on the campaign websites and complements prior work by outlining the gaps in privacy and security posture of campaign websites. Finally, prior work [126] has also studied the de-

ceptive and clickbait-oriented tactics in campaign email contents to incentivize interaction. We instead seek to understand how the campaign websites collect the contact information (including email) of the users, and how its use and sharing is disclosed to the users.

**Targeted ads, profiling using social media:** Prior work has focused on the privacy impact of social-media based voter profiling and targeted advertising, especially following the Cambridge Analytica scandal [108][180][92][177]. Additionally, prior research has analyzed the impact of voter profiling through big data on election outcomes [75] and tried to predict election results based on user activities collected from twitter [195][72]. In contrast, our work focuses on campaign websites and user data that is collected directly through that medium by the campaigns, rather than targeted advertising through social media. Further, prior work has also raised concerns about voter privacy in the age of online political campaigning [115, 98]. We build upon such concerns and study the privacy impact of political campaigns through the overall privacy posture of their websites.

**Privacy policy analysis:** Prior work has analyzed various aspects of privacy policies such as their availability in mobile apps [27, 231, 87, 48], readability and comprehension [128, 28, 109], as well as analyzing their vagueness [18, 24, 88], consent and opt-out choices [179, 151], contradictions [17, 46, 228, 88], and regulatory compliance [27, 28]. Our work instead focuses solely on the privacy policy availability of campaign websites and performs a semi-automated analysis on the policy text to glean disclosed data objects, for which we leverage past works (*e.g.*, Polisis [88]) where appropriate.

## Chapter 3

# Exploring the Security of Data

## Store-Based Home Automation

Internet-connected, embedded computing objects known as *smart home products* have become extremely popular with consumers. The utility and practicality afforded by these devices has spurred tremendous market interest, with over 20 billion smart home products projected to be in use by 2020 [71], with an estimated market revenue of 43 billion USD by 2025 [198]. The diversity of these products is staggering, ranging from small physical devices with embedded computers such as smart locks and light bulbs, to full fledged appliances such as refrigerators and HVAC systems. In the modern computing landscape, smart home devices are unique as they provide an often imperceptible bridge between the digital and physical worlds by connecting physical objects to digital services via the Internet, allowing the user to conveniently automate their home. However, because many of these products are tied to the user's security or privacy (*e.g.*, door locks, cameras), it is important to understand the attack surface of such devices and platforms, in order to build practical defenses without sacrificing utility.

As the market for smart home devices has continued to mature, a new software paradigm has emerged to enable home automation via the interactions between smart home devices and the apps that control them. These interactions may be expressed as *routines*, which

are sequences of app and device actions that are executed upon one or more triggers, *i.e.*, an instance of the trigger-action paradigm in the smart home. Routines are the building block of home automation [204, 209, 58, 166], and hence, it is natural to leverage routines to characterize existing platforms.

If we categorize available platforms based on how routines are facilitated, we observe two broad categories: (1) API-based Smart Home Managers such as Yeti [226], Yonomi [227], IFTTT [101], and Stringify [200] that allow users to chain together a diverse set of devices using APIs exposed by device vendors, and (2) platforms such as Google’s Works with Nest [141], Samsung SmartThings [192], and Philips Hue [160] that leverage *centralized data stores* to monitor and maintain the states of IoT devices. We term these platforms as Data Store-Based (DSB) Smart Home Platforms. In DSB platforms, complex routines are executed via reads/writes to state variables in a central data store. We discuss the characteristics and architecture of DSB platforms in Chapter 2.

**Key observations:** This work is motivated by a key observation that while routines are supported via centralized data stores in all DSB platforms, there are differences in the manner in which routines are created, observed, and managed by the user. That is, SmartThings encourages users to take full control of creating and managing routines involving third-party apps and devices via the SmartThings app. On the other hand, in Nest, users do not have a centralized perspective of routines at all, and instead, manage routines using third-party apps/devices. This key difference may imply unique security challenges for Nest. Similarly, being a much simpler platform within this category of DSB platforms, Hue represents another unique and interesting instance of the DSB platform paradigm.

**Contributions in this Chapter:** In this chapter, to answer **RQ**<sub>1</sub>, we perform a systematic security analysis of some of the less studied, but widely popular, data store-based smart home platforms, *i.e.*, Nest and Hue. In particular, we evaluate (1) the access control enforcement in the platforms themselves, (2) the robustness of other non-system enforce-

ment (*e.g.*, product reviews in Nest), (3) the use, and more importantly, the *misuse* of routines via manipulation of the data store by low-integrity devices,<sup>1</sup> and finally, (4) the security of applications that integrate into these platforms.

To our knowledge, this work is the first to analyze this relatively new class of smart home platforms, in particular the Nest and Hue platforms, and to provide a holistic analysis of routines, their use, and potential for their misuse in DSB platforms. Moreover, this work is the first to analyze the accuracy of app-defined permission descriptions and prompts, which provide highly critical context to the user. Furthermore, we provide a detailed account of our vulnerability disclosure experience with four separate vendors, and discover that certain vulnerabilities may not always be fixable. Finally, we study the *ramifications of platform evolution* on the transparency and artifacts required for security analysis. In doing so we not only discover concrete problems in DSB platforms, but also use empirical analysis to reveal challenges for feasibly performing similar research studies in the near future. Our novel findings ( $\mathcal{F}_1 \rightarrow \mathcal{F}_{11}$ ) are summarized as follows:

- **Misuse of routines** – The permission model in Nest is fine-grained and enforced according to specifications ( $\mathcal{F}_1$ ), giving low-integrity third-party apps/devices (*e.g.*, a switch) little room for directly modifying the data store variables of high-integrity devices (*e.g.*, security cameras). However, routines supported by Nest allow low-integrity devices/apps to indirectly modify the state of high-integrity devices, by modifying the shared variables they rely on ( $\mathcal{F}_4$ ).
- **Lack of systematic defenses** – Nest does not employ transitive access control enforcement to prevent indirect modification of security-sensitive data store variables; instead, it relies on a product review of application artifacts before allowing API access. We discover that the product review process is insufficient and may not prevent malicious exploitation of routines; *i.e.*, the review mandates that apps prompt the user before modifying certain variables, but does not validate *what* the prompt contains, allowing

---

<sup>1</sup>In the context of our study, we define a device as high-integrity if it is advertised as security-critical by the device vendor (*e.g.*, Nest Cam) while those that are not security-critical are referred to as low-integrity (*e.g.*, Philips Hue lamp).

apps to deceive users into providing consent ( $\mathcal{F}_5$ ). Moreover, permission descriptions provided by apps during authorization are also often incorrect or misleading ( $\mathcal{F}_6, \mathcal{F}_9$ ), which demonstrates that malicious apps may easily find ways to gain more privilege than necessary ( $\mathcal{F}_7$ ), circumventing both users and the Nest product review ( $\mathcal{F}_8$ ).

- **Lateral privilege escalation** – We find that smart home apps, particularly those that connect to Nest and have permissions to access security-sensitive data store variables, have a significantly high rate of SSL vulnerabilities ( $\mathcal{F}_{10}$ ). We combine these SSL flaws with the findings discussed previously (specifically  $\mathcal{F}_4 \rightarrow \mathcal{F}_9$ ) and demonstrate a novel form of a *lateral* privilege escalation attack. That is, we compromise a low-integrity app that has access to the user’s Nest smart home (*e.g.*, a TP Link Kasa switch), use the compromised app to change the state of the data store to trigger a security-sensitive routine, and indirectly change the state of a high-integrity Nest device (*e.g.*, the Nest security camera). This attack can be used to deceive the Nest Cam into determining that the user is home when they are actually away, effectively disabling it.
- **Lack of bare minimum protections** – Unlike Nest, the access control enforcement of Hue is woefully inadequate. Third-party apps that have been added to a user’s Hue platform may arbitrarily add other apps without user consent, despite an existing policy that the user must consent to by physically pressing a button ( $\mathcal{F}_2$ ). Making matters worse, an app may *remove* other apps integrated with the platform by exploiting unprotected data store variables in Hue ( $\mathcal{F}_3$ ). These vulnerabilities may allow an app with seemingly useful functionality (*i.e.*, a Trojan [118]) to install malicious add-ons without the user’s knowledge, and replace the user’s integrated apps with malicious substitutes. While repeating our experiments on a version of Hue updated to address these issues, we discover that Hue’s mitigation is only partially successful ( $\mathcal{F}_{11}$ ).

The rest of the chapter is structured as follows: Section 2.1 describes the key attributes of DSB platforms. Section 3.1 provides an overview of our security evaluation, and Sections 3.2→3.4 describe our individual analyses. Section 3.5 demonstrates an end-to-end

attack, and Section 3.6 provides a detailed account of the vendors’ response to our findings. Section 3.7 describes our empirical study of the feasibility of our security analyses with 6 additional smart home platforms. Section 2.3 describes the related work. Section 3.8 concludes with lessons learned.

### 3.1 Overview of Analysis in this Chapter

This work analyzes the security of home automation platforms that rely on centralized data stores (*i.e.*, DSB platforms). Third-party apps are the security principals on such platforms, as they are assigned specific permissions to interact with the integrated devices. That is, as described in Section 2.1, DSB platforms consist of (1) *third-party apps* that interact with the smart home (*i.e.*, centralized data store and devices) by acquiring (2) *platform permissions*, and execute a complex set of such interactions as (3) *trigger-action routines*. Our analysis methodology takes these three aspects into consideration, starting with platform permissions, as follows:

**A. Analysis of Platform Permissions (Section 3.2):** We analyze the *enforcement* of platform permissions/access control to discover inconsistencies by automatically building permission maps.

**B. Analysis of Routines (Section 3.3):** While analyzing permission enforcement shows us what individual devices can accomplish with a certain set of permissions, we perform an experimental analysis with real devices to identify the interdependencies among devices and apps through the shared data model, and the ramifications of such interdependencies on the user’s security. Additionally, Nest does not enforce transitive access control to prevent dangerous side-effects of routines, but instead employs a product review process as a defense mechanism. We analyze the effectiveness of this review process using the permission prompts used by existing apps as evidence.

**C. Analysis of Third-party Apps (Section 3.4):** We analyze the permission descrip-

tions presented by mobile apps compatible with Nest to identify over-privileged apps, or apps whose permission descriptions are inconsistent with the permission requested. We then analyze the apps for signs of SSL misuse, which we will further leverage to indirectly exploit security critical devices.

We combine the findings from these three analyses to demonstrate an instance of a *lateral privilege escalation* attack in a smart home (**Section 3.5**). That is, we demonstrate how an attacker can compromise a low-integrity device/app integrated into a smart home (e.g., a light bulb), and use routines to perform protected operations on a high-integrity product (e.g., a security camera).

## 3.2 Evaluating Permission Enforcement

The centralized data store described in Section 2.1 may contain variables whose secrecy or integrity is crucial; e.g., unprotected write access to the *web\_url* field of the camera may allow a malicious app to launch a phishing attack, by replacing the URL in the field with an attacker-controlled one. To understand if appropriate barriers are in place to protect such sensitive variables, we perform an analysis of the permission enforcement in Nest and Hue.

Our approach is to generate and analyze the *permission map* for each platform, i.e., the variables that can be accessed with each permission, and inversely, the permissions needed to access each variable of the data store. Note that while this information should ideally be available in the platform documentation, prior analysis of similar systems has demonstrated that the documentation may not always be complete or correct in this regard [64, 66].

### 3.2.1 Generating Permission Maps

We generate the permission map using automated testing as in prior work on Android [64]. We use two separate approaches for Nest and Hue, owing to their disparate access control models.

**Approach for Nest:** We first created a simulated home environment using the Nest Home Simulator [142], and linked our Nest user account to this simulated smart home. We then created our test Android app, and connected our test app to the simulated home (*i.e.*, our Nest user account) as described in Section 2.1.2. Note that the simulated smart home is virtually identical to an end-user’s setup, such that real devices may be added to it. Using the simulator allows us to investigate the data store information of Nest devices (*e.g.*, the Smoke/CO detector) that we may not have installed.

In order to generate a complete view of the data store, we granted our test app all of the 15 permissions in Nest (*e.g.*, *Away read/write*, *Thermostat read*), and read all accompanying information. To build the permission map for Nest’s 15 permissions, we created 15 apps, such that each app requested a single unique permission, and registered these apps to our developer account in the Nest developer console. Note that we do not test the effect of permission combinations, as our goal is to test the enforcement of individual permissions, and Nest’s simple authorization logic simply provides an app with a union of the privileges of the individual permissions.

We then connected each of the 15 apps to our Nest user account using the procedure described in Section 2.1.2. We programmed each app to attempt to read and write each variable of the data store (*i.e.*, the previously derived *complete view*). We recorded the outcome of each access, *i.e.*, if it was successful, or an access control denial. In the cases where we experienced non-security errors writing to data store variables (*e.g.*, writing data with an incorrect type), we revised our apps and repeated the test. The outcome of this process was a permission map, *i.e.*, the mapping of each permission to the data store variables that it can read and/or write.

**Approach for Hue:** We followed the procedure for Hue described in Section 2.1.3 to get a unique token that registers our single test app with the data store of our Hue bridge. In Hue, all the variables of the data store are “readable” (*i.e.*, we verified that all the variables described in the developer documentation [160] can be read by third-party apps).

Therefore, to build the permission map, we first extracted the contents of the entire data store. Then, for each subsection within the data store, our app made repeated write requests, *i.e.*, PUT calls with the payload consisting of a dummy value based on the variable type (*i.e.*, String, Boolean and Integer). All the variables that were successfully written to using this method were assigned as “writable” variables. Similarly, our app made repeated DELETE calls to the API and the variables that were successfully deleted were assigned as “writable” variables. This generated permission map applies to all third-party apps connected to Hue, since the platform provides equal privilege to all third-party apps.

### 3.2.2 Analyzing Permission Maps

The objective behind obtaining the permission map is to understand the potential for application overprivilege, by analyzing the granularity as well as the correctness of the enforcement. We analyze the permission map to identify instances of (1) *coarse-grained permissions*, *i.e.*, permissions that give the third-party app access to a set of security-sensitive resources that must ideally be protected under separate permissions, and (2) *incorrect enforcement*, *i.e.*, when an app has access to more resources (*i.e.*, state variables) than it should have given its permission set, as per the documentation; *e.g.*, apps on SmartThings may lock/unlock the door lock without the explicit permission required to do so [66].

To perform this analysis, we first identified data store variables that may be security or privacy-sensitive. This identification was performed using an open-coding methodology by one author, and separately verified by another author, for each platform. We then performed further analysis by separately considering each such variable, and the permission(s) that allow access to it. A major consideration in our analysis is the security impact of an adversary being allowed read or read/write access to a particular resource. Moreover, our evaluation of the impact of the access control enforcement was contextualized to the platform under inspection. That is, when evaluating Nest, we took into consideration the semantic meaning and purpose of certain permissions in terms of the data store

variables, as described in the documentation (*e.g.*, that the *Away read/write* permission should be required to write to the *away* variable [135]). For Hue, we only considered the security-impact of an adversary accessing data store variables. Our rationale is that the Hue platform defines the same static policy (*i.e.*, same permissions) for all third-party apps, and hence, its permission map can be simply said to consist of just one permission that provides access to a fixed set of data store variables. As a result, we judge application over-privilege in Hue by considering the impact of an adversarial third-party app reading from or writing to each of the security-sensitive variables identified in Hue’s permission map.

The creation of the permission maps for both Nest and Hue requires the application of well-studied automated testing techniques, and as such, can be replicated for similar platforms, with minor changes to input data (*e.g.*, the permissions to test for).

### 3.2.3 Permission Enforcement Findings $(\mathcal{F}_1 \rightarrow \mathcal{F}_3)$

**Finding 1: The permission enforcement in Nest is fine-grained and correctly enforced, *i.e.*, as per the specification  $(\mathcal{F}_1)$ .** We observe that the Nest permission map is significantly more fine-grained, and permissions are correctly enforced, relative to the observations of prior research in similar platforms (*e.g.*, the analysis of SmartThings [66]). Some highly sensitive variables are always read-only (*e.g.*, the *web\_url* where the camera feed is posted), and there are separate read and read/write permissions to access sensitive variables. Variables that control the state of the entire smart home are protected by dedicated permissions that control write privilege; *e.g.*, the *away* variable can only be written to using the *Away read/write* permission, the *ETA* variable has separate permissions for apps to read and write to it (*i.e.*, *ETA read* and *ETA write*), and the Nest Cam can only be turned on/off via the *is\_streaming* variable, using the *Camera + Images read/write* permission that controls write access to it. Moreover, since many apps need to respond to the *away* variable (*i.e.*, react when the user is home/away), device-specific read permissions (*e.g.*, *Thermostat read*, *Smoke + CO read*) also allow apps to *read* the *away* variable,

eliminating the need for apps to ask for higher-privileged *Away read* permission. The separate read and read/write permissions are correctly enforced, *i.e., our generated permission map provides the same access as is defined in the Nest permission documentation [135]*. This is in contrast with findings of similar analyses of permission models in the past (*e.g.*, the Android permission model [64], SmartThings [66]), and demonstrates that the Nest platform has incorporated lessons from prior work in permission enforcement.

**Finding 2: In Hue, the access control policy allows apps to bypass the user’s explicit consent ( $\mathcal{F}_2$ ).** We discovered two data store variables that were not write-protected, and which have a significant part to play in controlling access to the data store and the user’s smart home. First, any third-party app can write to the *linkbutton* flag. Recall from Section 2.1.3 that the user has to press the physical button on the Hue bridge device to authorize an app’s addition to the bridge. The physical button press changes the *linkbutton* value to “true”, and allows the app to be added to the *whitelist* of allowed third-party apps. However, we discovered that once installed, an app can toggle the *linkbutton* variable at will, *enabling third-party apps to add other third-party apps to the smart home without the user’s consent*. This exploitable access control vulnerability can allow an app with seemingly useful functionality to install malicious add-ons by bypassing the user altogether. In our tests, we verified this attack with apps that were connected to the local network. This condition is feasible as a malicious app that needs to be added without the user’s consent may not even have to pretend to work with Hue; all it needs is to be connected to the local network (*i.e.*, a game on the mobile device from one of the people present in the smart home). Note that it is also possible to remotely perform this attack, which we discuss in Section 3.6 ( $\mathcal{F}_{11}$ ).

**Finding 3. In Hue, third-party apps can directly modify the list of added apps, adding and revoking access without user consent ( $\mathcal{F}_3$ ).** Hue stores the authorization tokens of apps connected to the particular smart home in a *whitelist* on the Hue Bridge device. While analyzing the permission map, we discovered that not only

could our third-party test app read from this list, it could also directly delete tokens from it. We experimentally confirmed this finding again, by removing *Alexa* and *Google Home* from the smart home, without the user’s consent. An adversary could easily combine this vulnerability with  $(\mathcal{F}_2)$ , to remove legitimate apps added by the user, add adversary-controlled apps (*i.e.*, by keeping the *linkbutton* “true”), all without the user’s consent. More importantly, users do not get alerts when such changes are made (*i.e.*, since it is assumed that the enforcement will correctly acquire user consent). Hence, unless the user actually checks the list of integrated apps using the Hue Web app, the user would not notice these changes.

While the Nest permission model is robust in its mapping of data store variables and permissions required to access them, Section 3.3 demonstrates how fields disallowed by permissions may be indirectly modified via strategic misuse of routines, and describes Nest’s product review guidelines to prevent the same [137]. Section 3.4 describes how badly written and overprivileged apps escape these review guidelines, and motivate a technical solution.

### 3.3 Evaluating Smart Home Routines

Prior work has demonstrated that in platforms that favor application interoperability but lack transitive access control enforcement, problems such as confused deputy and application collusion may persist [65, 35, 132, 131]. Smart homes that facilitate routines are no different, but the exploitability and impact of routines on smart homes is unknown, which motivates this aspect of our study.

Recall that routines are trigger-action programs that are either triggered by a change in some variable of the data store, or whose action modifies certain variables of the data store. While both Nest and Hue share this characteristic, routines in Hue are fairly limited in scope, and their exploitation is bound to only affect the lighting of the smart home. As a result, the security analysis in this section is focused on the heterogeneous Nest platform that facilitates more diverse routines.

### 3.3.1 Methodology for the Analysis of Routines

While using the simulator as described in Section 3.2 allows us to understand what routines are *possible* on the platform, *i.e.*, what variables might be manipulated, and what Nest devices (e.g., the Nest Cam, Nest Thermostat) are affected as a result, we performed additional experiments with real apps and devices to study existing routines in the wild. For this experiment, we extended the smart home setup previously discussed in Section 3.2 with real devices.

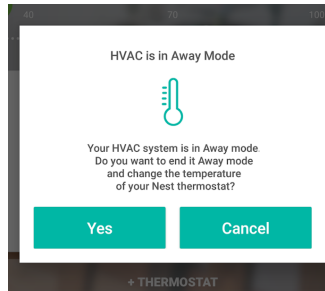
We started by collecting a list of devices that integrate with Nest from the *Works with Nest* website [143]. Using this initial list and information from the website, we purchased a set of 7 devices that possessed a set of characteristics relevant to this study, *i.e.*, devices that (1) take part in routines (*i.e.*, as advertised on the website), (2) are important for the user’s security or privacy, and (3) are widely-known/popular with a large user base (*i.e.*, determined by the number of installs of the mobile client on Google Play). We obtained a final list of devices (7 real and 2 simulated) to our Nest smart home, namely, the Nest Cam (*i.e.*, a security camera), Hue light bulb, Belkin Wemo switch, the MyQ Chamberlain garage door opener, TP Link Kasa Smart Plug, Google Home, Alexa, Nest Thermostat (simulated), and the Nest Protect Smoke & CO Alarm (simulated). Some security-sensitive devices did not participate in routines at the time of the study, and hence were excluded from our final device list.

We connected these devices to our Nest smart home using the Android apps provided by device vendors, and connected a small set of smart home managers (*e.g.*, Yeti [226] and Yonomi [227]) to our Nest smart home as well. For each device, we set up and executed every routine described on the Works with Nest as well as on the device vendor’s website, and observed the effects on the rest of the smart home (especially, security-sensitive devices). Also, we manipulated data store variables from our test app, and observed the effects on previously configured routines and devices.

### 3.3.2 Smart Home Routine Findings $(\mathcal{F}_4 \rightarrow \mathcal{F}_5)$

**Finding 4. Third-party apps that do not have the permission to turn on/off the Nest Cam directly, can do so by modifying the *away* variable ( $\mathcal{F}_4$ ).** The Nest Cam is a home monitoring device, and important for the users’ security. The *is\_streaming* variable of the Nest Cam controls whether the camera is on (*i.e.*, streaming) or off, and can only be written to by an app with the permission *Camera r/w*. The Nest Cam provides a routine as a feature, which allows the camera to be automatically switched on when the user leaves the home (*i.e.*, when the *away* variable of the smart home is set to “away”), and switched off when the user returns (*i.e.*, when *away* is set to “home”). Leveraging this routine, third-party apps such as the Belkin Wemo switch can manipulate the *away* field, and indirectly affect the Nest Cam, without having explicit permission to do so. We tested this ability with our test app (see Section 3.2) as well, which could indirectly switch the camera *on* and *off* at will. This problem has serious consequences; *e.g.*, a malicious test app with the *away r/w* permission may set the variable to “home” when the user is away to prevent the camera from recording a burglary. The key problem here is that a *low-integrity device/app can trigger a change in a high-integrity device indirectly, i.e.*, by modifying a variable it relies on, which is an instance of the well-known information flow *integrity* problem. Moreover, this is not the only instance of a high-integrity routine that relies on *away*; *e.g.*, the Nest x Yale Lock can lock automatically when the home changes to *away* mode [223].

Nest has a basic defense to prevent such issues: application design policies that apply to apps with more than 50 users [137]. App developers are required to submit their app for a product review to the Nest team once the app reaches 50 users, and a violation of the rather strict and detailed review guidelines can result in the app being rejected from using the Nest API. One of the review policies (*i.e.*, specifically policy 5.8) states that “*Products that modify Home/Away state automatically without user confirmation or direct user action will be rejected.*” [137]. Nest users may be vulnerable in spite of this defense,



**Figure 3.1:** The *Keen Home* app asks the user to modify the thermostat’s mode, but in reality, this action leads to the *entire* smart home being set to “home” mode, which affects a number of other devices.

for two reasons. First, as attacking a smart home is an attack on a user’s personal space, it is feasible to assume that most attacks that exploit routines will be targeted (*e.g.*, to perform burglaries). Assuming that the adversary can use social engineering to get the user to connect a malicious app to their Nest setup, *a targeted attack on a specific user will succeed in spite of the policy*, as the app would be developed solely for the targeted user and hence will have  $<50$  users, and be exempt from the Nest product review. Second, it is unclear how apps are checked against this policy; our next finding demonstrates a significant omission in Nest’s review.

**Finding 5.** Nest’s product review policies dictate that the apps must prompt users before modifying *away*, but there is no official constraint on *what the prompt may display* ( $\mathcal{F}_5$ ). Consider an example in Figure 3.1, which shows one such prompt by the *Keen Home* app [222] *when the user tries to change the temperature of the thermostat*. That is, when the user tries to change the temperature of the thermostat while the *away* variable is set to “away”, the app requires us to change it to “home” before the thermostat temperature can be changed. This condition is entirely unnecessary to change the temperature. More importantly, it presents the prompt to the user in a way that states that the home/away modes are specific to the HVAC alone. This is in contrast to the actual functionality of these modes, in which a change to the *away* variable affects the *entire* smart home; *i.e.*, we confirmed that the Nest Cam gets turned off as well once

we agree to the prompt. It is important to note that the *Keen Home* app has successfully passed the Nest product review, and has over 1000 downloads on Google Play [81]. This case demonstrates that the Nest product review does not consider the contents of the prompt, and a malicious app may easily misinform the user and make them trigger the *away* variable to the app’s advantage. Finally, in Section 3.4.1 we demonstrate that the problem of misinforming the user is not just limited to *runtime prompts* described here, but extends to application-defined *install-time permission descriptions* ( $\mathcal{F}_6 \rightarrow \mathcal{F}_9$ ).

### 3.4 Security Analysis of Nest Apps

In this Section, we investigate the third-party apps integrated with Nest. Unlike prior work [66], we not only report the permissions requested by apps, but also analyze the permission descriptions displayed to the user at install-time. Additionally, we analyze the rate of SSL misuse by both general smart home management apps as well as apps integrated with Nest. For this section, we do not consider the Hue platform as it has a limited ecosystem of apps as compared to Nest. We derived two datasets to perform the analyses that we describe in this section, the  $\text{Apps}_{\text{general}}$  dataset, which contains 650 smart home management apps extracted from Google Play, and the  $\text{Apps}_{\text{nest}}$  dataset, which includes 39 apps that integrate into the Nest platform (out of the total 130 Works with Nest apps, *i.e.*, 30%). Thus, while we cannot say that our analysis and findings ( $\mathcal{F}_6 \rightarrow \mathcal{F}_9$ ) generalize to all the apps compatible with Nest, they certainly apply to a significant minority (*i.e.*, 30%).

#### 3.4.1 Application Permission Descriptions

In the Nest platform, developers provide permission descriptions that explain how an app uses a permission while registering their apps in the Nest developer console. These developer-provided descriptions are the *only* direct source of information available to the user to understand why an app requires a particular permission, *i.e.*, Nest itself only provides a short and generic permission “title” phrase that is displayed to the user along

with the developer-defined description (*e.g.*, for *Thermostat read*, the Nest phrase is “See the temperature and settings on your thermostat(s)”). Owing to their significant role in the user’s understanding of the permission requirements, we analyze the *correctness* of such developer-defined descriptions relative to the permissions requested.

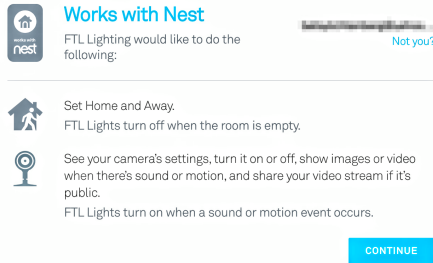
### 3.4.1.1 Analysis Methodology

As described in Section 2.1, upon registering permissions at the developer console, developers are granted an OAuth URL that they can direct the user to for obtaining an access token. As a result, permissions are not encoded in the client mobile app or Web app (*i.e.*, unlike Android), which makes the task of extracting permissions difficult. However, we observe that the permissions that an app asks for are *always* displayed to the user for approval (*i.e.*, when first connecting an app to their Nest smart home using OAuth). We leverage this observation to obtain permissions dynamically, *i.e.*, by executing apps to the point of integrating them with our Nest smart home, and recording the permission prompt displayed for the user’s approval.

### 3.4.1.2 Nest App Findings ( $\mathcal{F}_6 \rightarrow \mathcal{F}_9$ )

The two permissions that dominate the permission count are *Away read/write* and *Thermostat read/write*, requested by 20 and 24 apps respectively, from the  $\text{Apps}_{\text{nest}}$  dataset. Our specific findings from this analysis are as follows:

**Finding 6. A significant number of apps provide incorrect permission descriptions, which may misinform users ( $\mathcal{F}_6$ ).** As shown in Table 3.1, we found a total of 15 permission description violations in 13/39 apps from the  $\text{Apps}_{\text{nest}}$  dataset. We classify these incorrect descriptions into four violation categories (*i.e.*, VC1  $\rightarrow$  VC4), based on the specific manner in which they misinform the user, such as requesting more privileges than required for the described need (*e.g.*, read/write permissions when only reading is required), or misrepresenting the effect of the use of the permission (*e.g.*, stating *Away* as affecting only the thermostat). That is, *over 33.33% of the apps we could integrate have violating permission descriptions.*



**Figure 3.2:** An example from the Nest documentation on OAuth authorization [136] that displays a permission description violation (specifically, VC1) for the *Away r/w* and *Camera + images r/w* permissions. The developer’s permission description indicates that the FTL Lights only need to read data store variables, in both cases.

**Finding 7. In most cases of violations, apps request read/write permissions instead of read ( $\mathcal{F}_7$ ).** In nine cases, apps request the more privileged *read/write* version of the permission, when they should have clearly requested the *read* version, as per their permission description (*i.e.*, VC1 in Table 3.1). For example, consider the “MyQ Chamberlain” app (Table 3.1, entry 3), which asks for the *thermostat read/write* permission, but whose description only suggests the need for the *thermostat read* permission, *i.e.*, “Allows Chamberlain to display your Nest Thermostat temperature in the MyQ app”. More importantly, a majority of the violations of this kind occur for the *Away read/write* and *Camera+Images read/write* permissions, which may have serious consequences if these overprivileged apps are compromised, *i.e.*, as *Away read/write* regulates control over indicating whether a user is at home or out of the house, and *Camera+Images read/write* may allow apps to turn off the Nest cam via the *is\_streaming* variable. These violations exist in spite of Nest guidelines that mention the following as a *Key Point*: “Choose ‘read’ permissions when your product needs to check status. Choose ‘read/write’ permissions to get status checks and to write data values.” [135]. Finally, we found that the *Nest documentation may itself have incorrect instructions*, *e.g.*, the Nest’s documentation on OAuth 2.0 authentication [136] shows an example permission prompt that incorrectly requests the *Away read/write* permission while only needing read access, *i.e.*, with the description “FTL Lights turn off when the room is empty”, as shown in the Figure 3.2.

**Finding 8. The Nest product review is insufficient when it comes to reviewing the correctness of permission descriptions and requests by apps ( $\mathcal{F}_8$ ).** The Nest product review suggests the following two rules, violating which may cause apps to be rejected: (1) “3.3. Products with names, descriptions, or permissions not relevant to the functionality of the product”, and (2) “3.5. Products that have permissions that don’t match the functionality offered by the products” [137]. Our findings demonstrate that the 16 violations discovered violate either one or both of these rules (*e.g.*, by requesting read/write permissions, when the app only requires read). The fact that the apps are still available suggests that the Nest product review may not be rigorously enforced, and as a result, may be insufficient in protecting the attacks discovered in Section 3.3.

**Finding 9. Apps often incorrectly describe the *Away* field as a local field of the Nest thermostat, which is misleading ( $\mathcal{F}_9$ ).** One example of this kind (VC2 in Table 3.1) is the *Keen Home* app described in Section 3.3 (Table 3.1. entry 12), which states that it needs *Away* read/write in order to “Allow Smart vent to read the state of your Thermostat and change the state from Away to Home”. As a result, *Keen Home* misrepresents the effect and significance of writing to the *Away* field, by making it seem like *Away* is a variable of the thermostat, instead of a field that affects numerous devices in the home. Gideon and Muzzley (entries 10 and 11 in Table 3.1) exhibit a similar anomaly. Our hypothesis is that such violations occur because Nest originally started as a smart thermostat that gradually evolved into a smart home platform. Finally, in addition to misleading descriptions classified as VC1 and VC2, we discovered apps whose permission descriptions did not relate to the permissions requested (VC4), and apps whose descriptions satisfied both VC1 and VC2 (VC3).

The accuracy of permission descriptions is important, as the user has no other source of information upon which to base their decision to trust an app. Nest recognizes this, and hence, makes permissions and descriptions a part of its product review. The discovery of inaccurate descriptions not only demonstrates that apps may be overprivileged, but also

that Nest’s design review process is incomplete, as it puts all its importance on getting the user’s consent via permission prompts (*e.g.*, in Findings 5→9), but not on what information is actually shown.

### 3.4.2 Application SSL Use

The previous section demonstrated that smart home apps may be overprivileged in spite of a dedicated product review. An adversary may be able to compromise the smart home by exploiting vulnerabilities in such overprivileged apps. Thus, we decided to empirically derive an estimate of how vulnerable smart home apps are in terms of their use of SSL APIs, an important attack surface.

We used two datasets for this experiment, *i.e.*, the `Appsgeneral` dataset consisting of 650 generic smart home (Android) apps crawled from Google Play, and an extended version of the `Appsnest` dataset, *i.e.*, the `AppsnestExt` dataset, which consists of 111 Android apps built for Works with Nest devices (*i.e.*, including the ones for which we do not possess devices). We analyzed each app from both the datasets using MalloDroid [60], to discover common SSL flaws.

**Finding 10. A significant percentage of general smart home management apps, as well as apps that connect to Nest have serious SSL vulnerabilities ( $\mathcal{F}_{10}$ ).** 20.61% (*i.e.*, 134/650) of the smart home apps from the `Appsgeneral` dataset, and 19.82% (*i.e.*, 22/111) apps from the `AppsnestExt` dataset, have at least one SSL violation as flagged by MalloDroid. Specifically, in the `AppsnestExt` dataset, the most common cause of an SSL vulnerability is a broken *TrustManager* that accepts *all certificates* (*i.e.*, 20 violations), followed by a broken *HostNameVerifier* that does not verify the hostname of a valid certificate (*i.e.*, 11 violations). What is particularly worrisome is that apps such as *MyQ Chamberlain* and *Wemo* have multiple SSL vulnerabilities as well as the *Away read/write* permission. Next, we demonstrate an end-to-end attack on the Nest security camera, using one of the SSL vulnerabilities discovered from this analysis, and the *NestAway read/write* permission.

### 3.5 Lateral Privilege Escalation

While our findings from the previous sections are individually significant, we demonstrate that they can be combined to form an instance of a lateral privilege escalation attack [162], in the context of smart homes. That is, we demonstrate how *an adversary can compromise one product (device/app) integrated into a smart home, and escalate privileges to perform protected operations on another product, leveraging routines configured via the centralized data store.*

This attack is interesting in the context of smart homes, because of two core assumptions that it relies on (1) low-integrity (or non-security) smart home products may be easier to directly compromise than high-integrity devices such as the Nest Cam (*i.e.*, none of the SSL vulnerabilities in  $\mathcal{F}_{10}$  were in security-sensitive apps), and (2) while low-integrity devices may not be able to directly modify the state of high-integrity devices ( $\mathcal{F}_1$ ), they may be able to indirectly do so via *automated routines* triggered by global smart home variables ( $\mathcal{F}_4$ ). (3) Moreover, since the low-integrity device is not being intentionally malicious, but is compromised, the product review process would not be useful, even if it was effective (which it is not, as demonstrated by  $\mathcal{F}_5 \rightarrow \mathcal{F}_9$ ). This last point distinguishes a lateral privilege escalation from actions of malicious apps that trigger routines (*e.g.*, the “fake alarm attack” discussed in prior work [66]). These conditions make lateral privilege escalation particularly interesting in the context of smart home platforms.

**Attack Scenario and Threat Model:** We consider a common man-in-the-middle (MitM) scenario, similar to the SSL-exploitation scenarios that motivate prior work [60, 172]. Consider Alice, a smart home user who has configured a security camera to record when she is away (*i.e.*, using the *away* variable in the centralized data store). Bob is an acquaintance (*e.g.*, a disgruntled employee or an ex-boyfriend) whose motive is to steal a valuable from Alice’s house without being recorded by the camera. We assume that Bob also knows that Alice uses a smart switch in her home, and controls it via its app, which is integrated with Alice’s smart home. Bob follows Alice, and connects to the same public network (*e.g.*, a

Listing 3.1: The Kasa app’s unencrypted GET request.

```
1 {"data":{"uri":"com.tplinkra.iot.authentication.impl.RetrieveAccountSettingRequest"},
2   "iotContext":
3     {"userContext":{"accountToken":"<anonymized_alphanumeric_token>"},
4     "app":{"appType":"Kasa_Android"},
5     "email":"<anonymized>",
6     "terminalId":"<anonymized>"}}, ...
```

coffee shop), sniffs the access token sent by the switch’s app to its server using a known SSL vulnerability in the app, and then uses the token to directly control the *away* variable. Setting the *away* to “home” confuses the security camera into thinking that Alice is at home, and it stops recording. Bob can now burglarize the house without being recorded.

**The Attack:** The example scenario described previously can be executed on a Nest smart home, using the Nest Cam and the TP Link Kasa switch (and the accompanying Kasa app). We compromise the SSL connection of Kasa app, which was found to contain a broken SSL TrustManager in our analysis described in Section 3.4. We choose Kasa app as it requests the sensitive *Away read/write* permission, and has a sizable user base (1M+ downloads on Google Play [80]). It is interesting to note that the Kasa app has also passed the Nest product review process and is advertised on the Works with Nest website [140], but can still be leveraged to perform an attack. We use *bettercap* [23] as a MiTM proxy to intercept and modify unencrypted data. Additionally, as described in the attack scenario, we assume that (1) the victim’s Nest smart home has the Nest Cam and the Kasa switch installed, (2) the popular routine which triggers the Nest Cam to stop recording when the user is home is enabled, and (3) the user connects her smartphone to a network to which the attacker has access (e.g., coffee shop, office), which is a common assumption when exploiting SSL-misuse [60, 172].

The attack proceeds as follows: (1) The user utilizes the Kasa app to control the switch, while the user’s mobile device is connected to public network. (2) The attacker uses a MiTM proxy to intercept Kasa app’s attempt to contact its own server, and supplies the attacker’s certificate to the app during the SSL handshake, which is accepted by the

Kasa app due to the faulty TrustManager. (3) The Kasa app then sends an authorization token (see Listing 3.1) to the MiTM proxy (*i.e.*, assuming it is the authenticated server), which is stolen by the attacker. This token authorizes a particular client app to send commands to the TP Link server. (4) Using the stolen token, the attacker instructs the TP Link server to set the smart home’s *away* variable to “home”, while the user is actually “away”. This action is possible as the TP Link server (*i.e.*, Web app) has the *-Away read/write* permission for the user’s Nest smart home. (5) This triggers the routine in the Nest Cam, which stops recording.

In sum, the attacker compromises a security-insensitive (*i.e.*, low-integrity) product in the system, and uses it along with a routine to escalate privileges, *i.e.*, to modify the state of a security-sensitive (*i.e.*, high-integrity) product. It should be noted that while this is one verified instance of a lateral privilege escalation attack on DSB smart home platforms, given the broad attack surface indicated by our findings, it is likely that similar undiscovered attacks exist.

## 3.6 Vulnerability Reporting Experience and Current Status

We reported the discovered vulnerabilities to Philips ( $\mathcal{F}_2$ ,  $\mathcal{F}_3$ ), Nest/Google ( $\mathcal{F}_4 \rightarrow \mathcal{F}_{10}$ ), and TP Link ( $\mathcal{F}_{10}$ ) in mid-2018. Since then, vendors have responded, confirmed our findings, and even deployed fixes. This section describes our reporting experience with the vendors, if any fixes were deployed, the effectiveness of those fixes, backed by additional experimental analysis.

### 3.6.1 SSL Vulnerability in TP Link’s KASA

We reported the details of the SSL vulnerability exploited in Section 3.5 to TP-Link, who acknowledged the issue and resolved it to a bug in the Android 4.x compatibility library. While TP Link did not elaborate on the exact part of the library that was problematic, they stated that future updates of the Kasa app would contain a fix. We statically and

dynamically analyzed the most recent version of the Kasa app (version 2.13.0.858), and confirmed that (1) the vulnerable lines of code (*i.e.*, a TrustManager that accepts all certificates) were still present, however, (2) they were not being used for SSL connections, as our dynamic MiTM attack (Section 3.5) did not work.

### 3.6.2 Vulnerable Nest routines, and misinformation in third-party Works with Nest apps

Nest does not have a dedicated issue tracker for developers to report security vulnerabilities. Therefore, we provided a detailed bug report to Nest through their customer support, split into two reports: (1) Report 1, describing the vulnerability of security-sensitive Nest devices to lateral privilege escalation, via routines ( $\mathcal{F}_4$ ), and (2) Report 2, describing the inconsistent prompts, permission descriptions, and SSL misuse in third-party Works with Nest apps ( $\mathcal{F}_6, \mathcal{F}_7, \mathcal{F}_9, \mathcal{F}_{10}$ ), as well as the problems in Nest’s product review process ( $\mathcal{F}_5$  and  $\mathcal{F}_8$ ).

**1. Response from Nest:** Nest did not confirm Report 1, but acknowledged that it was forwarded to their concerned engineers. Moreover, Nest recommended us to *publicly disclose* Report 2, *along with the identities of the offending apps, in the Nest community forum*<sup>2</sup> to bring the matter to the attention of Nest developers. However, as Report 2 contained sensitive information involving potentially vulnerable and overprivileged apps, we decided against disclosing it in a public forum and asked Nest to notify the developer through a private channel, but received no further response.

Due to the lack of a sufficient response from Nest, we directly submitted two reports to Google through their bug reporting system<sup>3</sup>. Note that Nest operated independently from Google from 2015 to 2018, and hence, reporting to Google was a non-obvious step at that time.

---

<sup>2</sup><https://www.nest-community.com/s/>

<sup>3</sup><https://www.google.com/appserve/security-bugs/m2/new>

**2. Google’s response to Report 1 and current status:** Initially, a member of Google’s security team suggested that the lateral privilege escalation was purely due to the SSL vulnerability in TP Link’s KASA app, and hence, not relevant to Google or Nest. However, we explained how routines in Nest were key for the attack, which could be leveraged by compromising any low-security device (and not just KASA), the engineers assigned to the bug report acknowledged the existence of a design-level flaw in Nest routines. Further, we also clarified that the attacker did not have to be on the victim’s network to perform the attack. When asked for suggestions to improve the design and rectify the flaw, we provided three concrete recommendations, namely: warning users before enabling routines that affect security-sensitive devices, thorough application reviews to identify overprivileged apps, and temporarily suspending/deprecating routines that affect security-sensitive devices based on a state variable that can be written by untrusted third-party applications. However, we realize that either of these suggestions may not be acceptable to the platform considering the negative impact on user experience. *This exchange highlights the need for platforms to make changes at a design level, as fixing these problems after they have already occurred is hard.*

**3. Google’s response to Report 2 and current status:** As of today, most of the issues described in this report remain un-addressed, *including the instances of misinformation in the Nest documentation itself* (see  $\mathcal{F}_7$ ). Google’s response was that the onus of fixing these issues in apps was on the third-party app developers to review the permissions that their apps request. Due to this, Google, like Nest, suggested that the findings could be disclosed to the developers either directly or through their product forums. However, it should be noted that all the reported apps had undergone direct scrutiny from Nest through their review process and passed that process before deployment to the end-user. Moreover, overprivileged apps that violate the platform’s review process are harmful for the platform, as they may be leveraged by attackers to perform privilege escalation, as demonstrated previously in this chapter. This exchange brings up a crucial question, for situations

where the platform may not be willing to even address the over-privilege in existing apps when reported by consumers or researchers: *Who should the end-user should deem liable in the instance of a security incidence involving a smart home app; the developer of an overprivileged app, or the platform that vetted the app and allowed users to install it?*

### 3.6.3 Permission Enforcement Vulnerabilities in Hue

We reported findings  $\mathcal{F}_2$  and  $\mathcal{F}_3$  to Philips Lighting (*i.e.*, the owner of the Hue brand) along with a proof-of-concept script demonstrating the attacks. Philips Lighting acknowledged the existence of the vulnerabilities and confirmed that they were working on a firmware update that would deploy access control policies to address the issues, and were manually curating a list of verified apps as a stopgap measure in the meanwhile.

Hue informed us that the latest release version 1931069120 mitigates these vulnerabilities. According to the API changelog published with the firmware update, Hue claims to have made two key changes to the Hue data store to address  $\mathcal{F}_2$  and  $\mathcal{F}_3$ , *i.e.*, ensured that (1) applications cannot write to the *linkbutton* variable and (2) whitelist entries can only be deleted via a *cloud application-key*.

This section describes our efforts to experimentally evaluate these claims, and their effectiveness at addressing  $\mathcal{F}_2$  and  $\mathcal{F}_3$ . Our analysis relies on the two kinds of third-party apps allowed on Hue, *i.e.*, *local* and *cloud* apps, which we describe next, followed by our methodology and findings.

**Local and Cloud Hue apps:** Our exploits for the Philips Hue platform demonstrated in Section 3.2 ( $\mathcal{F}_2$  and  $\mathcal{F}_3$ ) can be executed from a *local app*, *i.e.*, a third-party app installed on a device connected to the same local network as the Hue bridge. For a feasible attack via a local app, the attacker-controlled app simply needs to be on the same network (*i.e.*, not necessarily on a device owned by the user). However, Hue supports another kind of third-party app, a *cloud app*, which uses the Hue remote API to remotely issue commands to the lights, and unlike local apps, does not need to be connected to the local network

(however, it does need a proxy local app, as we discuss later).

### 3.6.3.1 Analyzing the Updated Hue API, from both local and remote apps

We tested the effectiveness of Hue’s mitigations from both an attacker-controlled local app (wmlocalapp), as well as a cloud app (wmremoteapp). That is, we first used wmlocalapp (created in Section 3.2) to test whether our exploits for  $\mathcal{F}_2$  and  $\mathcal{F}_3$  still worked on the new Hue local API. Then, we created a cloud app named wmremoteapp in Hue’s developer portal by specifying only its name, a brief description, and OAuth callback URL, which was instantly approved after submission to Hue (*i.e.*, may not have undergone any review, beyond some extremely lightweight static analysis at most).

Further, as all the commands to the Hue lights are executed through the Hue bridge, we had to register a local app that would act as a proxy for the cloud app to execute requests through the Hue bridge (*i.e.*, wmlocalproxyapp). The process for registering this proxy is interesting, as it has direct implications on Hue’s security claims, and our findings from this experiment: we used the access token for the remote app to *remotely issue a linkbutton=true command to the hue endpoint URL <https://api.meethue.com/bridge/0/config>*, which simulated the button-press on the Hue bridge for a brief period of time, within which wmremoteapp issued a POST request to whitelist its local proxy app (*i.e.*, wmlocalproxyapp). At the end of this process, we were issued a *cloud application key*, with which we could make calls identical to the local API calls, via the endpoint URL `https://api.meethue.com/bridge/ < application – key >`. We used this cloud application key to attempt our exploits for  $\mathcal{F}_2$  and  $\mathcal{F}_3$  in a scenario where the attacker controls a cloud app.

### 3.6.3.2 Key Results from Updated Analysis

Our analysis reveals that while Hue’s changes indeed address some of the major shortcomings of its access control policy, the platform is still vulnerable.

Specifically, we confirmed that  $\mathcal{F}_3$  *no longer affects Hue*, *i.e.*, the new version prevents apps from deleting other apps from the whitelist. Further reverse-engineering revealed that Hue enforces this policy by obfuscating the *application-key* of the apps in the whitelist section of the data store. That is, apps cannot delete what they cannot address. However, this also means that the effectiveness of the mitigation relies on the complexity of the obfuscation; it will be invalid once an adversary devises a way to generate obfuscated names from arbitrary application metadata. An access control policy for the whitelist (*i.e.*, which Hue had discussed with us earlier as a possible mitigation) would be a more fundamental solution to this problem. Further, we discovered that  $\mathcal{F}_2$  *still holds*, and even bypasses Hue’s product review-based defenses, which leads to the following finding:

**Finding 11. Cloud apps can bypass user consent repeatedly ( $\mathcal{F}_{11}$ ).** Since a cloud app must have an accompanying local app to execute commands using the Hue bridge, it is reasonable to allow cloud apps with a valid OAuth token to modify the *linkbutton* and add their local counterpart app remotely. However, in our experiments, we discovered that *wmcloudapp* could modify *linkbutton repeatedly*, and thus, register *multiple local apps*. Moreover, local apps are not bound to the remote app that installed them. Hence, *wmcloudapp* could install as many local apps as we wanted, and they would persist even after the user removed the misbehaving *wmcloudapp*. The most important facet of this problem is that our misbehaving *wmcloudapp* is registered with Hue, and hence it would have been possible for users to install it. However, note that we ensured that our misbehaving app was clearly marked as a test application, and that no real user installed it during our experiments.

### 3.7 Feasibility of Analyzing Evolving Smart Home Platforms

The market for smart home products and platforms is now reaching a critical mass in terms of consumer adoption. This has resulted in an ecosystem of rapidly evolving and fragmented platforms. As of now, we do not have a concrete understanding of how platform

evolution helps, or hurts, the *applicability* of existing security analysis approaches. Acquiring such an understanding would be instrumental in helping future security researchers recognize the opportunities as well as challenges posed by evolving characteristics of smart home platforms.

We pose a seemingly simple but nuanced research question: *How feasible would the analysis performed in our work be on smart home platforms in the near future?* To address this question, we (1) identify the essential, platform-independent properties that facilitate the security analyses explored in this work, and (2) evaluate six additional platforms to understand if they exhibit these properties. We conclude the section by identifying the foremost challenge for similar research in the future, drawing from the evidence obtained in our evaluation.

### 3.7.1 Platform-independent *essential properties*

The security evaluation performed in Sections 3.2→3.4 can be categorized into five independent analyses: ( $\mathcal{A}_1$ ) an analysis of platform *permission enforcement*, ( $\mathcal{A}_2$ ) the accuracy of *install-time permission descriptions*, ( $\mathcal{A}_3$ ) the accuracy of *runtime permission prompts*, ( $\mathcal{A}_4$ ) the security *impact of routines*, and ( $\mathcal{A}_5$ ) *SSL misuse* by third party mobile apps. We now identify the five platform-independent *essential properties* that facilitate these analyses:

**Property 1 - ( $\mathcal{P}_01$ ): Availability of public API access to test permission enforcement.** In order to test whether the purported permission enforcement mechanisms that exist in a given platform function properly in practice, it is necessary to have access to public facing platform APIs that would enable us to generate permission maps via automated testing (*i.e.*, **for**  $\mathcal{A}_1$ , Section 3.2).

**Property 2 - ( $\mathcal{P}_02$ ) Platform-mandated *third-party-specified* permission descriptions.** Smart home platforms generally inform users about the effect of platform permissions (*e.g.*, that the home/away r/w permission can “Set Home and Away”, as seen in

Figure 3.2). However, some platforms (*e.g.*, Nest) may also require developers to provide additional context to the user, via *install-time permission descriptions* describing *why* their app needs a particular permission. The availability of such descriptions is critical for understanding how applications may misinform users about their actual intent, and violate platform design policies (*i.e.*, **for**  $\mathcal{A}_2$ , Section 3.4.1).

**Property 3 - ( $\mathcal{P}_03$ ) Platform-mandated *third-party-specified* runtime permission prompts.** In addition to install-time descriptions, platforms may also require applications to use run-time prompts before performing a sensitive action (*e.g.*, as Nest does for home-/away), thereby allowing the user to make a more informed decision. These prompts are necessary to understand if a third-party application’s actual use of a permission is valid (*i.e.*, **for**  $\mathcal{A}_3$ , Section 3.3).

**Property 4 - ( $\mathcal{P}_04$ ) Published third-party routines for home automation.** Routines or automations are generally supported by platforms through third-party integrations, *i.e.*, by integrating devices directly via Zigbee or Z-wave, or indirectly by provisioning API access to third-parties, or through third-party *IoT apps* hosted on the platform itself (*e.g.*, SmartThings SmartApps). As routines may be exploited by attackers, the availability of third-party routines is critical for assessing the presence or prevalence of vulnerabilities that would facilitate attacks such as the lateral privilege escalation attack explored in this work (*i.e.*, **for**  $\mathcal{A}_4$ , Sections 3.3 and 3.5).

**Property 5 - ( $\mathcal{P}_05$ ) Availability of third-party mobile applications.** Smart home platforms are inextricably tied to mobile apps that provide users with a convenient means of controlling various aspects of their smart home, and even facilitate routines (*e.g.*, Yeti [226] and Yonomi [227]). The availability of mobile apps is not only needed for analyzing the security of the communications used by the apps themselves (*i.e.*, **for**  $\mathcal{A}_5$ , Section 3.4.2), but also for understanding the use of permission descriptions and prompts by third-parties (*i.e.*, **for**  $\mathcal{A}_2 \rightarrow \mathcal{A}_3$ , Sections 3.3 and 3.4.1).

### 3.7.2 Evaluation of 6 Additional Smart Home Platforms

We analyzed six smart home platforms (in addition to Nest and Hue) for the presence of properties  $\mathcal{P}_{01} \rightarrow \mathcal{P}_{05}$ , in order to understand the feasibility of performing  $\mathcal{A}_1 \rightarrow \mathcal{A}_5$  on them. Table 3.2 summarizes the results of this feasibility analysis. We now provide a brief overview of our general empirical evaluation methodology, followed by the results of the feasibility analysis for each platform.

**General Evaluation Methodology:** We followed a systematic, 4-step methodology for the feasibility analysis: **(1) Platform Selection.** We selected six platforms from popular publicly available smart home platforms, based on one foundational trait that precedes  $\mathcal{P}_{01} \rightarrow \mathcal{P}_{05}$ : *allowing the integration* of third-party routines, mobile apps, and devices.

**(2) Testing for Public APIs.** For each platform, we then determined the availability of *public* APIs from all available sources (*e.g.*, documentation, official website). If we could register as a developer with the platform, acquire an API key, and make API calls to access platform resources, we considered  $\mathcal{P}_{01}$  satisfied (*i.e.*, conversely, platforms that allowed API access to a limited/closed set of partners did not satisfy  $\mathcal{P}_{01}$ ). **(3) Analyzing Permission Models.** We examined the provided developer documentation to extract the permission model, and to determine if developers were required to specify custom install-time permission descriptions ( $\mathcal{P}_{02}$ ) and runtime prompts ( $\mathcal{P}_{03}$ ) to provide users with more context. Moreover, we examined whether the prompts could be *programmatically triggered* for analysis through integration of our own test app/device to the platform (*i.e.*, and hence, tested the *extent* to which  $\mathcal{P}_{03}$  was satisfied). **(4) Mining third-party clients.** We tried to acquire artifacts that represent routines, such as IoT apps published in markets (*e.g.*, the SmartThings public repo [192]), descriptions of automation in text-form on the platform’s website (*e.g.*, the Works with Nest website [143]), or automations enabled by third-party mobile apps integrated with the platform. Aside from testing for  $\mathcal{P}_{04}$ , this step also allowed us to test for  $\mathcal{P}_{05}$  (*i.e.*, as we searched for mobile apps as well).

We carefully considered platform-specific nuances when executing Steps 1–4, and *ex-*

*perimentally confirmed our claims for all platforms.* The rest of this section provides a brief overview of each analyzed platform, followed by a summary of our analysis results.

### 3.7.2.1 Nest v1 and v2

Google is closing its *Works with Nest* platform on August 31, 2019 in favor of a more tightly-integrated *Works with Google Assistant* platform.<sup>4</sup> We term this new platform Nest v2, while the current version we analyzed in Sections 3.2→3.5 of this chapter is termed as Nest v1. The transition from v1 to v2 impacts the feasibility of security analysis, as it changes the fundamental nature of Nest, *i.e.*, from an open, decentralized platform to a relatively closed platform (*i.e.*, with API access to select vendors) centralized around the Google Assistant.

**Results of the Feasibility Evaluation for Nest v2:** From our analysis, we conclude that the *closed* nature of Nest v2 violates most of the properties, rendering corresponding analyses performed in this chapter infeasible. For instance, the ability of researchers to access the API in Nest v2 will be constrained, as the platform is geared towards helping vendors integrate their devices or products with Google Assistant. At most, researchers will be able to create their own virtual device and an interface for Google Assistant to access that device (*i.e.*, unlike Nest v1, which has a general-purpose public API that can be used to access multiple *other* devices and resources). Thus, Nest v2 violates  $\mathcal{P}_01$ , making it infeasible to automatically test for gaps in access control enforcement ( $\mathcal{A}_1$ ).

Further, Nest v2 does not require developers to write custom permission descriptions or prompt the user before using a permission, as *permissions are acquired by Google Assistant when integrating the device with the platform.* Hence, Nest v2 loses the context of requiring/using permissions, violates  $\mathcal{P}_02$  and  $\mathcal{P}_03$ , and invalidates  $\mathcal{A}_2$  and  $\mathcal{A}_3$ . Moreover, routines will only be created and managed via Google Assistant, which means that no

---

<sup>4</sup><https://blog.google/products/google-nest/helpful-home/>

repositories of routines will be available, requiring researchers to analyze *potential* routines (e.g., from integrations described on the Works with Google Assistant website). Hence,  $\mathcal{P}_04$  is partially satisfied, and it may be somewhat feasible to perform  $\mathcal{A}_4$ , although incredibly difficult to do so with completeness or at scale. Finally, since the Google Home mobile app is the only official way for the user to access the platform, we do not foresee the development of third-party mobile apps that integrate with Nest v2. However, it is common for vendors to provide mobile apps as alternate mediums to control their devices, and since some vendors are being tightly integrated to Nest v2 after a thorough review process <sup>5</sup>, Nest v2 may potentially partially satisfy  $\mathcal{P}_05$ , and hence,  $\mathcal{A}_5$ .

### 3.7.2.2 SmartThings Classic and v2

A particularly interesting aspect of SmartThings is that it allows developers to publish Groovy-based IoT apps (*i.e.*, called SmartApps) in a platform-provided market. This existing SmartThings “Classic” platform is now being phased out in favor of the new SmartThings v2 platform<sup>6</sup> launched on March 18, 2018 that drastically deviates from this characteristic, *i.e.*, SmartThings v2 has eliminated Groovy-based SmartApps. Instead, SmartApps are now manifested as Web hook endpoints [191] or AWS Lambda functions [190] in SmartThings v2, which integrate with SmartThings via its API.

#### Results of our Feasibility Analysis for SmartThings Classic and SmartThings

**v2:** Our analysis confirms that all properties except  $\mathcal{P}_02$  and  $\mathcal{P}_03$  hold for SmartThings Classic (*i.e.*, as SmartThings does not mandate developer-specified permission descriptions or prompts). Hence, a majority of our analyses are feasible on the classic version (*i.e.*,  $\mathcal{A}_1$ ,  $\mathcal{A}_4$ , and  $\mathcal{A}_5$ ). However, the changes in SmartThings v2 make  $\mathcal{A}_4$  and  $\mathcal{A}_5$  partially infeasible. Specifically,  $\mathcal{P}_04$  is affected due to the lack of centrally published and hosted SmartApps in SmartThings v2, *i.e.*, as SmartApps will be reduced to remote endpoints whose code is unavailable for analysis, which will leave researchers with only text descriptions of routines

<sup>5</sup><https://www.blog.google/products/google-nest/updates-works-with-nest/>

<sup>6</sup><https://blog.smartthings.com/news/smartthings-updates/the-new-smartthings-app-is-here/>

on vendor websites, rendering  $\mathcal{A}_4$  partially feasible. Similarly, while third-party mobile app integration is technically possible, it is currently unavailable, which means that  $\mathcal{P}_05$  does not fully hold, and performing  $\mathcal{A}_5$  would be infeasible at least in the near future.

### 3.7.2.3 HomeKit

Apple HomeKit [95] is a proprietary framework that allows interaction among different devices (called *accessories*) in the home through iOS apps. Once the accessories are integrated into the HomeKit framework, users can remotely control or automate them via iOS apps.

**Results of Feasibility Analysis for HomeKit:** While HomeKit is a closed platform similar to Nest v2, it does provide hobbyists with API-support to explore/test the platform. This access would allow researchers to create their own accessories (*i.e.*, devices), while the typical iOS testing and development tools may be used for analyzing the permission enforcement, and access to these devices (*i.e.*, fully satisfying  $\mathcal{P}_01$  and facilitating  $\mathcal{A}_1$ ). Further, developers are required to specify “usage descriptions”, in a “NSHomeKitUsageDescription” field, which is why  $\mathcal{P}_02$  holds, facilitating  $\mathcal{A}_2$  (however,  $\mathcal{A}_3$  is not applicable as there are no mandated prompts). Routines are not available in one place, but can be acquired by analyzing the Home app, *i.e.*,  $\mathcal{P}_04$  partially holds, and hence  $\mathcal{A}_4$  is partially feasible. Finally, as mobile apps are integral to this model,  $\mathcal{A}_5$  is feasible.

### 3.7.2.4 Home Assistant

Home Assistant [93] is an open-source framework for smart home management. Unlike other proprietary platforms where the users need to rely on the server for communication between devices, Home Assistant gives an option of hosting the server locally.

**Results of Feasibility Analysis for Home Assistant:** Home Assistant’s open nature provides valuable opportunities for analysis. For instance, it is open source, allowing researchers to build it locally and automate the creation of the permission map as we

discovered in our initial exploration (*i.e.*,  $\mathcal{A}_1$  is feasible). Note that Home Assistant does not enforce device-level permissions, but instead, enforces access control among multiple users (*i.e.*, hence, the scope of the permission map changes). Publicly available automations [94] satisfy  $\mathcal{P}_04$  and facilitate  $\mathcal{A}_4$ . Similarly, third-party apps for Home Assistant are not numerous, but exist, satisfying  $\mathcal{P}_05$  and facilitating  $\mathcal{A}_5$ . However, Home Assistant does not exhibit  $\mathcal{P}_02$  and  $\mathcal{P}_03$  due to the uniqueness of its permission model, *i.e.*, the user can directly define centrally managed groups that have a specific access to certain smart home resources, which precludes permission descriptions or prompts, making  $\mathcal{A}_2$  and  $\mathcal{A}_3$  inapplicable.

### 3.7.2.5 OpenHAB

OpenHAB is an open-source framework that users can host locally or on the OpenHAB cloud service. Devices (*i.e.*, *things*) are integrated with OpenHAB via *bindings* (*i.e.*, similar to device handlers in SmartThings). Users can then leverage these integrated things to create routines (called rules). While OpenHAB provides bindings for various communication protocols, there is no permission system in place to connect the third-party service.

**Results of Feasibility Analysis for OpenHAB:** OpenHab is similar to Home Assistant in that it is a highly customizable open platform, but unlike Home Assistant, there is no permission enforcement system in place. Thus,  $\mathcal{P}_01 \rightarrow \mathcal{P}_03$  do not hold for OpenHAB, and  $\mathcal{A}_1 \rightarrow \mathcal{A}_3$  are not applicable. However, as a significant number of OpenHAB rules (*i.e.*, routines) can be found in dedicated forums,<sup>7</sup>  $\mathcal{P}_04$  holds, and  $\mathcal{A}_4$  is feasible. Similarly, because there are third party apps that interface with this platform ( $\mathcal{P}_05$ ) an app-based analysis of this platform ( $\mathcal{A}_5$ ) is possible.

---

<sup>7</sup><https://community.openhab.org/c/tutorials-examples>

### 3.7.3 The challenge for future security research

Our feasibility evaluation reveals several interesting aspects of the smart home ecosystem. For example, some platforms such as Apple HomeKit, Home Assistant, and OpenHAB do not implement permissions at the granularity of a device (*i.e.*, instead, only implement multi-user separation, which is further absent in OpenHAB), a coarse-grained model that would be trivial to exploit once an authorization token is stolen (*i.e.*, even without a transitive exploit such as a lateral privilege escalation). More importantly, it demonstrates alarming trends for future research in this area. That is, none of the platforms are amenable to all of  $\mathcal{A}_1 \rightarrow \mathcal{A}_5$ , even partially (*i.e.*, except Nest v1 which was the initial focus of this chapter). More importantly, we see that as platforms evolve, they become less open and transparent to introspection by security researchers. For instance, SmartApps in SmartThings v2 are hidden behind endpoints on the Web, and no longer as open to scrutiny as those in SmartThings v1. Similarly, Nest v2 abstracts platform API, routines, and most functionality behind the Google Assistant API, which is not public and only available to certain certified partners. This is in complete contrast with the publicly accessible API of Nest v1 that enabled the analysis in our work. To continue investigating the security of smart home platforms, researchers must overcome the overwhelming *challenge of* (1) *identifying and mining novel sources of routines and apps at scale, and* (2) *developing alternate methods of accessing platform APIs, which includes engaging platform vendors to acquire official API access.*

## 3.8 Lessons from the Study

Our findings  $(\mathcal{F}_1) \rightarrow (\mathcal{F}_{11})$  demonstrate numerous gaps in the security of DSB platforms. We now distill the core lessons from our security findings from Nest and Hue, as well as the feasibility analysis with six additional platforms.

**Lesson 1** : *Seamless automation must be accompanied by strong integrity guarantees.* It

is important to note that the attack described in Section 3.5 can not be addressed by reducing overprivilege or via product reviews, since none of the components of the attack are overprivileged (*i.e.*, including TP Link Kasa), and our findings demonstrate that the Nest product review is insufficient ( $\mathcal{F}_5 \rightarrow \mathcal{F}_9$ ). The attack was possible due to the integrity-agnostic execution of routines in Nest ( $\mathcal{F}_4$ ). To mitigate such attacks, platforms need information flow control (IFC) enforcement that ensures strong integrity guarantees [25], and future work may explore the complex challenges of (1) specifying integrity labels for diverse devices and (2) enforcing integrity constraints without sacrificing automation.

**Lesson 2:** *Nest Product Reviews would benefit from at least light-weight static analysis.* Our findings demonstrate numerous violations of the Nest design policies that should have been discovered during the product review. Moreover, the review guidelines also state that products that do not securely transmit tokens will be rejected [137], but our simple static analysis using MalloDroid discovered numerous SSL vulnerabilities in Nest apps ( $\mathcal{F}_{10}$ ), of which one can be exploited (Section 3.5). We recommend the integration of light-weight tools such as MalloDroid in the review process.

**Lesson 3:** *The security of the smart home indirectly depends on the smart phone (apps).* Smartphone apps have been known to be susceptible to SSL misuse [60], among other security issues (*e.g.*, unprotected interfaces [35]). Thus, unprotected smartphone clients for smart home devices may enable the attacker to gain access to the smart home, and launch further attacks, as demonstrated in Section 3.5. Ensuring the security of smart phone apps is a hard problem, but future work may triage smartphone apps for security analyses based on the volume of smart home devices/platforms they integrate with, thereby, improving the apps that offer the widest possible attack surface.

**Lesson 4:** *Popular but simpler platforms need urgent attention.* The startling gaps in the access control of Hue demonstrate that the access control of other simple (*i.e.*, homogeneous) platforms may benefit from a similar holistic security analysis ( $\mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_{11}$ ).

**Lesson 5:** *New Analysis Methods are required as smart home platforms become more re-*

*strictive to integrations.* Our feasibility analysis in Section 3.7 demonstrates how popular smart home platforms are becoming less transparent, and more amenable to security analysis. While this tighter control can help to alleviate certain security problems such as public API misuse, or side-stepping review protocols, it also shifts more control into the hands of the platforms, making them more difficult to examine. Thus, new methods of analysis that work within the boundaries of modern platform restrictions are needed. For instance, acquiring and studying the security implications of the increasingly common user-driven routines (*i.e.*, those created by users through interactive platform-provided UIs) offers a potentially viable alternative to studying the developer-provided IoT apps.

### 3.9 Chapter Summary

Smart home platforms and devices operate in the users' physical space, hence, evaluating their security is critical. In this chapter, we evaluate the security of two such platforms, Nest and Hue, that implement home automation *routines* via centralized data stores. We systematically analyze the limitations of the access control enforced by Nest and Hue, the exploitability of routines in Nest, the robustness of Nest's product review, and the security of third-party apps that integrate with Nest. Our analysis demonstrates ten impactful findings, which we leverage to perform an end-to-end lateral privilege escalation attack in the context of the smart home. Our findings motivate more systematic and design-level defenses against attacks on the integrity of the users' smart home.

**Table 3.1:** Permission description violations discovered in Works with Nest apps

Application	Incorrect Permission Description
<b>VC1: Requesting Read/Write instead of Read</b>	
1. Home alerts	“ <b>thermostat read/write:</b> Allows Home alerts to notify you when the Nest temperature exceeds your threshold(s)”
2. Home alerts	“ <b>away read/write:</b> Allows Home Alerts to notify you when someone is in your home while in away-mode”
3. MyQ Chamberlain	“ <b>thermostat read/write:</b> Allows Chamberlain to display your Nest Thermostat temperature in the MyQ app”
4. leakSMART	“ <b>thermostat read/write:</b> Allows leakSMART to show Nest Thermostat room temperature and humidity. New HVAC sensor mode will notify you to shut off your thermostat if a leak is detected in your HVAC system.”
5. Simplehuman Mirror	“ <b>Camera+Images read/write:</b> Allow your simplehuman sensor mirror pro to capture and recreate the light your Nest Cam sees”
6. Iris by Lowe’s	“ <b>structure read/write:</b> View your Nest Structure names so Iris can help you pair your Nest Structures to the correct Iris Places”
7. Heatworks model 1	“ <b>away read/write:</b> Allows the Heatworks MODEL 1 to be placed into vacation mode to save on power consumption while you’re away”
8. Feather Controller	“ <b>Camera+Images read/write:</b> Allows Feather to show you your camera and activity images. Additionally, Feather will allow you to request a snapshot.”
9. Heatworks model 1	“ <b>thermostat r/w:</b> Allows your Heatworks MODEL 1 water heater to go into vacation mode when your home is set to away”
<b>VC2: Describing <i>Away</i> as a property of the thermostat alone, rather than a property of entire home</b>	
10. Gideon	“ <b>away read/write:</b> Allows Gideon to read and update the Away state of your thermostat”
11. Muzzley	“ <b>away read/write:</b> Allows Muzzley to read and update the Away state of your thermostat”
12. Keen home smart vent	“ <b>away read/write:</b> Allows Smart vent to read the state of your Thermostat and change the state from Away to Home”
<b>VC3: Both VC1 and VC2</b>	
13. WeMo	“ <b>away read/write:</b> Allows your WeMo products to turn off when your Nest Thermostat is set to Away and on when set to Home.”
14. IFTTT thermostat service	“ <b>thermostat read/write:</b> Now you can turn on Nest Thermostat Applets that monitor when you’re home, away and when the temperature changes.”
<b>VC4: Descriptions that do not relate to the permission</b>	
15. IFTTT thermostat service	“ <b>away read/write:</b> Now you can set your temperature or turn on the fan with Nest Thermostat Applets on IFTTT”
16. Life360	“ <b>away read/write:</b> We need this permission to automatically turn on/off your nest system”

**Table 3.2:** Feasibility of Analyses  $\mathcal{A}_1 \rightarrow \mathcal{A}_5$  on various smart home platforms.

Analysis	Nest v1	Nest v2	SmartThings Classic	SmartThings v2	HomeKit	Home Assistant	OpenHAB
$\mathcal{A}_1$ : Permission Enforcement	✓	×	✓	✓	✓	✓	×
$\mathcal{A}_2$ : Permission Description Accuracy	✓	×	×	×	✓	×	×
$\mathcal{A}_3$ : Permission Prompt Accuracy	✓	×	×	×	×	×	×
$\mathcal{A}_4$ : Impact of Routines	✓*	✓*	✓	✓*	✓*	✓	✓
$\mathcal{A}_5$ : SSL Misuse in Third Party Apps	✓	✓*	✓	✓*	✓	✓	✓

✓ = feasible, × = not feasible, and ✓\* = partially feasible

## Chapter 4

# HomeEndorser: Practical Integrity

## Validation in the Smart Home

The popularity of smart home devices [199] can be attributed in part to the convenience of home automation, wherein smart home devices automatically react to changes in the user’s physical environment. For example, the user may configure a security camera to begin recording when they leave home, but turn OFF when they return to preserve their privacy [124]. Such automation is often expressed using trigger-action programs known as *routines*, that execute a device action (turning OFF the camera) in response to a trigger (*e.g.*, “away” to “home”).

Routines are often enabled via third-party integrations/services, which automate device-actions by leveraging platform APIs, as we discuss in Chapter 2. Particularly, third-parties use the APIs to modify two distinct types of objects, *device states* (*e.g.*, the ON/OFF state of a light bulb), and *abstract home objects* (AHOs) that are not device-specific (*e.g.*, home/away, hereby referred to as the home AHO). These AHOs are in fact computed in several ways, often through third-party services (*e.g.*, set by the user [114], inferred by querying a combination of device states [97], or using some other proprietary approach [185]).

A particularly dangerous attack vector that emerges from this setting is where adversaries gain privileged access to devices *indirectly*, by falsifying an AHO that a high-integrity

device depends on (via a routine). For instance, consider a situation wherein an adversary may want to disable the security camera to perform a burglary unnoticed, but may not have direct API access to it. Any routine installed that can modify an AHO that the security camera may depend upon to deactivate, such as home/away being set to “home,” may disable the security camera without direct access [112, 113]. At its core, this is *an integrity problem* analogous to those seen in operating systems: a high-integrity process (here, the security camera) relies on the value of an object (*i.e.*, the home AHO), which can be modified by untrusted parties. Hence, smart home platforms must directly address the *lack of integrity validation of AHO-changes caused by automation*. This framing deviates from previously proposed mitigations that aims to reduce over-privilege [171, 119, 212] or restrict permission-use based on the runtime context of an API access [110]. Such mitigations impose infeasible usability penalties, while not addressing the fundamental lack of integrity.

Information flow control (IFC) has often been proposed to ensure the integrity of information consumed by sensitive processes [25, 69, 129, 55, 116, 230], through dominance checks that regulate flows based on subject and object labels [25]. However, a naive IFC approach may result in significant false denials. For instance, IoT platforms may attempt to ensure integrity by marking the home AHO with a high-integrity label and third-party services as low-integrity, which will block a service chosen by the user to update home, resulting in a false denial from the user’s perspective (*e.g.*, 19/33 Nest integrations from a prior dataset [112] would be blocked under this model).

IFC systems rely on *endorsement* [32, 116, 230, 229] to overcome this limitation, allowing trusted programs to change labels of objects to permit flows that would normally violate IFC. However, determining the conditions where an endorsement would be allowable in IoT systems is a challenge; indeed, prior work has often avoided directly addressing this problem, and generally facilitates endorsement by assigning the authority to certain *trusted* high-integrity processes, thereby delegating the task of determining *how to endorse correctly* to the programmer or administrator [116, 230, 37, 182? ]. However, in our case,

smart home users may lack information about dependencies among devices and AHOs to do this correctly. So, we ask instead: Is there something else we can rely on to provide endorsement for *practical* integrity validation?

Yes – the cyber-physical nature of the smart home provides us with a unique opportunity for practical endorsement, in the form of ground truth observations from devices (*i.e.*, device state changes) that can be used to validate proposed changes to AHOs. For instance, we can endorse the change to the home AHO (from “away” to “home”) if the door lock was legitimately unlocked (*i.e.*, with the correct keycode) recently, since the lock being unlocked represents the home owner’s intent and attempt to enter the home. In addition, rather than only depending on one device, we can leverage all the devices that may observe state changes that may correlate with each sensitive AHO change, such as motion sensors, cameras, microphones, etc. that may be available to detect changes that support the home AHO change. Thus, we state the following claim that forms the foundation of this work:

Abstract home objects (AHOs) shared among third-party services and devices for the purpose of home automation are inherently tied to a home’s physical state. Thus, any state change or modification to an AHO via an API call can be *endorsed* using the *local context* of the home, consisting of changes in a combination of device states.

**Contributions in this Chapter:** To answer **RQ<sub>2</sub>**, we introduce the paradigm of *home abstraction endorsement* to validate changes to AHOs initiated by untrusted API calls, and propose the HomeEndorser framework to enable it. HomeEndorser does not continuously monitor AHOs, but focuses on API-induced *changes* to AHOs, and performs a sanity check using *policies* that rely on recent physical state changes in smart home devices. If the check fails, the state change is denied, and the user is informed. HomeEndorser’s preemptive action prevents future automation based on maliciously changed AHOs. We make the following contributions in exploring this novel design space:

**1. Home Abstraction Endorsement:** We introduce the paradigm of *home abstraction endorsement*, which leverages the state changes of local devices to endorse proposed changes to AHOs, thereby making IFC endorsement practical by exploiting the cyber-physical

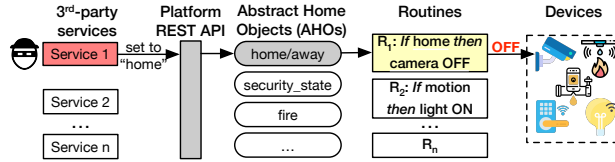
nature of the smart home.

**2. The HomeEndorser Framework:** We design the HomeEndorser framework to implement this practical approach to endorsement, consisting of (1) a *policy model* that allows a unified expression of location-specific device instances within a single policy (*e.g.*, endorsing home via multiple physical entry points), (2) a platform-based *reference monitor* that mediates all sensitive state changes using these policies, and finally, (3) a mechanism to enable experts to generate endorsement policy templates (defined once for all homes), which HomeEndorser then automatically instantiates for each home (considering device availability and placement), enforcing the *most restrictive but feasible policy*.

**3. Evaluation:** We implement HomeEndorser on HomeAssistant, a popular open-source smart home platform, and evaluate it using experimental and empirical analyses. We first demonstrate that the idea of home abstraction endorsement is feasible, even when using a limited set of correlating devices, by generating policies to endorse changes to the home AHO. We demonstrate the generality of our policy model by identifying several attributes that may be used to endorse five additional AHOs. We then demonstrate the effectiveness of HomeEndorser’s integrity validation using a case-study approach. Moreover, we test 10 realistic *home usage* scenarios [111], and 400 realistic *home automation event sequences* [124] in a smart home (apartment) testbed to demonstrate that HomeEndorser is generally not susceptible to false denials, and in fact, may prevent accidental unsafe situations. We demonstrate HomeEndorser’s practical performance overhead with micro/macro benchmarks (9.7-12.2% on average). Finally, we demonstrate the modest effort required to generate policy templates, configure HomeEndorser in end-user homes, and integrate HomeEndorser in popular smart home platforms.

## 4.1 Chapter Motivation

Smart home platforms such as SmartThings [193] and Nest [138] provide permission-protected APIs that enable *seamless integration of diverse third-party services*, ranging



**Figure 4.1:** An attack on the security camera through the manipulation of two shared state objects by adversary-controlled integrations.

from software support for devices (*e.g.*, the TP Link Kasa [80] and Wemo [84] mobile/-cloud apps), to *third-party automation platforms* that enable user/developer-provisioned routines (*e.g.*, IFTTT [101] and Yonomi [227]). Platform-provided API access allows the third-party service to read/write to two broad categories of “states” in the home: (1) *device states*, *i.e.*, values associated with individual devices such as the security camera (*e.g.*, the ON/OFF state, battery status), and (2) *abstract home objects (AHOs)* that are not associated with any specific device, and instead may be *computed* in several ways, often by taking into account *the user’s choice*. For example, an object that stores whether the user is home or away is an AHO (*i.e.*, the home AHO), and a third-party service designated by the user may compute it based on the location of the user’s phone [205], or using a proprietary/undisclosed method/device [185], or through a direct command from the user [114]. The use of AHOs by third-party services may lead to severe consequences for the *integrity* of the home, as we discuss in a motivating example inspired from prior attacks [112, 113]:

**Motivating Example:** Alice has a security camera in her home to deter burglars. For automated monitoring, Alice has configured two routines (as advertised by Simplisafe [184] and Nest [139]): **(R1)** the camera turns ON when Alice leaves home, and **(R2)** the camera turns OFF when Alice returns home. Additionally, Alice has integrated several third-party services (*e.g.*, TP-link Kasa app to control her Kasa switch).

Bob seeks to burglarize Alice’s home when she is away *without being monitored by the camera*, and controls one (or more) third-party services connected to Alice’s home. This can happen through one of several ways: (1) Alice installed Bob’s service, (2) Bob compromised a vulnerable service (as demonstrated by prior work [112][113]). Hence, Bob

can access the APIs with the permissions assigned to the service he controls. Using this access, Bob modifies `home` to the value “home”, falsely suggesting that Alice is home and triggering **(R1)**, thereby disabling the camera. Bob can then burglarize Alice’s home without being monitored, as shown in Figure 4.1.

This problem is not just limited to the `home` AHO. Consider another AHO, the `security_state`, which is often used in routines to control security-related devices such as cameras and glass break detectors [15, 174]. That is, security devices are armed when `security_state` is set to “deter”, and disarmed/ignored when it is set to “ok” (similar to routines from Ring [174] and TotalConnect [15]). If Bob controls a service with access to `security_state`, he can set it to “ok” and disable the camera. That is, adversaries may falsify *one of many AHOs to transitively* manipulate a highly sensitive device.

#### 4.1.1 API Misuse as an OS Integrity Problem

At first glance, the problem described in the motivating example may appear as an “application” security problem, which prior work has attempted to address using program analysis techniques, *i.e.*, by analyzing or instrumenting IoT apps (*i.e.*, developer-defined automation programs) to limit privileges or API use [110, 171, 119, 217, 33, 144, 34, 212]. However, there are two major shortcomings of treating this problem as one of “application” security: (1) *“IoT apps” are not visible/available anymore*. Platforms have consistently deviated from hosting third-party automations/code as “apps” on the platform itself, and towards a model where third parties simply obtain API access, with the code hosted on the third party developer’s cloud [193, 134]. Third parties simply issue commands remotely to the platform’s API endpoints, and hence, the platform (or security enhancements to it) do not have the visibility or access needed to analyze/instrument apps for security. (2) *The fundamental lack of integrity cannot be fixed purely by limiting privileges*. Even when privileges are limited to only third-parties that truly need them (*e.g.*, through program analysis/instrumentation), an adversary may yet compromise a third-party service that needs access (*e.g.*, the Kasa app with access to the `home` AHO [112, 113]), and leverage

its permissions to transitively attack a high integrity device, as shown in the motivating example. Thus, limiting privileges, or assigning them in entirety to a trusted third party (*e.g.*, as proposed in the ESO approach [181]), does not eliminate the issue.

To summarize, Bob transitively attacks a high-integrity device, *i.e.*, by modifying AHOs to trigger routines that manipulate it, because Bob *cannot compromise or modify (via API) directly*. That is, this problem is a smart home-specific instance of the classical *OS integrity problem* (*e.g.*, Biba integrity [25]), wherein a high-integrity process (*i.e.*, the camera) relies on an object (*i.e.*, the home AHO) which can also be modified by low-integrity process (*e.g.*, the Kasa integration). Therefore, our work seeks to treat the underlying limitation of the OS, *i.e.*, the fundamental lack of integrity validation.

#### 4.1.2 The Need for Integrity Validation

As high-integrity devices rely on AHOs such as `home`, traditional wisdom dictates that low-integrity (or third-party) integrations must be disallowed from writing to these objects. However, recall that in existing platforms (*e.g.*, SmartThings and Nest), AHOs are often *computed* by third-party services chosen by the user, and such integrations may genuinely need to modify AHOs to perform their functionality. Thus, disallowing *the user's choice* of third-party integration from writing to AHOs is bound to break numerous useful services that the user relies on (*e.g.*, IFTTT, Yonomi, Kasa), which is a prohibitive cost in terms of user experience that platforms may find undesirable. For example, in 2019, Google announced an end to its “Works with NEST” developer program in favor of a more restricted “Works with Google Assistant” program that would only be open to vetted partners [78], but immediately backtracked [79, 210, 40] given significant opposition from both users and third-party integrations/developers [50, 211, 49, 102, 213], eventually offering a more flexible program that allowed a broader set of developers access to internal home states (including AHOs) [79].

Therefore, there is a need for a solution that is (1) *backwards compatible* or practical, *i.e.*, does not break functionality by preventing third-parties from accessing AHOs, and

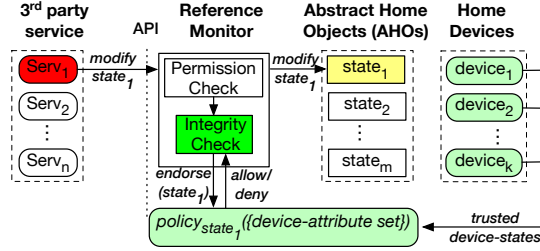
(2) *effective, i.e.*, which enables integrity validation for devices that rely on AHOs. To this end, this work proposes the moderate route of proactive integrity checking, *i.e.*, *runtime validation of proposed changes to AHOs*, which will enable reasonable integrity validation while being compatible with platform design and integration choices.

## 4.2 HomeEndorser Design Goals

This work introduces the novel paradigm of *home abstraction endorsement* that provides a strong *integrity guarantee* for AHOs, defined as follows: In the event that an untrusted service uses the platform API to modify a critical AHO, *e.g.*, `home` or `security_state`, the modification will be allowed iff it is consistent with the *local state of the home*, composed of the physical device states. Our approach builds upon the concept of trusted “guards” in the Biba integrity model [25], wherein a high integrity subject cannot receive input from a low integrity subject unless it is endorsed by a trusted guard. Similarly, in the smart home, we envision endorsement policies that apply trusted device states and hence serve as the trusted guards, ensuring the validity of API requests to change the AHOs that high-integrity devices rely on.

Our design is guided by the following goals:

- G1** *Expressive and Grounded Endorsement Policies.* The endorsement policy structure must be designed in a way that allows it to *express common deployment factors* in smart homes that may affect the endorsement, such as *device availability* and *locality*. Moreover, the policy generation mechanism must be grounded in real devices and their characteristics for practical relevance.
- G2** *Complete Mediation.* Third party services are generally deployed in cloud environments outside the platform’s control, as simple cloud end-points that can make REST API calls [77, 193]. Hence, the reference monitor should be *app-agnostic, i.e.*, should not depend on the analysis/instrumentation of apps/services, but should provide complete mediation for all API calls that modify AHOs, *irrespective of app logic*.



**Figure 4.2:** A conceptual overview of physical home endorsement.

- G3 Tamperproofness.** While our endorsement approach relies on device states, several device states may be modifiable by untrusted services using the platform API. We need to build a reference monitor that *only relies on trustworthy states* modifiable only by devices.
- G4 Freshness.** The task of endorsing an AHO change may require the reference monitor to examine *recent changes* in the states of physical devices, rather than simply reading the current state (*e.g.*, as sensor states may reset after apprising the platform of an event). Hence, the reference monitor must have trusted access to historical states.
- G5 Minimal Performance/Management Overhead.** The framework should minimize any delay *perceivable by the user*, as well as deployment and management effort.

### 4.3 The HomeEndorser Framework

We propose HomeEndorser, a framework that enables home abstraction endorsement by developing the methods and systems necessary to achieve the design goals **G1**→**G5**.

Figure 5.1 provides an overview of our approach. When a third party service attempts to modify an AHO, the platform first enforces a permission check. We envision an additional *integrity check* enabled by HomeEndorser for *endorsing the proposed change*. For arriving at an endorsement decision for the proposed AHO-change, HomeEndorser checks the corresponding *endorsement policy* that relies on recent changes in specific device-attributes/states (*e.g.*, motion detected/undetected, or door lock locked/unlocked). For example, for endorsing a proposed change in the home AHO from “away” to “home”, *one*

possible policy would be as follows: *if door-lock has been unlocked recently (i.e., using the correct keycode)*, then ALLOW the change, else DENY.

HomeEndorser’s policies are *automatically instantiated* in the context of the specific home (*i.e.*, the availability and placement of devices), from *expert-defined policy templates*. To enable this approach of one-time template-definition followed by flexible instantiation that is compatible with all (or most) user homes, we first define a *policy model* that can express deployment-considerations such as device locality and availability, addressing **G1** (Sec.4.3.1). Additionally, we design a *policy-template generation methodology* that allows experts to define endorsement policy templates in a systematic, ground-up manner (Sec. 4.3.3), using (i) automatically-generated *endorsement attributes*, *i.e.*, device-attributes that can be *trusted* for endorsement, due to being either read-only or treated as highly restricted by platforms (ensuring tamperproofness and addressing **G3**), and (ii) a manual open coding approach to identify *inferences* from changes to these endorsement attributes that can be used to endorse specific AHO-changes. On deployment, HomeEndorser uses these expert-defined templates to instantiate a home-specific *most restrictive but feasible policy* for each AHO-change that needs to be endorsed, defined as an instantiation of the policy template that contains the largest aggregate of device-attributes (with the assumption that including more device-attributes would make the inference stronger), but which is also supported given the devices present in the home and their locality.

HomeEndorser’s *reference monitor* is integrated into the user’s smart home platform in the form of an endorsement check in the platform’s subsystem responsible for executing all API calls, thereby ensuring complete mediation, and addressing **G2** (Sec. 4.3.2). When a third-party makes an API call to modify an AHO-change that HomeEndorser endorses, HomeEndorser invokes the most restrictive but feasible policy corresponding to that AHO-change. To check the policy, HomeEndorser uses a *state machine* to retrieve the *most recent change* in each relevant *device-attribute* included in the policy (**G4**). We consider the most recent change, rather than the current state of the device-attribute, as the two may be different (since most sensors reset after a change), and because the most recent

changes provide the context for endorsing the proposed AHO change. This design decision is instrumental in eliminating unnecessary false denials (see Section 4.5.4).

**Threat Model:** We consider a network-based adversary with the ability to control/-compromise any third-party integration/service connected to the target’s home (*e.g.*, as demonstrated by prior work [112, 66]). The attacker’s objective is to *indirectly* modify or disable high integrity devices (*e.g.*, security camera) using compromised or malicious third-party services. The adversary may issue API commands to the target’s home that may affect device states; however, the adversary does not have the ability to *compromise devices* in the home by other means (*i.e.*, which is a standard assumption in work that deals with API misuse [34, 51, 52]). The adversary cannot compromise the platform and the platform app; explicitly, the adversary cannot censor notifications sent from the platform to the user. Devices may be offline. Finally, we do not account for byzantine fault tolerance, but point to complementary prior work [26] that verifies reported device-states.

### 4.3.1 Policy Model

A key challenge for HomeEndorser is designing a policy model that can alleviate two practical constraints. First, unlike our motivating example where a single device-attribute (*i.e.*, the UNLOCKED state of the door lock) was used for endorsing home, in practice, endorsement policies may consist of more than one device-attribute that must be checked together. Second, the device location is an important factor in deciding what they can endorse. For example, while a door lock on the front door as well as a door lock on the back door can both indicate that the user is home, the two endorsements are naturally *mutually exclusive*. We account for these constraints with a policy design that can be expressed as a Disjunctive Normal Form (DNF) boolean formula, as follows:

**Definition 1** (Endorsement Policy). The policy for endorsing a change in AHO  $x$  to value  $y$ ,  $P_x(y)$ , is a DNF formula composed of one or more location-specific predicates ( $L_i$ ), *i.e.*,  $P_x(y) = L_1 \vee L_2 \vee \dots \vee L_n$ , where a location-specific predicate is defined as follows:

**Definition 2** (Location-specific Predicate). A location-specific policy predicate  $L_i$  for

location  $i$  (e.g., entryway), i.e.,  $L_i = d_j \wedge d_k \wedge \dots d_m$ , is a conjunction of one or more device-attribute checks  $d_j$ , defined as follows:

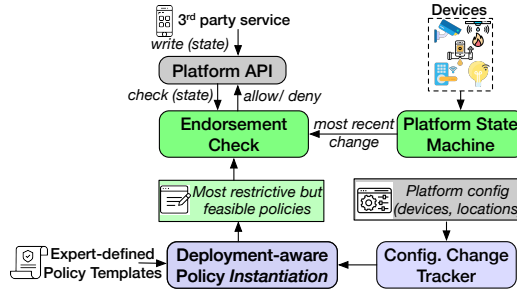
**Definition 3** (Device-attribute Check). A device-attribute check  $d_j$  is a condition  $d_j == s$ , where  $s$  is a physical state that the particular device-attribute must have exhibited in the recent past, for the device-attribute check to return *true*.

To illustrate, let us express the policy from the motivating example for endorsing the home AHO’s change to “home”. We express the policy using a door lock and a motion sensor at the entry way, as well as the same devices at the rear entrance:

$$\begin{aligned}
 P_{\text{home}}(\text{home}) = & (\text{door-lock\_lock} == \text{UNLOCKED} \wedge \\
 & \text{motion\_sensor} == \text{ACTIVE})_{\text{front-door}} \\
 & \vee \\
 & (\text{door-lock\_lock} == \text{UNLOCKED} \wedge \\
 & \text{motion\_sensor} == \text{ACTIVE})_{\text{back-door}}
 \end{aligned}$$

The above policy considers both the door lock being unlocked, and motion being sensed, to prevent false negatives. That is, for both the conditions above to be true, a user would have to unlock the door *and then enter*, i.e., confirming that they are home. On the contrary, if the user unlocks but leaves without entering, this policy condition would correctly result in a denial (as shown in Section 4.5.4). Similarly, the disjunction between the location-specific predicates enables their independent evaluation, thereby allowing the AHO-change as long as either one evaluates to a *true* result. Finally, we define two policy *actions*: ALLOW and DENY, corresponding to the *true* or *false* values that the DNF formula results in, respectively.

For Bob to circumvent HomeEndorser, he could attempt to modify home at the very moment when Alice is performing what are semantically opposite actions, but sufficient for the endorsement of home. For instance, Alice could be *leaving*, which would also involve (1) unlocking the door, and (2) triggering the door-way motion sensor. A naive implementation of our policy model would be susceptible to this attack, as it would consider the above



**Figure 4.3:** Design components of HomeEndorser’s enforcement

device state changes, even if performed in the opposite sense, to be evidence that Alice is returning home, because it matches  $P_{\text{home}}(\text{home})$ . However, smart home devices provide *unique* device attribute values even for similar actions, *i.e.*, the state value for unlocking the door using the keypad is different relative to simply unlocking it from the inside (*i.e.*, “owner” in the former case, and “manual” in the latter). Our policy templates 4.3.3 and enforcement (Section 4.3.2) consider device-attribute values at this precise granularity, preventing such an attack.

### 4.3.2 Secure and Practical Enforcement

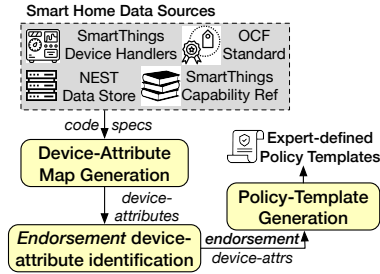
We integrate HomeEndorser’s enforcement into the smart home platform, to enable it to mediate all API commands from third-parties before they are executed (**G2**). This decision is influenced by how third-party services are currently integrated, *i.e.*, as cloud endpoints that use RESTful APIs to interact with the platform, but execute on their own proprietary servers, without a way for the platform to inspect them (*e.g.*, Nest and SmartThings v3). Therefore, our decision ensures that the endorsement check will occur regardless of how the integration is implemented, or where it is deployed. We now describe our platform-level enforcement in terms of its three design components, as shown in Figure 4.3.

**1. Deployment-aware Policy Instantiation:** When HomeEndorser is first set up in a home, it leverages the platform’s internal bookkeeping systems to extract all the devices and device-locations. Then, for each AHO the user decides to endorse, it uses the policy templates generated by experts (as described later in Section 4.3.3 to instantiate the *most*

*restrictive but feasible policy*, *i.e.*, the policy that contains the largest aggregate of device-attributes, given what devices are available in the home, and where they are located. Such dynamic instantiation is necessary to apply the policy templates to *any* home, as (1) a typical user’s setup is unlikely to have *all* the devices specified in the policy template, and simply using all the devices in the template (which would be the *universally most restrictive policy*) would always deny the state change. (2) the policy templates need to be configured as per the different locations in which devices are placed in the home, and (3) the enforced policies need to adapt as the user’s smart home evolves (*e.g.*, through device addition/removal), to provide the enforcement applicable to the current setup. Thus, HomeEndorser computes and then enforces the most restrictive but feasible policy for that specific home, and also reinstantiates/updates the policy upon a configuration change, such as the addition, removal, or relocation of a new device.

**2. The Endorsement Check:** HomeEndorser mediates all API requests, but only invokes the endorsement check if one of the AHOs selected by the user for endorsement is about to be modified, in a manner similar to performance-preserving *hook activations* as previously proposed for Android [91] (**G5**). The check consists of retrieving the policy for the specific AHO-change being endorsed and collecting state information from the device-attributes in the policy. If the policy decision is ALLOW, then the proposed change is allowed to go through; else, it is denied, and the user is notified. A key component of this check is HomeEndorser’s *platform state machine* that retrieves the device-attribute values, as described next.

**3. Retrieving the *most recent* changes using the Platform State Machine:** A naive approach of executing an endorsement check would be to query each device for its *current state* at the time of endorsement. However, such a check would most certainly fail and lead to a false denial because most sensors detect and report a change, and then *reset to a predefined neutral state*. For example, recall the endorsement policy predicate to endorse home consisting of the door lock and the motion detector (assuming one location



**Figure 4.4:** HomeEndorser’s policy specification methodology.

for simplicity):

$$\text{door-lock\_lock} == \text{UNLOCKED} \wedge \text{motion\_sensor} == \text{ACTIVE}$$

Unless the check happens exactly at the moment the user enters, the motion detector will reset to its INACTIVE state immediately after detecting motion, causing a false denial. Thus, for correct endorsement, we check the *most recent but fresh change* in the device states (**G4**), *i.e.*, the last state change before the state automatically reset, within a configurable time threshold to ensure freshness (*e.g.*, one minute).

A state machine that keeps track of all device state changes provides HomeEndorser’s reference monitor with this data, by tracking changes through callbacks placed in the entities that represent devices on the platform (*e.g.*, device handlers in SmartThings, or device-integrations in HomeAssistant). Every time the state of a device changes, the state machine callback is invoked to store the most recent state change along with the timestamp. The timestamp helps HomeEndorser discard states that are older than the preconfigured threshold, thereby preventing old states from causing *false allows* (**G4**).

### 4.3.3 Data-driven Policy Template Generation

Figure 4.4 provides an overview of our data-driven methodology to allow *experts* (*e.g.*, security researchers, platform vendors) to specify endorsement policy *templates* that are representative of what real device states can enable (**G1**), which HomeEndorser automatically instantiates in the context of end-user homes (as previously described in Section 4.3.2). We begin by automatically creating a *device-attribute map*, *i.e.*, a comprehensive mapping

```

metadata {
    definition (name: "Samsung Smart Doorlock",
              capability "Actuator"
              capability "Lock"
              capability "Battery"
              capability "Configuration"
              capability "Health Check"
              capability "Refresh"
    )
}

```

**Figure 4.5:** Specification of device-attributes in the SmartThings device handler preamble.

between device types (*e.g.*, cameras, door locks) and the attributes they embody. We then define the *endorsement attributes* to be used for tamperproof endorsement, and describe our approach for identifying them (**G3**). Finally, we use an open coding methodology for identifying the *observations* and *inferences* that can be made from the endorsement attributes, which are then used to generate policy templates (using the model from Section 4.3.1).

**1. Generating the Device-Attribute Map:** We develop a methodology to create a realistic device-attribute map from platform documentation and existing device-abstractions, such as SmartThings “device handlers”. We select the following information sources for building the map, based on platform popularity, and the potential of obtaining realistic mappings: (1) a device-resource map from the Open Connectivity Foundation (OCF) specification [148], used by the open source platform IoTivity [107], (2) the Nest data store [141], (3) the SmartThings capability reference [189], and (4) SmartThings device handlers [178]. As each of these sources exhibits a unique representation of devices and attributes, we develop customized, automated methods for extracting device-attributes from each source. To elaborate, OCF explicitly specifies a device types, the attributes associated with each device type in a JSON document [149], which we automatically analyze to obtain a device-attribute map for OCF. Similarly, the Nest data store provides a mapping of device-attributes in JSON-like format, which we similarly query. SmartThings provides a capability reference [189], which lists attributes (*i.e.*, capabilities) that devices may choose to exhibit, via developer-defined *device handlers*, which are software proxies for devices that facilitate device-interaction with the platform. To extract a device-

attribute map from SmartThings, we automatically analyze the 334 device handlers from the SmartThings repository [178], and capture the device-attribute relationships evident in the preamble (seen in Figure 4.5).

**2. Trusted Endorsement Attributes for Tamperproof Endorsement:** We observe that similar to AHOs, several device-attributes are modifiable by third-parties through the platform APIs, either for legitimate purposes or due to over-privilege. Therefore, for tamperproof endorsement, HomeEndorser must be able to trust the information received from the participating endorsers *i.e.*, device-attribute pairs (**G3**). We achieve this goal by defining a trusted subset of device-attributes to be used for our checks, *i.e.*, *endorsement attributes*. We propose two categories of endorsement attributes: (1) *read-only attributes*, *i.e.*, which are only writable by devices, and not via API calls, rendering them read-only from the third-party API caller’s perspective (*e.g.*, motion sensor reading), and (2) *designated attributes*, which are writable in theory, but are considered high-integrity by platforms and prior security research [201, 41] alike (*e.g.*, locking the door lock), and hence, heavily restricted. For example, Nest allows its own platform app to unlock the locks integrated with it (*e.g.*, the Nest X Yale Lock [76]), but does not allow third-party apps to do the same. Therefore, both *read-only* and *designated* device attributes would have a *higher integrity level* than an AHO such as `home`, and hence, would be trusted to endorse it.

We identify read-only device-attributes in a manner similar to our approach for extracting the device-attribute map, *i.e.*, by parsing platform documentation and device handler code. For identifying designated attributes, we leverage the rationale of popular platforms for assigning integrity levels to attributes only writable by the platform app, particularly those belonging to security-sensitive devices.

**3. Generating Policy Templates from Inferences:** We now address the question of *how* the endorsement attributes are used to endorse a specific AHO, by designing a holistic *inference study*. This template generation process is a one-time, expert-driven effort.

**Table 4.1:** AHOs inferred from Endorsement Attributes

<b>AHO</b>	<b>Endorsement attributes</b>
home	<security-panel, disarmed>
home	<motion-sensor, active>
fire	<temperature-sensor, temperature>
fire	<smoke-detector, smoke-alarm-state>
safety_state	<co-detector,co-alarm-state>
illuminance	<blind,openLevel>

We begin by identifying 5 additional AHOs, by considering the objects that have been previously examined and found to be susceptible to transitive attacks [51, 112], and by considering AHOs not mentioned in prior work, but which we encountered when building our device-attribute map (*e.g.*, the `safety_state` from Nest, which prior work does not account for). Then, we consider each device-attribute, and identify the type of information sensed or observed by that device-attribute, which we then translate to an inference that could be used for endorsing an integrity-sensitive change in one or more of the AHOs. For example, the device-attribute pair <security-panel, disarmed> indicates that the security panel/keypad was recently disarmed, which may indicate that the user recently arrived home, and hence, provide an inference to endorse the home AHO’s proposed change to “home”. As a key principle in the construction of policies is the a combination of device inferences, we combine inferences to construct deployment-independent policy templates using the structure defined in Section 4.3.1. An example set of inferences are shown in Table 4.1.

## 4.4 HomeEndorser Implementation

This section describes our policy specification study, as well as the reference monitor implemented in HomeAssistant. Our data is available in our online appendix [19].

**1. Policy Specification Study:** We automatically generated a combined device-attribute map from all the data sources consisting of 100 device-types and 510 device-attribute pairs. We found that these pairs utilized 41 endorsement attributes, *i.e.*, read-only or designated device-attributes. Two authors then independently identified the inferences that could

be drawn from these endorsement attributes to endorse changes in one or more of our 6 AHOs. When identifying inferences, the coders disagreed on 12 out of the 510 device-attribute pairs (2.4% disagreement rate), which were resolved through mutual discussion. The inferences led to 10 endorsement attributes for home AHO alone, all of which can be used to instantiate policies by HomeEndorser as we describe in Section 4.3.2. Note that the generation of the device-attribute map was fully automated and directly derived from platform sources, and hence, did not result in any disagreements.

**2. Implementation on HomeAssistant:** We implemented HomeEndorser in HomeAssistant, a popular open-source smart home platform, for two main reasons: (1) the availability of source code, and (2) the centralized execution of automation and state changes. Similar to the *platform state machine* described in Section 4.3.2, HomeAssistant has a state machine component that persistently records all device states. HomeEndorser modifies this state machine to keep track of the most recent state changes and their timestamps. It also hooks into the state machine to intercept the incoming state change requests to mediate all API accesses. Furthermore, HomeAssistant has a component known as an *Event Bus*, which announces and logs the addition/removal of devices, among other events. HomeEndorser adds callbacks in the Event Bus and re-instantiates the endorsement policies in response to any change in device deployment. Finally, HomeEndorser keeps track of device-connectivity using HomeAssistant’s built-in mechanisms, and falls-back to the next most restrictive policy in case a device becomes unavailable at runtime.

## 4.5 HomeEndorser Evaluation

We evaluate the effectiveness and practicality of HomeEndorser through the following research questions:

- **HomeEndorser-RQ<sub>1</sub>:** (*Feasibility of the policy model*) Is it feasible to produce endorsement policies using only a small subset of trusted *endorsement attributes*?
- **HomeEndorser-RQ<sub>2</sub>:** (*Generalizability of the policy model*) Do policies exist for en-

dorsing AHOs other than home?

- **HomeEndorser-RQ<sub>3</sub>**: (*Security*) Does HomeEndorser prevent an attacker from escalating privilege to a high-integrity device (e.g., a camera) using one or more AHOs?
- **HomeEndorser-RQ<sub>4</sub>**: (*False Denials*) What is the rate of false denials in typical benign usage, i.e., when users intentionally cause AHO changes, and over a period of home automation usage?
- **HomeEndorser-RQ<sub>5</sub>**: (*Runtime Performance*) What is the performance overhead introduced by HomeEndorser?
- **HomeEndorser-RQ<sub>6</sub>**: (*Cost*) How much effort is required to integrate and deploy HomeEndorser?

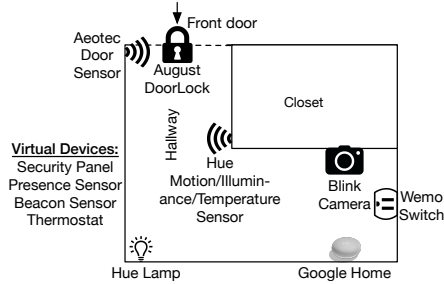
We evaluate **HomeEndorser-RQ<sub>1</sub>→HomeEndorser-RQ<sub>6</sub>** using a diverse array of empirical and experimental methods. We first derive an endorsement policy template for the home AHO 10 endorsement attributes, allowing us to automatically instantiate a maximum of 1023 policies based on device availability, which demonstrates feasibility in several home deployments, even with a limited number of endorsement attributes (**HomeEndorser-RQ<sub>1</sub>**). Further, we demonstrate that HomeEndorser may generally apply to 5 additional AHOs drawn from platform documentation and prior work, as these AHOs may be endorsed using policy templates composed of 5 device attributes on average (**HomeEndorser-RQ<sub>2</sub>**). We then deploy 11 devices (7 real, 4 virtual) in a real home to perform a case study wherein we assume that the attacker’s goal is to disable the security camera by tampering with AHOs that the camera depends on. Our experiments demonstrate that HomeEndorser effectively endorses the two AHOs that can be tampered with to disable the camera (**HomeEndorser-RQ<sub>3</sub>**). We perform 10 additional *realistic* scenarios obtained from (or inspired by) prior work [111] that represent various ways in which the user would intentionally change AHOs, and demonstrate that HomeEndorser correctly allows all of the intentional AHO changes, resulting in no false denials (**HomeEndorser-RQ<sub>4</sub>**). Furthermore, we evaluate HomeEndorser’s performance with

400 realistic smart home usage sequences (consisting of 8191 events in total) obtained from Helion [124], which results in 605 endorsement checks (all for benign execution) that HomeEndorser endorses correctly, with no false positives. Further, we demonstrate practical performance overheads using macro and microbenchmarks (**HomeEndorser-RQ<sub>5</sub>**). Finally, we demonstrate the minimal *costs* of deploying HomeEndorser, by (i) quantifying the effort required by experts to generate policy templates, (ii) demonstrating how HomeEndorser simplifies deployment by automatically instantiating policies without imposing any burden on the user, and (iii) distilling key properties that a platform would need in order to integrate HomeEndorser, qualitatively analyzing 4 popular smart home platforms, and demonstrating the feasibility of integrating HomeEndorser in them with *modest engineering effort* (**HomeEndorser-RQ<sub>6</sub>**).

**Table 4.2:** Real and Virtual Devices in Evaluation

Device	real/virtual	Number
August Door Lock	real	1
Blink Camera	real	1
Philips Hue Motion+Illuminance+Temp. Sensor	real	1
Aotec Door Sensor	real	1
Security Panel	virtual	1
Presence Sensor	virtual	1
Beacon Sensor	virtual	1
Thermostat	virtual	1
Wemo Switch	real	1
Philips Hue Lamp	real	1
Google Home	real	1

**Experimental Setup:** We integrated HomeEndorser into HomeAssistant (v0.112.0.dev0), running on a Macbook Pro machine with 16GB of RAM. Further, we connected 7 smart home devices to our smart home, and created 4 other *virtual* devices (see Table 4.2 for the full list). Our choice of devices was limited by compatibility with HomeAssistant, with a bias towards known/popular brands and device types that would allow us to evaluate HomeEndorser’s endorsement. We installed these devices in a room as shown in Figure 4.6, influenced by effective deployment recommendations from prior work [111], *e.g.*, placing the August door lock and the Aotec door sensor on the main door, and the motion



**Figure 4.6:** Layout of the physical device placement

sensor in the hallway directly past the door, and the Blink camera placed inside the room. We interacted with the real devices physically and with the virtual devices through the HomeAssistant UI.

**Table 4.3:** Sample policies for endorsing home (“away”→“home”)

	Policy
P <sub>1</sub>	<security-panel, disarmed> ∧ <motion-sensor, active>
P <sub>2</sub>	<Doorlock, unlocked> ∧ <presence-sensor, active> ∧ <beacon, active>
P <sub>3</sub>	<Garage-doorlock,unlocked> ∧ <beacon,active>

#### 4.5.1 Feasibility of the Policy Model (HomeEndorser-RQ<sub>1</sub>)

We do not expect users to possess all possible device-combinations that endorse a specific AHO. Instead, our approach enables the instantiation of a large and diverse set of candidate policies to increase the possibility of finding *a* policy that contains the limited set of devices the user possesses.

Using the approach specified in Section 4.3.3, we identified 10 endorsement attributes, for endorsing the home AHO (*i.e.*, the change from “away” to “home”), which could lead to 1023 potential policies. However, since HomeEndorser enforces the *most restrictive but feasible* policy dynamically by instantiating the template in the context of the home, it can automatically adjust to cases where any subset of the 10 endorsement attributes are present and enforce the most restrictive policy for that particular subset. For example, as shown in Table 4.3, a user who has a door lock and motion sensor, or another who has a security panel, or another who has a garage doorlock and a presence sensor, would all be able to endorse home using HomeEndorser (our online appendix [19] provides the full

list of potential policies). This demonstrates that our approach is feasible, *i.e.*, we can define a large number of diverse policies for an AHO (*i.e.*, home), using a limited set of endorsement attributes, and hence, support a diverse array of device-deployment scenarios (**HomeEndorser-RQ<sub>1</sub>**).

#### 4.5.2 Generalizability of the Policy Model (**HomeEndorser-RQ<sub>2</sub>**)

To demonstrate the generalizability of our policy model, we consider 5 additional AHOs (*i.e.*, security\_state, fire, water leak, illuminance, and safety\_state). For each AHO, we identified endorsement attributes from the device-attribute map and generated inferences using the process from Section 4.3.3. Our process resulted in 41 inferences (cumulatively) that would be useful for endorsement, with each AHOs being endorsed using *at least* 3 device-attributes (examples provided in Table 4.1 in Section 4.3.3). This demonstrates that our approach is generalizable to AHOs other than home, *i.e.*, similar policies are feasible for 5 other AHOs (**HomeEndorser-RQ<sub>2</sub>**).

#### 4.5.3 Case Study: Protecting the Security Camera from Privilege Escalation (**HomeEndorser-RQ<sub>3</sub>**)

Given the attacker’s goal to tamper with a high-integrity device that they cannot directly access (e.g., a security camera), HomeEndorser’s endorsement checks prevent the attacker from affecting malicious changes to *any* AHO that the aforementioned device depends on. Thus, we assume the threat described in the motivating example (Sec. 4.1), where the camera depends on both the home and security\_state AHOs, and experimentally demonstrate HomeEndorser’s effectiveness using two attack scenarios.

**Malicious Scenario 1 – Bob modifies home:** We deploy a malicious third-party service controlled by the attacker, Bob. We assume that Alice has granted to the service the permission (*i.e.*, a REST API token) to write to the home AHO. When Alice is out of the home, Bob writes to the value “home” to home, to disable the camera. Without

HomeEndorser, the home AHO will change, allowing Bob to remotely disable the security camera; however, we consider that Alice uses HomeEndorser with the policy  $P_{\text{home}_1}(\text{home})$ :

$$P_{\text{home}_1}(\text{home}) = (\text{door-lock\_lock} == \text{UNLOCKED} \wedge \\ \text{motion\_sensor} == \text{ACTIVE})_{\text{front-door}}$$

Thus, when Bob writes to `home`, the policy  $P_{\text{home}_1}(\text{home})$  is checked as follows: HomeEndorser queries the state machine, and obtains the most recent change to the door lock as well as the entryway motion detector. Since the door lock was not unlocked, and the motion detector has also not detected motion recently, the policy returns a DENY decision, preventing the attack. It is also important to note that Bob could attempt to circumvent HomeEndorser’s policy by satisfying one of the two conditions in it, *e.g.*, by sliding a thin object (*e.g.*, a card) through the door to trigger the motion sensor; however, the conjunction among device-attributes prevents this attack.

**Malicious Scenario 2 – Bob modifies `security_state`:** We deploy a malicious third-party service controlled by Bob, to which Alice has granted the permission to write to the `security_state` AHO. Bob will attempt to set the `security_state` to “ok” (as opposed to “deter”), which will trigger a routine that turns off the camera. Without HomeEndorser, the `security_state` state will successfully change, allowing Bob to disable the camera; however, we consider that Alice uses HomeEndorser with the policy  $P_{\text{security\_state}_1}(\text{ok})$ :

$$P_{\text{security\_state}_1}(\text{ok}) = (\text{security-panel} == \text{DISARMED} \wedge \\ \text{motion\_sensor} == \text{ACTIVE})_{\text{front-door}}$$

When Bob writes to `security_state`,  $P_{\text{security\_state}_1}(\text{ok})$  is checked. Since the security panel was not disarmed and the motion sensor was not active recently, the policy returns a DENY decision, preventing the attack. Thus, HomeEndorser successfully endorses both of the AHOs on the transitive attack path to the security camera.

#### 4.5.4 Intentional AHO Changes and Normal Usage (HomeEndorser-RQ<sub>4</sub>)

Users may themselves choose to change AHOs (e.g., by manually selecting home mode in their platform mobile app after getting home), or by using a third-party service that automatically requests appropriate state changes based on the user’s location (e.g., when the user arrives home). We derive a set of 10 *realistic user behavior scenarios* from prior work [111] to use as our benign examples, and enact those scenarios in our apartment testbed. We summarize 3/10 scenarios that are exemplary of HomeEndorser’s decisions in response to benign user behavior, with the rest available in Table 4.6.

**Scenario 1 – Unlocking the house, and then leaving:** Alice returns home and opens the front door after unlocking the front door lock. However, she gets a call from her office and leaves immediately without entering, accidentally also leaving the door open in the process. Regardless, Alice’s home/away service accidentally requests the home AHO to change to “home” (i.e., even when Alice has actually left), which would disable the camera and leave the home unmonitored. In response to the request, HomeEndorser gathers the recent states of the devices to check against the policy  $P_{\text{home}_2}(\text{home})$ .

$$P_{\text{home}_2}(\text{home}) = (\text{door-lock\_lock} == \text{UNLOCKED} \wedge \text{motion\_sensor} == \text{ACTIVE} \wedge \text{door\_sensor} == \text{ACTIVE})_{\text{front-door}}$$

The policy constraints are *partially* satisfied, as the door lock was recently manually unlocked, and the door sensor’s state changed to “open”. However, as Alice did not enter, the motion sensor did not detect any motion, and the policy results in a *correct denial*, preventing an unsafe situation in which the camera is turned off while the home is vulnerable (i.e., the door is unlocked). That is, HomeEndorser’s composite policy design leverages additional devices such as the motion detector to provide *stronger* endorsement, and hence, is useful for preventing accidental but unsafe changes.

**Scenario 2 – Disarming the security panel and entering:** In this scenario, Alice returns home and inputs the key-code in the security panel near the door, disarming the

home. She then enters the home triggering the motion sensor. At the same time, a home security service requests change to the `security_state` AHO on Alice’s behalf, from “deter” to “ok”, which if allowed, would disable the security camera, as well as any other security devices (*e.g.*, alarms).

HomeEndorser gathers the recent states of the devices to check against the policy  $P_{\text{security\_state}_1}(\text{ok})$  (provided previously in Section 4.5.3). Since the security panel was manually disarmed and the motion sensor was active recently as well, the policy is satisfied and the state change is correctly allowed.

**Scenario 3 – Direct state change request:** Alice returns, and manually changes the home AHO to “home” using the HomeAssistant UI. HomeEndorser identifies that the request was not made through the REST API, and allows it without checking the policy, honoring the user’s explicit intent.

In addition to home usage scenarios provided by prior work [111], we also evaluate HomeEndorser’s performance under realistic home *automation* usage, as generated by Helion [124], a recent system that learns the naturalness of home automation and generates large realistic home automation event sequences that are more representative of smart home usage than the random events used in prior work [110].

**Performance under *realistic home automation usage*:** For this experiment, we executed a set of 400 unique event sequences provided by Helion [90] consisting of 8191 total events and 64 unique devices on our testbed (with HomeEndorser), which required us to create 51 virtual devices in addition to 13 that were already present. To assess the accuracy of the endorsement, we save a snapshot of all the device states at the time HomeEndorser makes a decision and automatically compare all the snapshots against the policy in effect. Upon executing these event sequences with HomeEndorser, we observed that HomeEndorser was invoked in 605 home AHO state changes, and made the correct decision in all of them, *i.e.*, correctly allowing AHO state change in 562 cases and correctly denying it in 43 cases.

**Backwards Compatibility:** One of our key objectives is to build a practical framework that does not crash or in any other way impede legitimate functionality. Therefore, we evaluate the backwards compatibility of HomeEndorser with automation developed by real users or app developers (*i.e.*, IoT apps or user-driven routines). For this, we randomly selected 20 automations from the SmartThings repository [178]. We then leveraged our existing deployment of HomeEndorser and set up 10 of these automations in HomeAssistant that were applicable to the devices in our deployment, filtering out the remaining 10 for which we did not have devices. We then triggered the automations throughout the day, in a random order and at random times, while constantly monitoring for system crashes, dropped or delayed device actions, or any other unusual device, automation, or platform failures. Note that since this is an automated experiment, certain automated triggers may be denied due to endorsement checks; we consider these denials as legitimate, and not failure cases. Throughout the whole observation period, we did not observe any crashes or device failures. Note that the selection of routines included one involving home shared state changing from *away* to *home* that was denied throughout the day, *i.e.*, *when user is home, disarm the security panel*. On the other hand, routines that were changing states not endorsed by HomeEndorser were allowed throughout the day.

To summarize, our evaluation demonstrates that HomeEndorser does not cause false denials under benign behavior. In fact, as shown previously in scenario 1, its denials prevent *accidental* and harmful state changes by users (*i.e.*, and hence, are correct) (**HomeEndorser-RQ<sub>4</sub>**). We further observed that in several cases, by the time the endorsement was requested (*i.e.*, time of REST API call), the relevant devices (*e.g.*, motion sensors) had reverted to their default states (*e.g.*, the motion sensor reverted to the “inactive” state). This means that if we had only checked the *current* state of the devices, *i.e.*, the state at the time of endorsement, the endorsement check would have resulted in a *false denial*. However, HomeEndorser’s approach of checking the *most recent* change prevents such potential false denials. Finally, we did not observe any device or automation crashes/failures in the home in response to HomeEndorser’s endorsement, whether executing with Helion’s

scenarios or otherwise.

**Table 4.4:** Performance overhead of HomeEndorser (HE) (in comparison with the unmodified HomeAssistant baseline)

No.	Operation	HE Baseline (ms)	HE (ms)	Overhead(ms)	Overhead(%)
1.	Policy Instantiation ( <i>Boot up time</i> )	23.851 ± 1.738	33.669 ± 5.042	9.818	41.16
2.	Policy update during runtime	-	4.350 ± 0.515	-	-
3.	Changing non-endorsed AHO ( <i>Hook invocation cost</i> )	9.854 ± 0.723	9.916 ± 0.814	0.062	0.63
4.	Changing endorsed AHO ( <i>Endorsement check cost</i> )	9.451 ± 0.605	10.367 ± 0.482	0.916	9.69
5.	Automation execution with endorsed AHO	16.582 ± 2.388	18.598 ± 0.669	2.016	12.16
6.	Automation execution with non-endorsed AHO	14.609 ± 1.026	14.311 ± 0.477	-0.298	-2.04

#### 4.5.5 Runtime Performance (HomeEndorser-RQ<sub>5</sub>)

We compute microbenchmarks to capture each aspect of the platform that HomeEndorser affects, in particular, the time taken for (1) *policy instantiation* (*i.e.*, additional delay at boot time), (2) *policy update* during runtime (*i.e.*, overhead of device addition/removal) (3) the *endorsement hook invocation cost* (*i.e.*, overhead of an API call to a state *not being endorsed*), and, (4) the *endorsement check* (*i.e.*, overhead of an API call to a state being endorsed). Furthermore, we perform 2 macrobenchmarks to assess the overall end-to-end impact of HomeEndorser on the execution times of remote IoT services that leverage the REST API for (5) executing an automation involving an AHO being endorsed, and (6) executing an automation involving an AHO not being endorsed. We measure the worst-case performance by performing all measurements with the largest policy, *i.e.*, the policy involving the maximum number (*i.e.*, 7) of device-attributes that can be instantiated in our device deployment (*i.e.*, P<sub>109</sub>, online appendix [19]). We compute baselines for the benchmarks using an unmodified build of HomeAssistant with the same devices, and perform each experiment 50 times in both environments. Table 4.4 shows the mean results with 95% confidence intervals.

**Results:** As seen in Table 4.4, HomeEndorser has negligible performance overhead for

operations that do not involve the AHO being endorsed, *i.e.*, the hook invocation cost when calling an API that modifies a non-endorsed AHO (microbenchmark), as well as executing an automation that changes a non-endorsed AHO (macrobenchmark), with the overhead falling within the error margin.

For endorsed AHOs, HomeEndorser adds only 0.916ms (9.69% overhead) to an AHO-change invoked via an API call (microbenchmark), and adds 2.016ms (12.16% overhead) to the overall execution time of an automation execution that changes an endorsed AHO (macrobenchmark). In fact, the maximum overhead of 9.818ms (41.16%) that HomeEndorser adds is to the overall bootup time of HomeAssistant, which is not that frequent, and not perceivable by the user. This is due to the fact HomeEndorser’s policy instantiator is called repeatedly as HomeAssistant adds devices during bootup. After the bootup, the overhead to update policies when devices get added or removed is only 4.350 ms. Finally, we note that the endorsement check overhead is not dependent on the policy size, as HomeAssistant’s (and hence HomeEndorser’s) state machine obtains device state changes in parallel.

#### 4.5.6 Effort Required to Integrate and Configure HomeEndorser (HomeEndorser-RQ<sub>6</sub>)

This section describes the effort required to deploy and integrate HomeEndorser. Particularly, we demonstrate that (1) experts can generate policy templates for HomeEndorser using the systematic semi-automated methodology defined in Section 4.3.3 with minimal effort and time, (2) HomeEndorser can *automatically* instantiate its policies in end-user homes as per the availability and placement of devices, which significantly reduces configuration effort, and finally (3) most platforms may directly integrate HomeEndorser without any design-level extensions, which indicates its practicality (HomeEndorser-RQ<sub>6</sub>).

**1. Effort by experts:** HomeEndorser’s semi-automated, data-driven, process for generating policy templates is a one-time effort, as described in Section 4.3.3, and templates only need to be updated when new functionality emerges in an entire category of devices

such as door locks (*i.e.*, and not individual brands), or when an entirely new category of device is introduced to the market. The only manual effort involves the identification of the endorsement attributes from attribute inferences (as described in Section 4.3.3). For instance, it took 2 authors around 4 workdays to identify the 10 endorsement attributes for the home AHO, from the 510 device-attributes automatically identified by our approach.

**3. Platform integration:** Through lessons learned during our implementation of HomeEndorser, we distill 4 key platform-properties that are necessary to successfully integrate HomeEndorser in a smart home platform, in a way that satisfies the general purpose of physical home endorsement, as well as certain enforcement-related design goals (**G2**→**G4**). We then discuss these identified properties in the context of 4 popular platforms, namely IoTivity [107], OpenHAB [153], SmartThings [193], and Nest/Google Assistant [77], and estimate the effort that would be required for the wide-spread integration of physical home endorsement, across all platforms.

Property 1 (Prop<sub>1</sub>) - Ability to obtain device states: In order for HomeEndorser to correctly enforce the policies, it has to be able to obtain states from all devices involved in the policies. Ideally, the platform should have a Platform State Machine that can readily provide device state information.

Property 2 (Prop<sub>2</sub>) - Complete mediation of state changes: For HomeEndorser to intercept all incoming API requests that seek to change AHOs, and trigger endorsement checks (**G2**), the platform must have a central component that intercepts all the API requests. Furthermore, this component must be unmodifiable by third parties (**G3**).

Property 3 (Prop<sub>3</sub>) - Timestamp information of device states: HomeEndorser requires *recent* device state information to prevent any false positives that can occur because of devices reporting cached states or the platform itself reporting the and old/last known state because of an unresponsive device (**G4**).

Property 4 (Prop<sub>4</sub>) - Ability to monitor device changes: HomeEndorser needs the ability

to dynamically adapt its policies based on the current setup of the smart home, and hence, the platform needs to monitor the addition, removal, and change in placement of devices.

**Table 4.5:** The (minimal) cost of Integrating HomeEndorser with respect to the properties identified in Section 4.5.6

	HomeAssistant	IoTivity	OpenHAB	SmartThings	Nest
Prop <sub>0</sub>	✓	✓	✓	✓*	✓*
Prop <sub>1</sub>	✓	✓	✓	✓*	✓*
Prop <sub>2</sub>	✓	✓	✓	✓	✓*
Prop <sub>3</sub>	✓	×	✓	✓	✓*
Prop <sub>4</sub>	✓	✓	✓	✓*	✓*

✓ = Directly portable, ✓\* = Directly portable, but needs confirmation from source code, × = design-level constraint/extension

Table 4.5 shows the results of our platform analysis based on Prop<sub>1</sub> → Prop<sub>4</sub>, and particularly, demonstrates that only IoTivity requires a design-level extension for integrating HomeEndorser (in terms of Prop<sub>3</sub>), and *all other platforms may feasibly integrate HomeEndorser*, sometimes with negligible engineering efforts. Each integration may require minor adjustments, *e.g.*, for HomeAssistant, we modified the state machine for complete mediation, whereas in OpenHAB, we would need to implement a reference monitor across various bindings (*i.e.*, hook into services exposed as bindings), in a manner similar to how extensible access control has been previously implemented in OSes such as Linux [225] and Android [91]. While IoTivity provisions states to the reference monitor (Prop<sub>2</sub>), a state machine with the ability to collect freshness information will need to be implemented (Prop<sub>3</sub>), which would be a design-level extension. Finally, we mark certain properties for Nest and SmartThings as ✓\* in Table 4.5 as we can almost certainly assert that they can integrate HomeEndorser with minimal engineering efforts, we would need source code to confirm.

## 4.6 Threats to Validity of this Chapter

With HomeEndorser’s systematic and data-driven approach, our work lays the groundwork for secure and practical endorsement for AHOs in the smart home. The threats to the

validity of our approach and results are discussed below:

**1. Byzantine Fault Tolerance:** We rely on devices to not be compromised and to report correct states, as stated in the threat model in Section 4.3 (although HomeEndorser does dynamically adapt to devices that may be offline/non-responsive). That said, HomeEndorser’s integrity validation of AHOs complements prior efforts [26] that aim to validate device states via fingerprinting, and combining the two approaches is a promising future direction.

**2. Completeness and Rigor of Policy Generation:** The device-attribute map used for identifying endorsement attributes was automatically constructed from platform-provided resources (Section 4.3.3). This map, consisting of 510 device-attribute-pairs, is an *evolving dataset* that is *as complete as the information sources* used to derive it (*e.g.*, device handlers, capability maps), and new device attributes/types that emerge may be integrated into it with minimal effort. Furthermore, we used systematic, 2-author, open coding to identify the endorsement attributes from this map that would enable endorsement of each of the 6 AHOs (Section 4.4). This approach resulted in negligible disagreements (*i.e.*, only 2.4%), illustrating high confidence in the identified inferences.

**3. Multi-user smart homes:** Similar to most prior work in the area of smart home API misuse [110, 171, 119, 217], HomeEndorser does not claim to address multi-user scenarios, which are a novel but orthogonal design challenge. Further, we note that certain AHOs (*e.g.*, fire) may be independent of the number of users, and moreover, HomeEndorser’s location-specific policy-predicates (Section 4.3.1) may also mitigate the implications of multiple simultaneous device-interactions.

**4. Device Availability and Placement:** HomeEndorser’s policies consist of a diverse range of devices. Hence, the user does not need to have all the devices, as HomeEndorser can automatically choose the most restrictive policy that is applicable to a user’s home based on device-availability and placement information obtained from the platform (Section 4.3.2). However, we assume optimal device placement and sensor configuration to be out of scope,

and direct the reader to complementary work that informs on optimal deployment [111].

## 4.7 Chapter Summary

This chapter presents the HomeEndorser framework, which validates proposed changes in high integrity abstract home objects (AHOs) in correlation with device states. HomeEndorser aims to address the threat posed by compromised/malicious services that may use their authorized API access to change AHOs in a way that causes high-integrity devices to behave unsafely. To do this, HomeEndorser leverages the insight that changes in AHOs are inherently tied to a home’s physical environment, enabling each sensitive change to be *endorsed* by correlating to changes in physical device states. HomeEndorser provides a policy model for specifying endorsement policies in terms of device state changes, and a platform reference monitor for endorsing all API requests to change AHOs. We evaluate HomeEndorser on the HomeAssistant platform, finding that we can feasibly derive policy rules for HomeEndorser to endorse changes to 6 key AHOs, preventing malice and accidents with feasible performance overhead. Finally, we demonstrate that HomeEndorser is backwards compatible with most popular smart home platforms, and requires modest human effort to configure and deploy.

**Table 4.6:** Realistic Scenarios used in evaluation of HomeEndorser

Scenario	User Behavior	Incoming request	Policy involved	HomeEndorser behavior
S4 : Entering the home	Alice returns home, unlocks the front door lock and opens the door. She then closes the door and enters, triggering the motion sensor near the door. Around the same time, her home/away tracker service requests a state change from away to home using REST API.	An app requests a state change from away to home using REST API.	$P_{\text{home}_2} = (\text{door\_lock\_lock} == \text{UNLOCKED} \wedge \text{motion\_sensor} == \overline{\text{ACTIVE}} \wedge \text{door\_sensor} == \text{ACTIVE})_{\text{front-door}}$	HomeEndorser gathers the recent states of the devices. Since the door lock was manually unlocked and the sensors were active recently, the policy is satisfied and HomeEndorser allows the change, as Alice intended.
S5 : A home without a smart door lock	Alice does not possess a smart door lock, but has a presence sensor and a motion sensor. She enters the home, triggering the sensors in the hallway.	An app requests a state change from away to home using REST API.	$P_{\text{home}_3}(\text{home}) = (\text{presence\_sensor} == \text{ACTIVE} \wedge \text{motion\_sensor} == \overline{\text{ACTIVE}})_{\text{front-door}}$	Since only the sensors are present, HomeEndorser gathers the recent states of only the sensors. As they were recently active, the policy is satisfied and the state change is correctly allowed.
S6 : Entering the house without closing the door	Alice comes home and opens the front door after unlocking the front door lock manually. She leaves the door open, and comes in by triggering the motion sensor in the hallway.	An app requests a state change from away to home using REST API.	$P_{\text{home}_2} = (\text{door\_lock\_lock} == \text{UNLOCKED} \wedge \text{motion\_sensor} == \overline{\text{ACTIVE}} \wedge \text{door\_sensor} == \text{ACTIVE})_{\text{front-door}}$	HomeEndorser gathers the recent states of the devices. Since the door-lock was manually unlocked and the sensors were active recently, the policy is satisfied and the state change is <i>endorsed</i> .
S7 : Unlocking the home and leaving	Alice comes home and opens the front door after unlocking the front door lock manually. She steps outside immediately after entering and closes the door behind her, thereby triggering the door sensor again.	An app requests a state change from away to home using REST API.	same as $P_{\text{home}_2}$	HomeEndorser gathers the recent states of the devices. The door lock was manually unlocked and the door sensor was triggered both when opening and closing the door. However, the motion sensor was not active recently so the policy is not satisfied and the state change is <i>denied</i> .
S8 : Diverse device setup1	Alice comes home and opens the front door after unlocking the front door lock manually. She comes in by triggering the motion sensor and the beacon sensor in the hallway.	An app requests a state change from away to home using REST API.	$P_{\text{home}_3} = (\text{door\_lock\_lock} == \text{UNLOCKED} \wedge \text{motion\_sensor} == \overline{\text{ACTIVE}} \wedge \text{door\_sensor} == \text{ACTIVE} \wedge \text{beacon\_sensor} == \text{ACTIVE})_{\text{front-door}}$	HomeEndorser gathers the recent states of the devices. Since the door-lock was manually unlocked and the sensors were active recently, the policy is satisfied and the state change is <i>endorsed</i> .
S9 : Diverse device setup2	Alice comes home and inputs the keycode in the security panel near the door, disarming the home. She then enters the home triggering the motion sensor in the hallway.	An app requests a state change from away to home using REST API.	$P_{\text{home}_4} = (\text{security\_panel} == \text{DISARMED} \wedge \text{motion\_sensor} == \overline{\text{ACTIVE}} \wedge \text{presence\_sensor} == \text{ACTIVE})_{\text{front-door}}$	HomeEndorser gathers the recent states of the devices. Since the security panel was manually disarmed and the motion sensor was active recently as well, the policy is satisfied and the state change is <i>endorsed</i> .
S10 : Home with just 2 devices	Alice comes home and unlocks the front door lock manually. She comes in by triggering the motion sensor in the hallway.	An app requests a state change from away to home using REST API.	$P_{\text{home}_5} = (\text{door\_lock\_lock} == \text{UNLOCKED} \wedge \text{motion\_sensor} == \text{ACTIVE})_{\text{front-door}}$	HomeEndorser gathers the recent states of the devices. Since the door lock was manually unlocked and the motion sensor was active recently, the policy is satisfied and the state change is <i>endorsed</i> .

## Chapter 5

# Polityzer: Understanding the Privacy Practices of Political Campaigns

Political campaigns are increasingly relying on their online presence, *i.e.*, social media, campaign websites and mobile apps, to engage with potential voters. Campaigns have been observed to leverage their web presence as one of the primary means of gathering information on voters, which is often combined with publicly and commercial sources to create accurate profiles of individual voters [173, 8]. Such information is often personal, *e.g.*, email, phone number, and salary, and highly private in some cases, *e.g.*, citizenship, partner’s name and contact information, with serious privacy implications [98]. To our knowledge, while the use and impact of social media on election campaigns has been previously studied [43][72][123][186], the privacy posture of *campaign websites* is yet unexplored at a large scale.

The privacy practices of campaign websites must be systematically studied for four reasons. First, political campaigns are (at least in the U.S.) generally classed as “nonprofit organizations”, and hence, data privacy regulations such as California Privacy Rights Act (CPRA) do not apply to them [4]. This gap in regulation may mean a lack of incentive in following privacy best practices. Second, while U.S. political campaigns are required by the Federal Election Commission (FEC) to collect donor names, mailing addresses,

occupation, and employer [62], they may collect significant additional private data. Third, prior work shows that campaigns often share data [163], and deploy aggressive tactics to get users to submit information [208] or interact with their political emails [126]. Fourth, campaign websites are ephemeral in nature, and hence, it is unclear what happens to user data after the election. Thus, the user may lose any agency over their data to prevent future misuse, in which case, transparent disclosure of collection and sharing practices by the website is the only recourse for users. These factors, along with the increased transparency users desire regarding the use of their political data [188], and the governmental interest in regulating this space [74, 57, 103, 215], motivate us to empirically understand the privacy posture of campaign websites.

This chapter describes Polityzer, a semi-automatic framework for a systematic, large-scale analysis of the privacy practices of political campaign websites. The design of Polityzer leverages the fact that political campaigns generally interact with potential voters, volunteers, and donors through the *campaign websites*, and thus the privacy implications of political campaigns can be approximated through a comprehensive analysis of campaign websites. We use Polityzer to analyze the websites of 2060 campaigns established for the 2020 US Presidential, Senate, and House elections, to answer four fundamental *research questions (RQs)*:

**Polityzer-RQ<sub>1</sub> (*Collection*)** – What data do campaigns collect from their websites?

**Polityzer-RQ<sub>2</sub> (*Disclosure*)** – Do campaigns properly disclose the collection, sharing and retention of this data to users?

**Polityzer-RQ<sub>3</sub> (*Conflict*)** – Does the collection and sharing of campaign data conflict with their privacy disclosures?

**Polityzer-RQ<sub>4</sub> (*Risk*)** – Do campaign websites expose users to privacy or security risks such as malware or trackers?

Polityzer addresses these questions through a semi-automated methodology that combines text and website analysis: First, it extracts unique types of privacy-sensitive data col-

lected by election campaigns, through an analysis of forms contained in campaign websites (**Polityzer-RQ<sub>1</sub>**). Second, it compares the collected data types against the campaign website’s privacy policy to assess whether campaigns properly disclose the collection to users (**Polityzer-RQ<sub>2</sub>**). Third, it facilitates a study to measure the conflicts in a campaign’s privacy disclosure (**Polityzer-RQ<sub>3</sub>**) by examining the privacy policies of campaigns and fundraising platforms to understand potential/indirect collection (and sharing) not disclosed in a campaign’s policy, but may occur due to the use of the fundraising platform. Finally, it leverages popular security tools (VirusTotal[13], ApiVoid[12]) to assess the general security and privacy-hygiene of campaign websites in terms of malware, hosting, and SSL/TLS misuse (**Polityzer-RQ<sub>4</sub>**). The contributions of this chapter are summarized as follows:

- **Polityzer:** We design and implement Polityzer to enable large-scale analysis of the privacy practices of political campaign websites. Polityzer is highly precise in terms of identifying campaign sites without privacy policies, with a false positive rate of 1.29%.
- **Study:** We use Polityzer to perform the *first large-scale analysis of the privacy practices of campaign websites*, analyzing 2060 sites of House, Senate, and Presidential candidates from the 2020 U.S. election.
- **Findings:** Our analysis leads to *20 key findings* that demonstrate significant privacy gaps. For instance, we find that 70.78% campaigns do not provide privacy disclosures, of which 64.27% that collect sensitive data. Similarly, even where privacy policies are present, 41.22% campaigns do not properly disclose data collection. Moreover, we find that 144/162 (88.89%) campaigns among those with privacy policies may be *inadvertently* (and without disclosure) sharing data with other campaigns through the use of the common fundraising platforms. We also find security weaknesses and use of trackers in campaigns that collect user data. These findings echo prior concerns regarding the privacy practices of political campaigns [99, 54], and demonstrate how websites are indeed used by campaigns to collect private data at scale, but without transparency and accountability.

- **Dataset:** To enable future research, we curate a dataset of 2060 campaign websites and 507 privacy policies belonging to senate, house, incumbents, and presidential candidates. Our artifact is available in our online appendix [165].

## 5.1 Motivation

Political campaigns collect user data from three key sources: publicly available information (*e.g.*, voter rolls), commercial sources (*i.e.*, data brokers or other campaigns), and their campaign websites and apps. Campaigns do not consider one source more important than the other, but instead, aggregate data collected from all sources to form complete profiles on individual voters [173, 8]. Websites, in particular, are critical for three reasons. First, websites enable campaigns to scale their data collection beyond what door-to-door campaigning allows. More importantly, websites provide campaigns with “organic traffic”, *i.e.*, people who naturally navigate to the site or find it via search, and are hence much more likely to donate, volunteer, or register and provide email and other information [197]. As we see later in Section 5.4.2, this importance is evident from the fact that 40.91% of the campaigns registered to the FEC in 2020, including almost all of the winning campaigns, had active websites at the time of the election, many of which collected data outside of what the FEC mandates.

Second, websites also enable campaigns to reliably fill in the gaps in their voter profile databases obtained from other sources. For instance, voters have expressed the desire and ability to opt out from providing their contact information on voter rolls to prevent spam messages [31]; however, campaign websites and apps have been observed collecting the contact information of voters’ friends and social connections [203], thereby completing profiles potentially against the voters’ wishes. We see similar cases of data collection (*e.g.*, partner’s name and email, friends’ names and email) in Section 5.5.2. Finally, *after elections*, it is a standard practice for candidates to rent out or sell their databases to other candidates, PACs, political parties, or private brokers [167], potentially to recuperate

campaign expenditure [203]. That is, campaign websites form an integral data collection vantage point for candidates, helping them collect, aggregate and monetize the data of citizens, in a manner that is quite similar to for-profit social networks, or other commercial websites that face far more scrutiny.

Therefore, given the critical position of campaign websites in a candidate’s data aggregation apparatus, this chapter seeks to empirically understand their privacy posture, further motivated by the increasing interest from governments, evidence of user concern over collection of sensitive political data, and the harm that may befall users if researchers overlook campaign websites, as this section describes.

### 5.1.1 Expectations of Governments and Regulators

Governments, at least in Europe, are cognizant of the privacy risk from data collection by campaign websites, and hold them to the same privacy standards as for-profit organizations. To elaborate, the European Union has taken the lead in protecting the privacy of political data collected by (campaign or other) websites, by classifying “political data” as a special, *opt-in*, category of personal data under the GDPR [74], *i.e.*, which cannot be processed without the owner’s explicit consent. The GDPR also requires election campaigns to inform the users about collected data and the purpose behind the collection, and also to hold the collected data securely [57]. Individual countries in Europe have also issued specific regulatory guidance for political campaign websites, *e.g.*, the UK’s guidance for processing personal data for political campaigning purposes [103] in compliance with both GDPR and the UK’s Data Protection Act [207].

In contrast, the United States does not have a regulation that specifically governs private data collection by political campaigns. Instead, the U.S. has so far specified bare-minimum expectations that prevent the government (including members of the U.S. Congress) from misusing private voter data, such as the “Franking Privilege”, which prevents members of congress from using such data for election campaigns [42]. However, *there are signs that this status quo is changing*, potentially due to the significant push for

consumer data privacy around the globe, and calls by privacy advocates for presidential candidates’ websites to be held to a higher standard than for-profits [203].

To elaborate, the U.S. Congress is currently considering the *Voter Privacy Act* [215] which seeks to grant voters access to personal data that campaigns have on them, and rights to erase their data from campaign databases and prohibit targeted ads. This act targets any political candidate, campaign, or entity using an “interactive computer service” for data collection, *i.e.*, a campaign website or mobile app. Among other things, this proposed act mandates campaigns to disclose the categories of personal information collected on an individual, thus significantly strengthening the transparency around campaign data practices. Such emerging regulations and existing precedents from the EU motivate our data-driven analysis of political campaign websites.

### 5.1.2 User Expectations and Desire for Privacy

There is strong evidence that users are increasingly concerned about data collected for political purposes, *e.g.*, a survey following the Cambridge Analytica scandal found that 73.9% of users were concerned about websites using their data for political purposes [188]. However, to our knowledge, there is no prior work that systematically studies what users precisely expect from political campaigns in terms of digital data privacy. To better understand user expectations in this context, we build upon prior work in Web privacy.

To elaborate, prior work [161] shows that in the general context of data collected by websites, *uninformed users* have no expectations of online privacy. However, *users desire strong privacy guarantees once they are exposed to privacy policies*, and informed on the ways in which their data is collected and used [161]. This shift in behavior is particularly evident in the for-profit context, wherein privacy studies [170, 168, 221] and pertinent regulation [73, 3] have caused user-awareness and privacy expectations to mature over the last decade. For example, a 2021 survey [36] showed that 86% users cared about data privacy, with 79% willing to act to protect their privacy and 47% already switching companies over data privacy practices.

We anticipate that the case of political data is no different, *i.e.*, *users only care when informed*. That is, even in the early stages of the use of Web data in political campaigning, *users expressed their discomfort when explicitly asked about profiling* and data aggregation by campaigns, with one noting that “I would not know any person who would be okay with some outside group having access to that much personal data” [169]. Thus, even if user expectations are unclear at present, it would be premature to conflate the lack of awareness with a lack of desire for data privacy in the political context, given ample evidence about how strongly the public feels about data privacy in general [36, 188, 53, 85, 29, 155]. Our hope is that our timely, empirical, evaluation of campaign websites, particularly during a major US election cycle, would inform users on what data campaigns collect and share, and motivate users to expect stronger guarantees comparable to the for-profit context.

### 5.1.3 Why Should Researchers Care?

Although Section 5.1.2 provides evidence that users are likely to expect privacy guarantees for political data, we also entertain a counter-possibility: *what if users don't care at all?* One might argue that since users support the campaign, they may not expect the privacy guarantees they expect from for-profits, effectively “donating” their data for a cause. This argument motivates a pivotal question for researchers: *should we exempt campaign websites from analysis simply because users blindly trust them and want to help them?*

The answer to this question is no, both due to the debatable premise that users trust campaigns to such a degree, and the severe harms that will result from overlooking political campaigns just because users trust them. To elaborate, there is no evidence to suggest that users trust political campaigns enough to forego the rights to their private information in perpetuity. In fact, a prior survey demonstrates that users have a very dim view of the public sector, with only 11% considering them as “trusted” to protect their private information [130]. This potential lack of immediate trust could explain why campaigns have to resort to aggressive tactics to collect data, *e.g.*, such as staging a photo op with Santa Claus for kids, and then required voters to sign up with their email addresses to

download their children’s photos [208].

Alternately, even if a user did trust a campaign and willingly provided data for a cause, such blind faith may be unwarranted and harmful, due to campaign behavior *after elections*. For instance, a candidate in the 2016 presidential election was found to have *sold the email list collected via their website to rivals*, and separately, rented the email list out, charging \$10,500 per unsolicited email sent to their 675k subscribers [164]. Similarly, an app from a 2016 presidential campaign shared the contact list and location data of voters with Cambridge Analytica[214], potentially for building “psychological profiles” of the user and their contacts [206]. Regardless of how much the user supports a candidate’s cause, it is implausible that they would condone behavior such as selling out their data to rivals.

Finally, campaigns often switch party affiliation after the election [21], or alternately, voters themselves change their minds [220, 158]. This transience of voter-campaign relationships makes it ill-advised to assume that voters providing data to a campaign relinquish their rights in perpetuity. Considering these factors, it is incumbent on researchers to analyze the privacy posture of campaign websites, and usher in increased transparency into their data collection and usage, to protect users even if, and precisely because, they may blindly trust campaigns.

## 5.2 Ethical Considerations

Politics is a sensitive subject, and we are cognizant of the several ethical “lines” that this work could cross if performed without significant care. Therefore, to preempt harm, and with the goal of *uncovering privacy gaps agnostic of political implications*, we imposed a set of ethical constraints on this work, based on four guiding principles: (P<sub>1</sub>) *Focus on privacy, and not politics*, (P<sub>2</sub>) *Limiting harm to candidates*, (P<sub>3</sub>) *Limiting harm to campaign resources*, and (P<sub>4</sub>) *Transparency*. This section describes these principles and the constraints we impose to adhere to them.

### 5.2.1 P<sub>1</sub>: Focus on Privacy, and not Politics

Our goal is to highlight the gaps in the privacy postures of US political campaigns, and understand their implications *on user privacy*. As the loss of privacy affects users regardless of their political inclination, our position is that all political campaigns, regardless of affiliation, should adhere to privacy best practices. Therefore, we seek to limit our analysis and discussion to only what is relevant with respect to user privacy, and prevent a *partisan* interpretation of our results, as *user privacy should be a bipartisan issue*.

With this rationale, we impose the following constraint on the study: we refrain from analyzing our data in terms of specific political parties, affiliations, or the known political positions of individual candidates. To elaborate, we strip the party designation of candidates from the collected data before performing any analysis on it (which also prevents biasing ourselves), and do not later seek to attach party-specific insights in our findings. Instead, our analysis considers general congressional designations; *e.g.*, House, Senate and Presidential candidates, and committee memberships.

### 5.2.2 P<sub>2</sub>: Limiting Harm to Candidates

Although prior work explicitly discloses the names of organizations with privacy gaps (*e.g.*, PolicyLint[17], PoliCheck[18], TaintDroid[56]), we deliberately refrain from disclosing the identity of candidates/campaigns in our findings to prevent reputational harm. To elaborate, we do not name candidates and anonymize any identifiable information when describing the data, results, findings, or our interaction with candidates. Similarly, we anonymize the composition of sub-samples of campaigns chosen for our sharing analysis in Section 5.7 to further mitigate harm. Finally, we paraphrase our interactions with campaigns rather than quoting them verbatim and redact any personal information to prevent harm and de-anonymization of the candidate. As our only correspondence was for the responsible disclosure of findings, the interaction does not warrant seeking an IRB approval [133].

### 5.2.3 P<sub>3</sub>: Limiting Harm to Campaign Resources

Over the course of the study, we take several decisions to minimize or prevent harm to campaign resources or personnel. Particularly, we do not interact with any human subject or collect information about any human at any stage of our analysis. To analyze data sharing among campaigns, we use the same *black-box analysis* approach employed in prior work [163][126]. Specifically, our experiment also involves providing a *valid but fictional email address* to the campaign website and monitoring sharing by analyzing emails automatically delivered to our inbox, without the involvement of any human subject. As expected, we only received automated campaign ads during the entire study, which causes negligible harm to the system. Our only interaction with the campaigns occurred after the study, during the disclosure of findings, as described in Section 5.10.2. Finally, our automatic crawler respects `robots.txt` files in campaign websites, which we then collect manually.

### 5.2.4 P<sub>4</sub>: Transparency

While we take significant care to prevent a partisan interpretation of this chapter (P<sub>1</sub>), one might argue that not discussing political affiliations may in fact have a partisan effect, *i.e.*, that of hiding patterns of misbehavior in one political party. That is, the need for *full transparency* in terms of revealing the political affiliations is directly at odds with that of keeping the focus on privacy and off politics.

The position of the authors is that the benefits of a non-partisan interpretation of our results, in the form of enabling future research, informing the public, and motivating policymakers to regulate campaigns regardless of affiliation, are far superior to the perceived loss of transparency, *i.e.*, we choose to adhere to P<sub>1</sub>, at the cost of P<sub>4</sub>. However, to enable future researchers (or the general public) to make full (transparent) use of our data and analysis, we will release our raw data and code, including the crawled campaign websites. As this data is already publicly accessible, releasing it does not cross any ethical boundaries, while our framework, Polityzer, will allow researchers to perform similar analysis

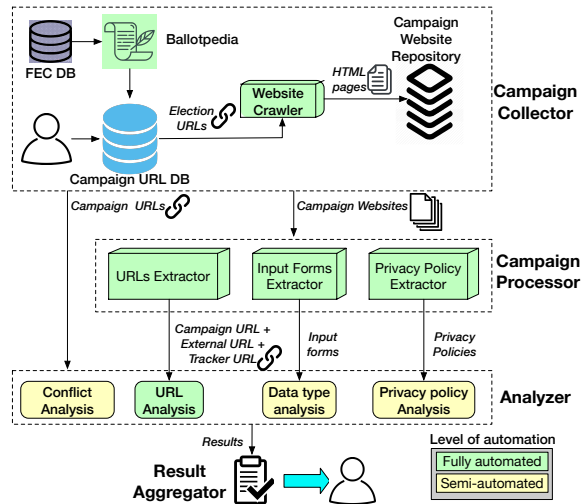


Figure 5.1: Overview of the Polityzer framework

based on political factors such as party affiliation, if they so choose.

### 5.3 Overview

Motivated by the criticality of campaign websites for user data collection, emerging expectations from governments regarding political data privacy, increasing user concerns, and the significant harms to users if campaign websites escape scrutiny, our analysis takes a *normative* position, *i.e.*, that *the websites of political campaigns be held to similar privacy standards as those of for-profit organizations*. This position reflects in our research questions (**Polityzer-RQ<sub>1</sub>**→**Polityzer-RQ<sub>4</sub>**), and guides our design and use of Polityzer.

Figure 5.1 illustrates the design of Polityzer, composed of the following modules:

- 1. Campaign Collector:** The Campaign Collector builds a database of the election campaign websites, and consists of a campaign URL database that houses the input website links of the campaigns, and an automated website crawler which downloads all the websites listed in the campaign URL database and stores them in a campaign websites repository for analysis. While campaign URLs can be directly fed into the Data collector manually, to enable large-scale collection, we develop an automated approach that collects candidate information from the FEC and uses Ballotpedia[22] to extract their campaign website URL

(see Section 5.4.1).

**2. Campaign Processor:** The Campaign Processor automatically extracts useful information from the website pages using 3 submodules; (i) the *URL Extractor* that extracts the external/outbound links as well as any trackers that may be present, (ii) an *Input Form Extractor* that extracts the input forms in the website through which user data is collected, and (iii) the *Privacy Policy Extractor* which checks whether the website has a privacy policy document and extracts it.

**3. Analyzer:** The Analyzer performs 4 types of analyses on the processed websites; (i) *Data type analysis (Polityzer-RQ<sub>1</sub>)* to understand the scope of private data collection (Section 5.5), (ii) *Privacy Policy Analysis (Polityzer-RQ<sub>2</sub>)* to understand disclosure practices (Section 5.6), (iii) *Conflict analysis (Polityzer-RQ<sub>3</sub>)* to discern conflicts in the privacy disclosures of campaigns (Section 5.7) and, (iv) *URL analysis (Polityzer-RQ<sub>4</sub>)* to characterize the general security posture of the campaign website (Section 5.8).

**4. Result Aggregator:** Once the analysis is complete, the Result Aggregator generates a privacy report containing the aggregated results per campaign. We describe the results of each submodule of Analyzer in Sections 5.4→5.8.

## 5.4 Campaign Collector

For our analysis to be tractable, we use the campaign collector to download the websites belonging to the following federal races in the U.S.: (1) *House* elections, (2) *Senate* elections, and (3) *Presidential* election, as previously described in Section 2.2. We now describe our methodology, followed by an overview of the resultant datasets.

### 5.4.1 Website Collection Methodology

The collector module takes campaign links as input and downloads and organizes the websites per campaign type i.e. House, Senate or President. The campaign links can be fed to Polityzer automatically or manually, depending on whether an automated process

is viable for a given election. For our analysis of the US-based campaigns, we constructed an automated pipeline using two data sources; (i) the FEC, and (ii) Ballotpedia. We now describe this pipeline followed by the methodology to build the campaign repository.

**1. Obtaining candidate lists from the FEC:** As discussed in Section 2.2, the FEC database is the ground truth for obtaining the list of all federal election candidates in the US election [63], which we use to obtain candidate names and metadata (*e.g.*, state, district, and party affiliation).

**2. Resolving campaign websites of individual candidates:** The FEC database does not link to the candidates’ websites as candidates are only obligated to provide names, addresses, and party information. Hence, the challenge is *to obtain a candidate’s campaign URL with only their name and metadata*, which we address using third-party sources along with web searches.

First, we search for corresponding candidate profiles in a non-profit political encyclopedia, *Ballotpedia* [22]. These profiles include the candidates’ campaign URLs, which we seek, the accuracy of which is ensured by dedicated Ballotpedia staff. However, obtaining a Ballotpedia profile for a candidate is non-trivial. To elaborate, a typical URL for a candidate’s Ballotpedia profile is simply `https://ballotpedia.org/<candidate-name>`. Just appending a candidate’s name to this URL template does not work, as the candidates use their legal name in their FEC registration while Ballotpedia profiles use their informal names that are often different; *e.g.*, “John Doe” (Ballotpedia profile name) could be registered as “Jonathan Doe” (legal name) with the FEC. Further, to distinguish candidates with the same name, Ballotpedia also appends the candidate’s home state to the profile URL, *i.e.*, `.../John_Doe_(State1)` and `.../John_Doe_(State2)`.

To address these challenges, we supplement our approach with Google search to identify the correct Ballotpedia profile URL of the candidates. For each candidate, we create a Google search query using both their FEC-provided name and the metadata (*e.g.*, state, party registration). We obtain the correct Ballotpedia profile by performing three

additional checks on the top 10 search results, *i.e.*, we check if (1) the URL root is `ballotpedia.org`, (2) the URL format is that of a Ballotpedia profile, *i.e.*, `ballotpedia.org/candidate_name` or `ballotpedia.org/candidate_name_(state)`, and (3) the URL contains the candidate’s last name. This approach always finds a candidate’s Ballotpedia profile, if present, because the search query involves relevant information (*e.g.*, `district/state`) that automatically narrows down the search space, never returning >one match.

Once we have the Ballotpedia profile URLs for the candidates, we automatically extract the campaign website link listed in the profile using an HTML parser. Note that not all candidates have websites listed on their Ballotpedia profile. We do not directly perform Web searches for the campaign website for such candidates, as there is no ground truth outside Ballotpedia to validate the mapping.

**3. Downloading websites to build the *campaign repository*:** The collector module uses an automated crawler built using scrapy-selenium [38] that spawns a headless browser with a user-agent specifying Mozilla as the browser. The crawler takes input in the form of a campaign URL and downloads the HTML pages as well as other website resources (*e.g.*, PDFs and CSVs) into a campaign repository. The crawler uses a breadth-first-search approach, and traverses all links up to a fixed *depth* starting from the homepage. Default depth is set at two to allow the crawler to finish within a reasonable timeframe, regardless of the relative website sizes in terms of number of links present, but can be re-configured as per user needs. Note that the crawler does not traverse outbound links (*i.e.*, links that point to external web pages on other domains), and instead saves the link and terminates that specific search.

### 5.4.2 Results of Campaign Website Collection

We collected all the House, Senate and Presidential candidates that were registered for the U.S. general election of 2020 in the FEC database [61] as of September 10, 2020, which yielded a total of 5036 candidates. This list contained duplicates, as the same candidate

may register for multiple offices simultaneously. Removing such duplicates brought the total number of unique candidates to 4885.

As described in Section 5.4.1, our automated pipeline to resolve candidate websites helped us map 3259 (66.7%) of the candidates to their respective Ballotpedia profiles. The missing 33.3% can generally be attributed to missing Ballotpedia profiles. Of these 3259 Ballotpedia profiles, 2630 provided a URL to the campaign website, *i.e.*, we obtained the campaign website links for 53.83% (2630/4885) of the overall sample of unique candidates identified from the FEC database. We further manually added 68 incumbent senators who were not participating in the election and hence, were absent from the FEC database, using the same methodology to obtain their campaign websites. We did not consider four retiring senators and one vice-presidential candidate whose website redirected to the presidential candidate’s website.

Finally, we used the crawler to download the campaign websites. Not all campaigns were active at this point, since the primaries (see Section 2.2) had completed and many candidates had dropped out. As a result, some of the website links were dead or led to 404 errors. The collector successfully downloaded **2060 (40.91%) websites**, including 13 that belonged in multiple groups *e.g.*, dropping their presidential campaign to run for house or senate. The downloads were completed on November 2, 2020, a day before the election.

To better understand the privacy practices in terms of candidates that appeared on the final ballot, versus those that had already dropped out, or those that are in office (and may or may not be re-contesting), we define a campaign “status” terminology. Campaigns whose candidates appeared on the final ballot in November were categorized as *active*, while those that did not (potentially due to a primary election loss) were classified as *inactive*. This classification was applied to the Senate, House, and Presidential campaigns, resulting in **6 datasets** (*i.e.*, *house\_active*, *house\_inactive*, *senate\_active*, *senate\_inactive*, *president\_active*, *president\_inactive*). Finally, the 68 incumbent senators not up for re-election were classified as *senate\_incumbents*, the **seventh dataset**.

**Table 5.1:** Dataset Overview

Dataset	No. in our dataset	Total in FEC	Cand./seat
<i>house_active</i>	952 (90.49%)	1052	7.49
<i>house_inactive</i>	710 (31.84%)	2252	
<i>senate_active</i>	112 (94.92%)	118	16.375
<i>senate_inactive</i>	151 (37.44%)	406	
<i>senate_incumbents</i>	68	-	-
<i>president_active</i>	4 (100%)	4	1208
<i>president_inactive</i>	63 (5.32%)	1204	
<b>total</b>	<b>2060 (40.91%)</b>	5036	

Table 5.1 shows the number of websites in the 7 datasets. The collector was able to download a higher percentage of websites for *active* candidates in all categories, which is expected, as active candidates were likely to have operational websites until the election. The lowest download rate occurred for *president\_inactive* dataset, which is also explainable, as the number of candidates was quite large for what would eventually be one seat. Finally, *all campaigns in senate\_incumbents had operational campaign websites even though they were not running*, potentially accruing user data even when the candidates may not be actively campaigning.

## 5.5 Analysis of Data Collection (Polityzer-RQ<sub>1</sub>)

To obtain an estimate of what is at stake, Polityzer first seeks to understand the collection of private user data by campaigns. Certain data types must be collected by US campaigns to fulfill donor reporting obligations imposed by the FEC. Hence, Polityzer covers all data types, irrespective of the FEC requirements, but distinguishes FEC vs non-FEC data types when discussing privacy implications.

### 5.5.1 Methodology

Polityzer approximates the types of private data collected by campaign websites through an analysis of HTML forms, using the Input Forms Extractor in the Campaign Processor module shown in Figure 5.1. The Input Forms Extractor automatically extracts all forms from each campaign website and extracts the “labels” adjacent to each input field, producing a label-set for each campaign website to enable analysis.

We use a manual approach to identify *private* data types from the label-set for each campaign. Our goal is to identify private data in a political context, such as voter registration information or party affiliation, as these political types may not be present in existing ontologies, *e.g.*, that of PoliCheck [18]. For coverage, we ensure that the final set for each campaign contains the union of types identified from our labeling and those in PoliCheck’s ontology.

### 5.5.2 Results and Findings

Table 5.2 shows the data types collected by campaigns and their distribution across our datasets. We split the collected data types into three categories: (1) *FEC-required* data types such as fine-grained location, employer information and occupation, (2) *non-FEC* data objects, *i.e.*, private data such as gender and party affiliation collected by campaigns at their own discretion, and (3) *FEC\* data types*, *e.g.*, information such as credit card numbers, or banking information that are not explicitly required by the FEC, but need to be maintained as a part of the donation records/receipts. We have manually identified and associated the data types to the corresponding campaign websites in each relevant finding in this section.

**Finding 1: 1462 (70.97%) of campaigns collected personal information through their websites ( $\mathcal{F}_1$ ).** Table 5.2 shows the 28 unique data types collected, with at least one candidate collecting data of each type. We found an additional 139 (6.75%) campaigns collecting political opinions on issues such as education, guns, and abortion rights. Although these opinions are not PII, they are often collected along with at least an email or a phone number on the same form, and hence, are sufficient to allow campaigns to build user profiles without explicit consent.

**Finding 2: Campaigns collect highly private data types that are not required by the FEC ( $\mathcal{F}_2$ ).** As shown in Table 5.2, all categories of campaigns, *i.e.*, house, senate or presidential campaigns as well as active or inactive categories, collect multiple data types that do not need to be reported to the FEC, indicating that the purpose of the website

**Table 5.2:** Data collected by campaign websites. **FEC** indicates data required by the FEC, **!-FEC** indicates data not required by the FEC, **FEC\*** indicates the data not explicitly required by the FEC, but often a part of donation receipts shared with the FEC.

Class	Data type	House		Senate			President		total
		<i>act.</i>	<i>inact.</i>	<i>act.</i>	<i>inact.</i>	<i>incumb.</i>	<i>act.</i>	<i>inact.</i>	
<b>FEC</b>	<i>name</i>	661	397	82	72	47	4	34	1297
	<i>location_coarse</i>	532	228	66	40	51	4	19	940
	<i>location_fine</i>	313	134	36	23	24	3	8	541
	<i>employer_info</i>	100	52	18	13	13	4	5	205
	<i>occupation</i>	95	45	16	11	14	2	4	187
<b>!-FEC</b>	<i>email_address</i>	709	443	88	80	58	4	42	1424
	<i>phone_number</i>	478	252	70	39	39	4	19	901
	<i>website</i>	78	52	11	5	3	2	5	156
	<i>education_info</i>	16	5	2	1	0	5	0	29
	<i>social_media</i>	15	5	1	2	1	2	1	27
	<i>language</i>	6	9	0	0	0	0	1	16
	<i>friend_email</i>	2	1	0	1	4	0	1	9
	<i>date_of_birth</i>	2	4	1	0	1	1	0	9
	<i>friend_name</i>	2	1	0	1	4	0	0	8
	<i>party_affiliation</i>	4	2	0	0	0	2	0	8
	<i>resume</i>	4	1	0	1	0	1	0	7
	<i>union_status</i>	2	0	0	0	0	1	0	3
	<i>photo_self</i>	1	1	0	0	0	0	0	2
	<i>fax_number</i>	4	0	0	0	0	0	0	4
	<i>age</i>	0	3	0	0	0	0	0	3
	<i>gender</i>	4	3	0	0	0	0	0	7
	<i>partner_name</i>	1	0	1	0	0	0	0	2
	<i>partner_employer</i>	1	0	1	0	0	0	0	2
	<i>parent_phone</i>	1	0	0	0	0	0	0	1
	<i>parent_email</i>	1	0	0	0	0	0	0	1
	<i>parent_name</i>	1	0	0	0	0	0	0	1
	<i>citizenship_status</i>	0	0	0	1	0	0	0	1
	<i>race</i>	0	1	0	0	0	0	0	1
<b>FEC*</b>	<i>creditcard_info</i>	34	15	4	7	6	1	0	67
	<i>banking_info</i>	6	3	2	1	3	0	0	15
	<i>pay_method</i>	24	20	9	6	4	1	0	64
	<i>retired_info</i>	34	12	7	3	9	1	0	66

extends beyond just collecting donation. Email and phone numbers are the most common, which campaigns may associate with other sources to form complete profiles on users, as discussed in Section 5.1. Other sensitive non-FEC data types are *education information*, *date of birth*, *resume*, *gender*, *party affiliation* and in rare cases, even *race*, union status and *photo* of the user. Such data is highly private, and may enable invasive campaigns to profile voters, and sell/share the profiles in perpetuity (see Section 5.1.3).

**Finding 3: Communication-related PII's are the most collected data types ( $\mathcal{F}_3$ ).**

Among campaigns that collect at least one data type, email is collected by at least 98.96%

**Table 5.3:** Top 10 most collected data types among campaign websites

<b>Datatype</b>	<b>count (total=1446)</b>
<i>email_address</i>	1431 (98.96%)
<i>name</i>	1304 (90.18%)
<i>location_coarse</i>	918 (63.49%)
<i>phone_number</i>	903 (62.45%)
<i>location_fine</i>	535 (37.0%)
<i>password</i>	245 (16.94%)
<i>employer_info</i>	174 (12.03%)
<i>occupation</i>	167 (11.55%)
<i>username</i>	150 (10.37%)
<i>website</i>	134 (9.27%)

campaigns (Table 5.3). Additionally, phone numbers are collected by 62.45%, meaning that communication related data types are among the most collected, and is indicative of the fact that communication with likely voters is one of the primary objectives of a campaign website. Note that some websites also collect username and passwords, most likely due to the presence of template login pages, as we verified with five randomly chosen websites. We have removed these two fields from the list of data types.

**Finding 4: Campaigns collect data that impacts the privacy of people *other than the user* ( $\mathcal{F}_4$ ).** We found that 12 campaigns collect contact information of the user’s friends and parents, *i.e.*, their names and email addresses, echoing prior evidence of campaign apps collecting contact lists [10] (see Section 5.1). These requests are often presented as a means of *sharing the campaign*, *i.e.*, wherein the user “shares” the campaign with their friends by submitting the friend’s name and email. Similarly, a campaign website collected information on the user’s parents as part of a fellowship application. At least two campaigns also collected partner’s name and employer information as part of the donation form. *Such collection is by nature without consent*, since the entity whose data is being shared cannot consent, and harms the privacy of people that are not directly interacting with the campaign. As we discuss later in  $\mathcal{F}_6$ , “friend’s email and name” were among the data types not disclosed in the privacy policies of several campaigns.

**Finding 5: Campaigns collect social and economic opinions of users along with their PII, thereby gaining the ability to directly associate individual users to**

**Table 5.4:** Campaigns’ collection of political opinions with PII in the same page

<b>Dataset</b>	<b>Collect political opinion with PII</b>	<b>No privacy policy</b>
<i>house_active</i>	34 (3.6%)	21 (61.76%)
<i>house_inactive</i>	12 (1.69%)	9 (75%)
<i>senate_active</i>	4 (3.57%)	0 (0%)
<i>senate_incumbents</i>	2 (2.94%)	1 (50%)
<i>senate_inactive</i>	5 (3.31%)	2 (40%)
<i>president_active</i>	2 (50%)	0 (0%)
<i>president_inactive</i>	2 (3.17%)	0 (0%)
<b>total</b>	61 (2.96%)	33 (54.10%)

**their political leanings** ( $\mathcal{F}_5$ ). 69 campaigns collect opinions on various issues such as abortion, immigration, guns, and taxes through a survey page or their volunteer sign up forms. A user’s beliefs on these issues is often a reliable indicator of their political leanings [70]. As it is a common practice to share voter data [163], this may lead to the exposure of the user to unsolicited micro-targeted ads from campaigns they did not directly interact with, even if the user trusts the specific campaign collecting the data (see Section 5.1.3). Of the 141 webpages of the 69 campaigns that collect such opinions, 127 pages (90.0%) from 61 campaigns (88.41%) do so while also collecting either an email or a phone number from the same page, allowing the campaigns to map the political stances to an individual. More importantly, using the analysis from Section 5.6, we find that *33/61 (54.10%) of these campaigns lack a privacy policy*, as we show in Table 5.4.

In summary, the collection of such a wide-range of private data underscores the critical position of websites in the data aggregation apparatus of campaigns.

## 5.6 Analysis of Privacy Disclosure (Polityzer-RQ<sub>2</sub>)

Privacy policies play a critical role in conveying how campaigns handle the significant sensitive data they collect via their websites. Moreover, privacy policies may also reveal if the data is being shared with third-parties such other political campaigns. This analysis focuses on understanding whether the privacy policies in campaign websites properly convey such relevant information accurately to the user.

### 5.6.1 Methodology

As existing privacy regulations (*e.g.*, CCPA [3]) do not apply to political campaign websites, we do not assume that a campaign website will have a “Privacy Policy” link on the main page, as is the best practice, also mandated by CCPA. Instead, we seek to find *any* form of privacy disclosure in the website, which allows us to deduce *if* campaigns disclose their privacy practices to users, and how accurate the disclosure are with respect to the private data they collect. Our analysis is organized in two steps:

**Step 1 – Checking the campaign website for a privacy disclosure:** For each campaign website, we first attempt to check if the website provides a privacy disclosure, *i.e.*, not a privacy policy per se, but any document that describes the collection and sharing of private data. For this, Polityzer searches campaign websites using a bag of words approach which searches for any hyperlinks (*e.g.*, `https://<campaign-url>.com/privacy-policy/`) or link-text (*e.g.*, “Privacy Statement”) that may contain terms indicative of a privacy disclosure. We obtain this set of disclosure-related terms from prior work [121], and increment it with additional words that may indicate privacy or any legal disclosure, specifically, “terms”, “conditions” and “disclosure”, leading to the following set of privacy disclosure-related terms: [privacy, terms, conditions, notice, statement, disclosure].

Finally, after Polityzer automatically extracts all the hyperlinks containing the disclosure-related terms in a campaign website, we manually check each shortlisted link to confirm whether it truly leads to a privacy policy page, which we extract for further analysis. In the spirit of performing a conservative analysis, we consider cases where the link pages led to empty/error pages as “having a privacy policy”, given the presence of the hyperlink.

**Step 2 – Analyzing the privacy policies for collection and sharing accuracy:** We use Polisis [88] to extract collection and sharing statements from the privacy policies, followed by manual annotation of data mentioned in the statements, and a comparison with the data actually collected by the campaigns, as found in Section 5.5. To elaborate, for each privacy policy, we automatically use the Polisis API [89] to obtain a category prediction for

each sentence (*e.g.*, “collection”, “sharing”). To elaborate, we extract sentences receiving the highest confidence scores for “collection” or “sharing”, which are most relevant to our goal.

Once the collection/sharing sentences are identified, we manually annotate them to identify data objects in the sentences (*e.g.*, name, address), looking for *precise* data objects and ignoring generic terms. For instance, in the sentence "we may collect personal information including your name and email", we annotate only the name and the email address. This annotation was performed by two authors in 10 days, wherein the first author identified the data objects in the sentences, which were then confirmed by the second author.

Finally, for each campaign, we compare the data objects extracted from the sentences with the corresponding set of data objects collected by the campaign’s website (*i.e.*, obtained from our analysis in Section 5.5). This analysis allows us to identify several types of anomalies, including data objects that the campaign collects but does not disclose in the privacy policy. Further, we also analyze the sharing sentences to identify explicit mentions of sharing data with other candidates, campaigns, or committees, and perform a general search for mentions of FEC disclosure requirements.

### 5.6.2 Results and Findings

**Table 5.5:** Missing privacy policies in campaign websites

	<b>w/o priv.policy</b>	<b>collecting priv. data</b>
<i>house_active</i>	640/952 (67.22%)	441/640 (68.98%)
<i>house_inactive</i>	589/710 (83.38%)	359/589 (60.95%)
<i>senate_active</i>	52/112 (45.54%)	34/52 (65.39%)
<i>senate_incumbents</i>	15/68 (23.53%)	11/15 (73.33%)
<i>senate_inactive</i>	122/151 (82.12%)	65/122 (53.28%)
<i>president_active</i>	0 (0%)	0 (0%)
<i>president_inactive</i>	40/63 (66.67%)	27/40 (67.5%)
<b>total</b>	1458/2060 (70.78%)	937/1458 (64.27%)

Out of the 2060 websites we analyzed, Polityzer’s automated approach for privacy link extraction led to 975 campaigns with potential privacy disclosures (*i.e.*, 975 links), and conversely, 1085 websites without disclosures. Recall that Polityzer’s automated ap-

**Table 5.6:** Total privacy policy extracted for analysis per dataset

<b>Dataset</b>	<b>No. of priv.policy extracted</b>
<i>house_active</i>	267
<i>house_inactive</i>	90
<i>senate_active</i>	56
<i>senate_incumbents</i>	49
<i>senate_inactive</i>	19
<i>president_active</i>	4
<i>president_inactive</i>	22
<b>total</b>	507

proach is conservative, *i.e.*, it gives significant benefit of the doubt to campaigns and over-approximates to find *all potential* privacy disclosures. Hence, its effectiveness is in terms of a low false positive rate, with a positive being the lack of a privacy disclosure. Upon manual validation of this result, we find only 14/1085 false positives, *i.e.*, a false positive rate (FPR) of 1.29%. Of the 14/1085, four used different terms to describe their disclosure (*i.e.*, disclaimer, transparency, fine print, and ToS), five were hosted with third party sites (*e.g.*, *pastebin*), and five resulted from link extraction errors in Polityzer.

We further manually refined the 975 potential privacy disclosure links from Polityzer, and found 560 actual privacy policies, and 415 were not. Of these 560 we were able to extract a ***final 507 disclosures for analysis***, with the rest leading to 404 errors. Table 5.6 shows the distribution of analyzed privacy policies per dataset.

We also validated the 415 websites without disclosures from our manual refinement of the 975 potential disclosure links, and found that 28 were marked in error. To summarize, we identified 1458 websites as lacking privacy disclosures, considering 42/1500 false positives (*i.e.*, overall 2.8% FPR), with Polityzer’s automated keyword-based approach suffering from only 1.29% (14/1085) FPR.

Finally, recall that we used Polisis to identify collection and sharing statements from privacy policies, and filter out the rest. Thus, “effectiveness” in this context would be the ability of Polisis to identify most of the collection/sharing sentences, and conversely, filter out or miss as few as possible, *i.e.*, have few false “negatives” (with a positive being a relevant collection/sharing sentence). To understand if using Polisis filters out relevant

sentences, we manually validated each of the 14,454 sentences filtered out by Polisis. We found that only 465/14454 sentences were incorrectly labeled as not being “collection” (339 sentences) or “sharing” (126 sentences), *i.e.*, a false negative rate of 3.22%. We manually integrated these sentences into our analysis.

Our analysis of privacy disclosures led to the following findings, which have all been manually validated.

**Finding 6: 1458/2060 (70.78%) of the campaign websites did not have a privacy disclosure, of which, 937/1458 (64.27%) collect private data ( $\mathcal{F}_6$ ).** As shown in Table 5.5, of the 70.78% campaigns lacking privacy disclosures, it is concerning that 937 (64.27%) collect private user data, including sensitive information such as credit card, employer information, phone number, and location. We also observe that active campaigns were more likely to offer a privacy disclosure than inactive campaigns, likely due to the longer period of time the campaign websites were actively engaging with users and potentially subject to scrutiny.

**Table 5.7:** Data collection without privacy disclosure.

<b>Dataset</b>	<b>Undisclosed</b>
<i>house_active</i>	118/267 (44.19%)
<i>house_inactive</i>	49/90 (54.44%)
<i>senate_active</i>	10/56 (17.86%)
<i>senate_incumbents</i>	16/49 (32.65%)
<i>senate_inactive</i>	7/19 (36.84%)
<i>president_active</i>	3/4 (75%)
<i>president_inactive</i>	6/22 (27.27%)
<b>total</b>	209/507 (41.22%)

**Finding 7: 209/507 (41.22%) of campaigns do not fully disclose all private data in their privacy policy. ( $\mathcal{F}_7$ ).** Similar to  $\mathcal{F}_5$ , Table 5.7 shows how inactive campaigns are more likely not to disclose all collected private data, relative to active campaigns. Table 5.8 lists the top 10 undisclosed types. These data types that were not disclosed include (1) data types that were most collected such as phone number (111/507 or 21.89%), email (88/507 or 17.36%) and location (47 or 9.27%), (2) data types known to be shared with third-parties (including the FEC) such as occupation (24/507 or 4.73%) and employer

**Table 5.8:** Top 10 Most commonly undisclosed datatypes

Datatype	count (total=507)
<i>phone_number</i>	111 (21.89%)
<i>email_address</i>	88 (17.36%)
<i>location_coarse</i>	80 (15.78%)
<i>name</i>	71 (14.01%)
<i>location_fine</i>	47 (9.27%)
<i>password</i>	34 (6.71%)
<i>employer_information</i>	29 (5.72%)
<i>occupation</i>	24 (4.73%)
<i>website</i>	14 (2.76%)
<i>credit_card_info</i>	13 (2.56%)

information (29/507 or 5.72%), and (3) sensitive demographic data such as retirement status (10/507 or 1.97%), party affiliation (4/507 or 0.79%) and race (2/507 or 0.39%). Even data types that affect the privacy of the user’s friends (*e.g.*, friend’s email, 2/507 or 0.39%) were not disclosed. Conversely, 316/507 (62.33%) campaigns disclose data types that they do not collect in practice, potentially due to the use of templates (see Section 5.9).

**Finding 8: 389/507 (76.73%) campaigns disclose sharing with third-parties in their privacy policy ( $\mathcal{F}_8$ ).** However, this does not mean they explicitly mention who the third-party is; *i.e.*, campaigns may not mention sharing with other political campaigns, or even with the FEC. As campaigns are known to sell voter data after elections (see Section 5.1.3), this lack of transparency is particularly concerning.

**Table 5.9:** Campaign websites that disclose sharing data with other campaigns

Dataset	Share with other campaigns
<i>house_active</i>	91/267 (34.08%)
<i>house_inactive</i>	23/90 (25.56%)
<i>senate_active</i>	23/56 (41.07%)
<i>senate_incumbents</i>	20/49 (40.82%)
<i>senate_inactive</i>	5/19 (26.32%)
<i>president_active</i>	2/4 (50.0%)
<i>president_inactive</i>	15/22 (68.18%)
<b>total</b>	179/507 (35.31%)

**Finding 9: 179/507 (35.31%) campaigns mention sharing data with other political campaigns ( $\mathcal{F}_9$ ).** Sharing with other campaigns, especially to the campaign’s central party, may be a standard practice [163]. However, only 35.31% of mention sharing their

data with other political campaigns, despite its privacy implications on the user. This trait is more prevalent among *active* campaigns (see Table 5.9). Similarly as in the case of  $\mathcal{F}_8$ , this finding reflects the norm for campaigns to share data with other campaigns, sometimes even rivals [164].

**Finding 10: None of the campaigns explicitly discuss their retention practices in relation to the completion of campaign ( $\mathcal{F}_{10}$ ).** We used keyword-matching using ‘retain’ or ‘retention’ to find 165/507 campaigns that discuss retention practices. Of these 165 campaigns, none of them explicitly discuss what happens to user data upon the *completion of the campaign*. When campaigns do discuss retention of data for a period of time in 54 instances, they do so by providing vague and non-descriptive explanation. For instance, campaigns may say they retain data as long as "necessary for business purpose“, or "necessary for fulfillment of purpose for which data was given“. Further, with the campaign likely ending, users have no recourse to prevent data misuse once it is collected. This finding shows why researchers must investigate campaign websites regardless of how much users trust them (Section 5.1.3), as without committing to a retention policy, campaigns gain perpetual access to user data, which is undesired given the transience of user-campaign relationships, and general campaign (mis)behavior after elections.

## 5.7 Inter-campaign Sharing Analysis (Polityzer-RQ<sub>3</sub>)

Users may expect their shared data to stay with the campaign they engaged with. This section explores data sharing among campaigns through an experiment, and by analyzing the conflicts between the privacy disclosures of political campaigns and major fundraising platforms.

### 5.7.1 Methodology

We performed two studies to understand the privacy implications of data sharing among campaigns. First, we conducted an *email experiment* to evaluate if candidates share private email addresses with others. Second, we identified campaigns connected to the two most

prominent fundraising platforms and analyzed the privacy policies of the platforms as well as the connected campaigns for conflicts.

**1. Experimentally studying email data proliferation:** We signed up for the newsletters of 26 campaigns evenly distributed among major political parties, consisting of ten Senate, ten House and two Presidential candidates, as well as four House and Senate campaigns from our state, to increase the likelihood of a response owing to the local reference.

To observe the effects of sharing our emails with each of the 26 entities in isolation from the others, we used 26 dedicated email accounts. We examined the domains of the senders to identify emails from external parties, *i.e.*, if the domain differed from that of the campaign website we signed up with. Additionally, we visited the external domain to identify its affiliation (*i.e.*, another campaign, or a PAC). We refer the readers to the study by Podob et al. [163] for a more exhaustive analysis of the sharing practices involving all major campaigns in the 2016 election.

**2. Understanding conflicts among the privacy policies of platforms and campaigns:** We observe that political campaigns often create parallel instances on fundraising platforms such as ActBlue[1], WinRed[5], Anedot[2] or DonorBox[7], which act as payment providers and also central avenues for attracting voters. As campaigns point to these platforms from their websites, data exchange between the campaign website and the fundraising platform is highly likely. Therefore, we explore conflicts between the privacy policies of campaigns and their fundraising platforms.

We design a simple approach to identify campaign websites connected to two major fundraising platforms, Platform<sub>1</sub> or Platform<sub>2</sub>. For each of the 2060 candidate websites we automatically extract all the outbound links, and identify a connection if the root of any of the outbound link is Platform<sub>1</sub> or Platform<sub>2</sub>. We then compare the collection and sharing statements from the privacy policies of the two platforms with those of the connected campaigns (extracted using the methodology described in Section 5.6).

### 5.7.2 Results and Findings

We categorize the emails received between November 5, 2020 to February 20, 2021 as *during-election* emails, as the period includes the November election and the Senate runoff election in Georgia. The emails from February 21, 2021 to September 14, 2022 are categorized as *after-election* emails. We received 1708 during-election emails and 933 after-election emails, with an average of 65 during-election and 35 after-election emails per campaign. All the findings from the analysis of these emails have been manually validated.

**Finding 11: 3/26 campaigns shared our email with another entity without disclosing in the privacy policy ( $\mathcal{F}_{11}$ ).** In total, 8/26 (30.77%) campaigns we studied shared our emails with other political entities (such as PACs), five during-election and three after-election. Of these eight, *three make no mention of sharing user data with other political entities* in their respective privacy policies.

**Table 5.10:** Use of Platform<sub>1</sub> or Platform<sub>2</sub> for fundraising.

Dataset	Use platform	No privacy policy
<i>house_active</i>	567/952 (59.56%)	341/567 (60.14%)
<i>house_inactive</i>	269/710 (37.89%)	210/269 (78.07%)
<i>senate_active</i>	63/112 (56.25%)	11/63 (17.46%)
<i>senate_incumbents</i>	55/68 (80.88%)	13/55 (23.64%)
<i>senate_inactive</i>	60/151 (39.74%)	46/60 (76.67%)
<i>president_active</i>	3/4 (75.0%)	0/3 (0%)
<i>president_inactive</i>	22/63 (34.92%)	10/22 (45.45%)
<b><i>total</i></b>	1039/2060 (50.44%)	631/1039 (60.73%)

**Finding 12: Of the 1039 campaigns that use fundraising platforms, 631/1039 (60.73%) do not have a privacy policy ( $\mathcal{F}_{12}$ ).** According to their privacy policies, both Platform<sub>1</sub> and Platform<sub>2</sub> *share user data with the campaigns*. However, since 631 such campaigns do not have privacy disclosures, the privacy policies of both platforms are rendered ineffectual in practice, *i.e.*, the privacy guarantees promised by the platforms to donors do not hold, due to data sharing with campaigns that provide no disclosure or guarantees at all.

**Finding 13: Campaigns using Platform<sub>1</sub> for fundraising may be indirectly sharing**

**with other campaigns** ( $\mathcal{F}_{13}$ ). Platform<sub>1</sub> in its privacy policy states that it may share user data with third parties for marketing purposes, including *other political committees or campaigns that may be of interest* to the user. This means that users donating to one campaign may have their data shared with other campaigns. Hence, we argue that in the spirit of good disclosure, campaigns should explicitly specify such sharing in their privacy policies. However, of the 162 campaigns that use Platform<sub>1</sub> and have a privacy policy, 144 (88.89%) campaigns do not disclose sharing with other campaigns or Platform<sub>1</sub>, despite using Platform<sub>1</sub> for fundraising.

## 5.8 Security Risk Analysis (Polityzer-RQ<sub>4</sub>)

Campaign websites collect extensive amounts of private and sensitive user data (Section 5.5), often without adequate disclosure (Section 5.6, Section 5.7). Therefore, it is important to evaluate the general security hygiene of these websites to develop an understanding of the risks associated with malicious or unintended data disclosure. We performed three analyses, described below, followed by the findings.

**1. Identifying malicious/phishing URLs:** Building upon prior work [120, 30, 96] that use VirusTotal [13] as ground truth for identifying malicious websites, we first analyzed each campaign website by passing each URL (including outbound links) included in the website through VirusTotal’s API. We then aggregated the results from VirusTotal by checking how many of the scanning engines in VirusTotal marked a URL as either “malicious” or “phishing”.

**2. Identifying and characterizing trackers:** To check for the presence of trackers, we used an existing tracker list that was originally designed for AdBlock [6]. For each campaign website, we check if any of the associated URLs match the regular expression-based rules from the tracker list. We also extract the root domain of the tracker upon identification. As a final step to identify *malicious* (*i.e.*, and not just undesirable) trackers, we run the list of URLs of discovered trackers through VirusTotal.

**3. Checking whether websites use TLS:** We send a GET request to the website URL using https and label a website as “using TLS” if the response is successful.

Finally, we use APIVoid [12] to obtain general hosting information for deducing the jurisdiction that would apply to the campaign website in case security problems were found. For each candidate URL, we query APIVoid using REST APIs and obtain the following information: the IP address of the server hosting the URL, server’s hosting company, and the country where the website is hosted. We further analyze this data to identify campaigns hosted outside of the US, since this study focuses on US political campaigns.

**Table 5.11:** No. of unsafe campaign websites across datasets

Dataset	No. of unsafe sites
<i>house_active</i>	11/952 (1.16%)
<i>house_inactive</i>	4/710 (0.56%)
<i>senate_active</i>	0/112 (0%)
<i>senate_incumbents</i>	0/68 (0%)
<i>senate_inactive</i>	0/151 (0%)
<i>president_active</i>	0/4 (0%)
<i>president_inactive</i>	2/63 (3.17%)
<b>total</b>	17/2060 (0.83%)

**Finding 14: Campaign websites are generally secure ( $\mathcal{F}_{14}$ ).** Although 17/2060 (0.82%) campaign websites were flagged as unsafe by at least one of the engines in VirusTotal (Table 5.11), only four among them were flagged by at least two engines and only one by more than two engines. This shows that at least 2052 (99.18%) campaign websites were marked as secure by VirusTotal. Our results are conservative as we cannot analyze false negatives, *i.e.*, confirm the absence of malicious code in candidate websites, since Polityzer’s dataset only consist of html pages and not the associated scripts.

**Finding 15: Campaign websites are hosted on servers outside of the US ( $\mathcal{F}_{15}$ ).** In all, 53/2060 websites were hosted by servers located outside the US, 15 of which were well-known service providers such as CloudFlare and Google. After their removal, we finally got 38 campaign websites hosted outside the US, in countries including Czechia, Denmark, and Japan (full list in Table 5.12). We find that 33 of these sites belonged to *inactive* campaigns while 5 belong to *active* campaigns.

**Table 5.12:** non-US countries where websites are hosted

Country	Num of campaigns
<i>Canada</i>	11
<i>Germany</i>	11
<i>Australia</i>	5
<i>Japan</i>	3
<i>France</i>	2
<i>VietNam</i>	1
<i>UK</i>	1
<i>Lithuania</i>	1
<i>HongKong</i>	1
<i>Czechia</i>	1
<i>Denmark</i>	1

As this analysis was performed after the election, it is unclear if the websites were always hosted offshore, or bought by an offshore entity after the election. It is also possible that once the campaign ends, URLs are bought by offshore entities for potentially malicious future use, *e.g.*, one URL in an *inactive* House campaign server is hosted by a company called the *Iranian Research Organization for Science & Technology* located in Hong Kong.

That five *active* campaigns are hosted offshore is concerning, as they were still active and collecting data at the time of the analysis. Due to the diverse laws governing data in different countries, such offshore storage can have serious privacy implications for users; *e.g.*, recent changes in Hong Kong’s data security laws that allow the government to access data stored in Hong Kong’s data centers [9, 11].

**Table 5.13:** No. of non-TLS websites

Dataset	non-TLS sites	Collect PII
<i>house_active</i>	66/952 (6.93%)	44/66 (66.67%)
<i>house_inactive</i>	65/710 (9.15%)	28/65 (43.08%)
<i>senate_active</i>	12/112 (10.71%)	6/12 (50%)
<i>senate_incumbents</i>	2/68 (2.94%)	2/2 (100%)
<i>senate_inactive</i>	16/151 (10.60%)	2/16 (12.5%)
<i>president_active</i>	0/4 (0%)	0
<i>president_inactive</i>	7/63 (11.11%)	4/7 (57.14%)
<b><i>total</i></b>	168/2060 (8.16%)	86/168 (51.19%)

**Finding 16: 168 (8.16%) campaign websites do not use HTTPS for communication ( $\mathcal{F}_{16}$ ).** We observe that HTTPS adoption rate among the campaign websites may be better than HTTPS adoption in general, which is around 80% for Alexa top

100,000 [219][218]. As shown in Table 5.13, 86 (51.19%) of these non-HTTPS campaign websites collect PII including phone numbers, fine-grained location, and credit card data.

**Finding 17: Campaign websites have malicious outbound links ( $\mathcal{F}_{17}$ ).** 71 campaigns had at least one outbound link that was classified as malicious by at least two engines in VirusTotal. While the campaign website is unlikely to be malicious, this result indicates that the links that campaigns include within their website may not be adequately vetted.

**Table 5.14:** No. of campaign websites with trackers

Dataset	w/ trackers	w/o priv.policy
<i>house_active</i>	741/952 (77.84%)	467/741 (63.02%)
<i>house_inactive</i>	457/710 (64.37%)	361/457 (78.99%)
<i>senate_active</i>	90/112 (80.36%)	34/90 (37.78%)
<i>senate_inactive</i>	100/151 (66.23%)	74/100 (74%)
<i>senate_incumbents</i>	67/68 (98.53%)	15/67 (22.39%)
<i>president_active</i>	4/4 (100%)	0 (0%)
<i>president_inactive</i>	45/63 (71.43%)	23/45 (51.11%)
<b>total</b>	1504/2060 (73.01%)	974/1504 (64.76%)

**Finding 18: 1504 (73.01%) campaign websites use trackers ( $\mathcal{F}_{18}$ ).** Trackers are used extensively among the campaign websites, but are more likely in *active* campaigns (Table 5.14). We found 280 unique trackers in the websites with `www.google-analytics.com` and `connect.facebook.net` being the two most common. Among the 280 trackers, two were identified as malicious: `ad.yieldmanager.com` (by one VirusTotal engine) and `www.freeresultsguide.com` (by three engines).

**Finding 19: 974/1504 (64.76%) of campaign websites with trackers do not have a privacy policy ( $\mathcal{F}_{19}$ ).** Similar to Findings  $\mathcal{F}_5$  and  $\mathcal{F}_{10}$ , *inactive* campaigns across each data set are more likely to not have privacy policies despite having trackers in their websites, which may lead to their users not even being aware of possible data collection.

**Finding 20: 112/446 (25.11%) campaign websites do not mention trackers in their privacy policies ( $\mathcal{F}_{20}$ ).** This is in keeping with the privacy policies of campaign websites missing key data types, as we detailed in Finding  $\mathcal{F}_6$ . In both Findings  $\mathcal{F}_{17}$  and

$\mathcal{F}_{18}$ , it is important to note the loss of user data privacy resulting from trackers [59] *e.g.*, trackers from Facebook or google analytics can collect privacy-sensitive user data from websites[127].

## 5.9 Use of Privacy Policy Templates

**Table 5.15:** Extra data mentioned in the privacy policy but not collected in the website.

<b>Dataset</b>	<b>Extra in priv.policy</b>
<i>house_active</i>	155 (58.05%) (total=267)
<i>house_inactive</i>	58 (64.44%) (total=90)
<i>senate_active</i>	36 (64.29%) (total=56)
<i>senate_incumbents</i>	36 (73.47%) (total=49)
<i>senate_inactive</i>	12 (63.16%) (total=19)
<i>president_active</i>	3 (75%) (total=4)
<i>president_inactive</i>	16 (72.73%) (total=22)
<b>total</b>	316 (62.33%) (total=507)

We found that 316/507 (62.33%) campaigns include data types in their privacy policy that they do not collect in practice, as seen in Table 5.15. Some unique data types that fall under this category are tax ID number, religion, phone contact list, and mobile device ID number. The presence of such *surplus* datatypes can be explained in a number of ways. First, the privacy policy of both the website and the mobile app of the campaign (if present) could be the same, which means the data may still be collected, just not via the website. Second, the campaign may intend to collect such data in the future or is using a template privacy policy without removing such extra datatypes. Use of policy templates among campaign websites is likely, based on our comparison of privacy policy texts, that we discuss next. Finally, this could also be due to a gap in our analysis, as we only analyze the form labels in webpages, and may miss form inputs if the form is improperly labeled in the html.

To gauge the use of privacy policy templates across different campaigns, we compared the text of each privacy policy with the rest of the privacy policies in our dataset. We do this by first converting the privacy policy text into TF-IDF vectors [86] and calculating their cosine similarity [14] with each other. For high likelihood of similarity, we only

consider policies with over 98% of cosine similarity score as similar. Finally, we randomly choose 5 privacy policies that have at least 1 similar privacy policy to assess their overall privacy implication.

**Finding 1: 239 privacy policies have at least one highly similar corresponding policy..** The cluster with the highest number of similar privacy policies is 23, while the cluster with the least number is 2. In the random sampling of 5 clusters we manually analyze, the main part of the policy text were identical, with the only textual difference being in the introductory paragraph. Due to this, the data types described within the privacy policies were also identical.

**Finding 2: Campaigns with similar privacy policy did not have similar data collection practices..** In our analysis of 5 randomly chosen similar privacy policy clusters of sizes 20, 8, 7, 4 and 2 respectively, we found no similarity in the types of data they collect, despite the fact that the disclosure about the data types being the same. This likely indicates that the privacy policies were simply written from a template (*e.g.*, by replacing the name of the campaign and the candidate) rather than as a way to rigorously inform the user about the privacy practices *corresponding* to that campaign website.

## 5.10 Discussion

The severe privacy violations demonstrated in our analysis are all legal in the U.S., given the lack of a dedicated privacy regulation applying to political campaigns. However, “legal” does not mean “appropriate” here, *i.e.*, similar violations by commercial websites would have attracted significant scrutiny and criticism from both regulators and researchers, as they have in the past [39, 45]. Our position is succinctly captured in this quote from the Online Trust Alliance [105], who audited apps belonging to presidential candidates in the 2016 election: *In light of worldwide privacy concerns and the court of public opinion, are the candidates’ practices considered responsible or ethical? Should the next president of the United States be held accountable to the same standards as a business?* [203]. That is,

we believe that the findings from this study expose inconsistencies that go beyond what is expected in keeping with the spirit of good disclosure, as well as laws in prominent (non-US) jurisdictions, and general consumer expectations of privacy. Several campaigns that follow privacy best-practices may adhere to this view as well, such as the 29.22% that provide privacy disclosures, 58.78% of which disclose all data collection.

In summary, the last decade has seen significant advancements in consumer data privacy due to the concerted efforts of researchers to change the perceptions of both governments and consumers, and this work seeks to initiate a similar transformation in this highly relevant domain. To this end, we organize the discussion along three key areas: We summarize the privacy implications of the findings (limitations discussed in Section 5.11). We then explore *why* the campaigns' privacy postures are this way, leveraging the responses from campaigns contacted for responsible disclosure. Finally, we conclude with actionable outcomes from this study that would help researchers as well as regulators bring about tangible change and accountability in data privacy in this domain.

### 5.10.1 The Privacy Posture of Campaign Websites

Our findings show that campaign websites collect extensive amounts of highly sensitive data ( $\mathcal{F}_1 \rightarrow \mathcal{F}_5$ ), and confirm the important position websites occupy in the campaigns' data aggregation apparatus, as outlined in Section 5.1. While the significant collection of private data *not required by the FEC* ( $\mathcal{F}_2$ ) is indeed concerning, we find that the privacy risks from this collection are made severe due to the sharing practices, and a general lack of transparent disclosure.

To elaborate, aside from common privacy violations, such as the lack of a privacy policy ( $\mathcal{F}_6$ ) or incomplete policies ( $\mathcal{F}_7$ ), we find that many campaigns use boilerplate language regarding sharing ( $\mathcal{F}_8$ ), and some even share data with other campaigns without disclosing such sharing at all ( $\mathcal{F}_{11}$ ). Similarly, most campaigns with access to data from fundraising platforms lack privacy disclosures entirely ( $\mathcal{F}_{12}$ ), rendering ineffective the guarantees promised in the platform's disclosure, as well as the disclosures of other campaigns that

provide data to the platform ( $\mathcal{F}_{13}$ ).

What is worse is that no campaign precisely discloses what happens to the data after completion of the campaign ( $\mathcal{F}_{10}$ ). The implication here is that once users provide data, campaigns may use, share, and sell the data in perpetuity, without the data owner’s consent. This perpetual ownership campaigns acquire not only exposes users to privacy harms (*e.g.*, data being sold to rivals [164]), but also to security risks such as identity fraud and surveillance given the potential for data leaks [202, 16], especially in cases where the websites are hosted in non-US jurisdictions ( $\mathcal{F}_{15}$ ), use vulnerable communication ( $\mathcal{F}_{16}$ ), or are connected to malicious, non-vetted, entities ( $\mathcal{F}_{17}$ ). The undisclosed presence of aggressive trackers in many campaign websites ( $\mathcal{F}_{19}$ ,  $\mathcal{F}_{20}$ ) compounds these harms, by exposing the user’s general browsing habits to the campaigns as well.

To summarize, the collection of a significant range of private data, coupled with insufficient disclosure, bad security practices, and undisclosed sharing, exposes users to significant privacy risks and loss of agency. To put these findings into context, we make two final observations:

*Observation 1* ( $\mathcal{O}_1$ ) – The campaign websites of 253 current *lawmakers* did not have a privacy policy, of which, 200/253 collect personal information.

*Observation 2* ( $\mathcal{O}_2$ ) – 99 of these 253 lawmakers serve on privacy-relevant congressional committees on cyber, technology, or consumer protection. Such committees often scrutinize the security or privacy practices of businesses, *e.g.*, 4 of these lawmakers participated in a Senate hearing titled “*Does Section 230’s Sweeping Immunity Enable Big Tech Bad Behavior?*”. We hope that the findings from this study help responsible members of congress in holding their own campaigns to the same standards they govern.

### 5.10.2 Rationale for the Present Privacy Postures

During the responsible disclosure of our findings to campaigns without privacy policies, we received 20 responses that provide insight into the campaigns’ rationale regarding data privacy (see the online appendix [165] for details).

Particularly, 6/20 campaigns were open to adding a privacy disclosure to their websites, but were ill-equipped due to the lack of technical support or privacy know-how, some even asking us for a template. These responses are encouraging as they show a *willingness to follow privacy best-practices*. In contrast, 5/20 campaigns misunderstood the rationale behind privacy disclosures, (incorrectly) arguing that privacy policies are non-binding, and hence ineffectual. Another argued that since they did not collect data (which we verified to be correct), they did not need one, which goes against commonly understood best-practices.

Further, some (3/20) did not consider their campaigns active, and hence saw no need to retroactively add a privacy policy. However, we note that the websites were still active at the time of this exchange, and could have been collecting data. Some others confused our inquiry with their stance on privacy in general, or mistook us as service providers proposing to create a policy for them (which could also explain the lack of responses from campaigns).

Finally, 2/20 campaigns admitted that the absence of the privacy policy was directly *because of the lack of federal privacy regulation* for campaigns. One candidate expressed frustration at their party's privacy posture, and suggested us to convince their party to require their candidates to have a privacy policy. The same candidate stated that they were asked by the central party to share the campaign's donor list, corroborating our findings. Finally, the candidate also expressed the need for dedicated resources for campaigns, such as a website that explains the best practices, provides templates, and how-tos, thereby aiding the largely volunteer-run campaigns to develop a good privacy posture.

### **5.10.3 Towards Privacy, Transparency, and Accountability in Political Campaign Websites**

This chapter of the dissertation develops artifacts and insights that will benefit researchers, users, and policymakers alike. Particularly, our data, results, and the HomeEndorser framework will help researchers further explore privacy in the context of political campaigns, and extend our methodology and analysis pipeline to analyze other relevant artifacts, such

as campaign-related mobile apps. Moreover, researchers will be able to use HomeEndorser to periodically evaluate campaigns, enabling longitudinal understanding of the privacy postures of political campaign websites.

Similarly, we are encouraged to see legislative efforts [215] towards regulating the privacy practices of political campaigns in the U.S., particularly their digital components, such as websites and apps. We envision that the measurement results and findings from this study, as well as future research that builds upon it, will provide empirical grounding for such legislative efforts. For instance, our findings motivate the dire need to require campaign websites to provide privacy disclosures, particularly including details on how long they retain user information. Such a criteria will not only force campaigns to be transparent about their routine sharing or sale of data after the election, but also enable users to make informed choices when committing their data to a particular campaign.

Finally, we see significant privacy benefits to users down the line. Particularly, we find the general obscurity on the users' privacy expectations from campaigns unsurprising, given the lack of privacy studies to that end. The data and findings from our large scale study provide a unique opportunity for researchers to address this gap, repeating prior work on gauging user privacy expectations [161] in this critical context. More importantly, we hope that just as prior work [161] found, presenting the public with the key measurements and findings regarding the privacy practices of campaign websites may also educate them on their privacy implications, motivating *informed* voters, who will then demand increased privacy, accountability, and transparency, from the campaigns. This, in turn, may be the final push needed for strong privacy legislations governing political data in the U.S., just as user privacy concerns motivated regulations such as the GDPR and CPRA.

## 5.11 Threats to Validity

We list the threats to validity of this study, as follows:

**Completeness of website collection and email study:** Our dataset does not include candidates whose Ballotpedia profile was not found or who did not have a campaign website

link in their Ballotpedia profile. Similarly, our dataset does not include Political Action Committees (PACs) and SuperPACs, as our primary focus was on the privacy practices of campaigns. Further, we started the crawling from September 15, 2020, which was after the primaries, so we may have “lost” some data that could otherwise have been collected.

**Completeness of email study:** The email study encompasses emails received between November 5, 2020 to September 14, 2022, and only considers the top donation-earners. For a more exhaustive analysis of the sharing practices of campaigns, we refer the readers to the study by Podob et al. [163] which includes all major campaigns in the 2016 election over a full election cycle.

**Completeness of data type collection:** For compiling the data types collected by each campaign website, we automatically extract all the forms in all the webpages of the website and extract the “labels” in those forms. In doing so, we may miss cases where input fields are not bounded by proper `<form>` tags in the webpages or cases where input fields that are present within forms are not properly labeled. Additionally, one author manually resolved each of the extracted labels based on the limited context provided by the label texts. The author discarded any labels that could not be reasonably resolved (*e.g.*, label texts such as `input_1`), including the labels meant for automated crawlers (*e.g.*, ‘leave this box empty’). Hence, our methodology of data type collection (and the resultant findings) offers a *lower bound* on the data collected by the campaign websites.

**Using VirusTotal:** VirusTotal has been widely used as the ground truth for malware classification by prior work[120][30][96][154][216]. That said, VirusTotal’s engines are not without flaws, particularly false positives due to over-approximation, and our findings obtained through them ( $\mathcal{F}_{12}$  and  $\mathcal{F}_{15}$ ) should be interpreted as *potential* problems in this context. Further, to counteract the infeasibility of validating the findings of VirusTotal’s scanners, prior work generally uses a threshold for trusting the VirusTotal labels [120][30]. Therefore, like prior work [120], we consider a threshold of 1 engine, but report the number of engines that label a website as malicious or phishing in our findings related to VirusTotal

(Section 5.8), to allow an informed interpretation of our findings, and convey the potential for false positives.

## **5.12 Chapter Summary**

This chapter presented the Polityzer framework to evaluate the privacy practices of political campaign websites, and the first large-scale study of this nature through an analysis of 2060 campaign websites involved in the 2020 US elections. Our analysis of the collection, privacy-disclosure, sharing, and general security risks associated with campaigns that collect highly-sensitive private data led to 20 key findings, which demonstrated critical insufficiencies in several aspects of privacy. Through conversations with campaign organizers, we further confirm the need for privacy standards that apply to political campaigns, which may motivate campaigns to improve. The datasets and results generated through this study will inform the development of such regulations and standards, while also enabling future research in this relatively understudied area.

## Chapter 6

# Future Work

This dissertation focuses on assessing the security and privacy risks of consumer-oriented software systems and developing practical frameworks that can aid in reducing such risks.

In chapter 3, to answer **RQ<sub>1</sub>** (*i.e.*, how threats arise), we performed a holistic evaluation of access control in platforms and exploitability in integrations that allowed us to perform an end-to-end lateral privilege escalation attack. Specifically, we performed a study of 2 popular data-store based smart home platforms *i.e.*, Philips Hue and Google Nest. From our analysis, we found that Hue’s permission enforcement and access control model is broken, and it allowed an adversary to arbitrarily modify privileged sections of the data store, thereby allowing apps to remove legitimate apps or add malicious apps without the user’s consent. On the other hand, we found that Google Nest’s app review process allowed many apps with dubious or misleading permission prompts to integrate with the platform, and making users vulnerable to overprivileged apps. Finally, we performed an end-to-end lateral privilege escalation attack by exploiting a routine through which we were able to compromise the security camera by first gaining access to a smart switch.

The key insights generated from chapter 3 allowed us answer **RQ<sub>2</sub>** (*i.e.*, developing practical defenses) in chapter 4, where we explore a practical framework, HomeEndorser, that provides the foundation of integrity guarantees in the smart home. Specifically, HomeEndorser leverages the information readily available from the devices present in the home

to validate whether incoming state change requests are consistent with the observations of local device, thereby guarding arbitrary or malicious modifications by integrations of protected entities called Abstract Home Objects (AHOs).

Finally, to answer **RQ<sub>3</sub>** (*i.e.*, how stakeholders convey risks), we develop a framework in Chapter 5, Polityzer, to perform a privacy posture analysis of election campaign websites in the US federal elections. Specifically, we analyzed the privacy-sensitive data types that are collected via campaign websites, and how comprehensively they disclose the privacy risks to the user through their privacy policies. The analysis led to 20 key findings, which revealed, among other things, that a large majority (70.78%) of campaigns do not provide a privacy policy at all, and even when they do, a significant portion (41.22%) do not fully disclose all private data they collect in their privacy policy. We further discuss the need for privacy regulations in this space that can potentially improve the status quo.

We now discuss research directions based off of Chapters 3→5 that we will explore in the future.

**Exploration of privacy posture of online election campaigns in other jurisdictions:** The findings from Chapter 5 revealed a significant gaps in the way current election campaigns handle the disclosure of their privacy practices to the consumers. However, while the federal election campaigns in the US provided an important data point, it is vitally important that we conduct further studies in other jurisdictions to gain localized insights into what privacy risks there are, and what solutions can best be explored in those jurisdictions for improvement. For instance, in the European Union, political data is protected under the General Data Protection Regulation (GDPR) [74], and election campaigns need to abide by special disclosure requirements [57]. However, it is unclear how well current election campaigns in the EU follow these disclosure requirements. An analysis of the EU election campaigns will also provide an important comparison with the US campaigns and the two system of regulations (or lack thereof) that exists between the two jurisdictions. Further, we will also analyze state-level and other local election campaigns

in the US, many of whom also rely on campaign websites as their primary interaction point with their constituencies.

**Policy-to-Behavior analysis:** An immediate next area of research stemming from this dissertation (in particular, Chapter 5) is the need to identify the conflicts between the actual behavior of the software artifacts with the privacy policies outlined in the stakeholder’s disclosure document. In particular, mapping the security and privacy requirements from privacy policies which are typically written in natural language (or in legalese) to the behavior of a software artifact remains an exciting but challenging research direction. The focus of this future work is to build security frameworks that can identify these conflicts by leveraging similar analysis and system design techniques we have used throughout this dissertation. In particular, such automation frameworks are not just beneficial to the consumers but are also extremely important to the software industry as well, as this will help companies remain compliant to their own privacy policies and avoid heavy fines levied by privacy regulations all over the world.

**Vulnerabilities in Software Supply Chain:** Throughout this dissertation, we explored many tenets of the software supply chain issue, not just in terms of software artifacts or dependencies (*e.g.*, IoT platform, third party integrations/apps, automation/routines) but also in terms of stakeholders (*e.g.* users, vendors, third party developers, online campaigns, fundraising platforms). As attacks on the software supply chain become even more prominent [146], it is vital to identify security vulnerabilities in other ecosystems (*e.g.*, the open-source software artifacts) and measuring their overall security and privacy impacts on the consumers. This dissertation provides a foundation on which further such analysis can be performed in the future.

## Chapter 7

# Conclusion

In this dissertation, our main objective is to understand the security and privacy threats that can arise in consumer-oriented software systems and understand their implications on the end-users. To achieve this goal, we analyze these systems under 3 aspects: i) investigate how threats can arise due to their various interdependent components, ii) design a practical solution by leveraging the abstractions provided by such systems, and iii) analyze how stakeholders convey these threats to the consumers.

In chapter 3, we analyze various components of two popular smart home platforms, Philips Hue and Google Nest, in chapter 2, including their permission enforcement mechanism, their app review process as well as the SSL problems in apps that have been approved to work with these platforms. Our analysis lead to 10 salient findings with various security implications, such as attackers being able to bypass user consent in Hue to the ineffective app review process leading to dubious apps being approved in Nest. Finally, we leveraged our findings to carry out, to our knowledge, the first end-to-end lateral privilege escalation attack in smart homes wherein we compromise a smart switch and use that access to affect the functionality of a security camera with the help of a platform-enabled routine.

Similarly, in chapter 4, we designed HomeEndorser, an endorsement framework that aims to provide integrity guarantees to supplement the permission model in platforms that will help the platform to better protect the modifications of Abstract Home Objects

(AHOs), which are the main components through which routines are characterized. Thus, HomeEndorser is able to minimize the effects of an attacker gaining improper access into a smart home by validating incoming AHO modifications, and thereby preventing security sensitive routines from being executed arbitrarily.

Further, in chapter 5, we designed Polityzer, a framework to systematically analyze the privacy postures of election campaign websites. Using Polityzer, we find a large majority of them lack a privacy disclosure. Even in cases where privacy policies were provided, they are often incomplete. We also found that campaigns may be inadvertently sharing data with other campaigns through common fundraising platforms, without disclosing such sharing. Finally, we discuss the potential future directions stemming from this dissertation in chapter 6.

# Bibliography

- [1] Actblue. <https://secure.actblue.com/>. Accessed August 2022.
- [2] Anedot | powerful giving tools made easy. <https://www.anedot.com/>. Accessed August 2022.
- [3] California consumer privacy act (ccpa). <https://www.oag.ca.gov/privacy/ccpa>. Accessed August 2022.
- [4] California privacy rights act (cpra). [https://leginfo.legislature.ca.gov/faces/codes\\_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5](https://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5). Accessed August 2022.
- [5] Directory | winred. <https://winred.com/>. Accessed August 2022.
- [6] Easylist - overview. <https://easylist.to/>. Accessed August 2022.
- [7] Free donate button - donorbox nonprofit fundraising software. <https://donorbox.org/>. Accessed August 2022.
- [8] How social media is shaping political campaigns. <https://knowledge.wharton.upenn.edu/article/how-social-media-is-shaping-political-campaigns/>. Accessed August 2022.

- [9] In hong kong, a proxy battle over internet freedom begins. <https://www.nytimes.com/2020/07/07/business/hong-kong-security-law-tech.html>. Accessed August 2022.
- [10] Privacy of no concern for ted cruz mobile app in campaign's massive data mining operation. <http://www.allgov.com/news/top-stories/privacy-of-no-concern-for-ted-cruz-mobile-app-in-campaigns-massive-data-news=858277>. Accessed August 2022.
- [11] Tech companies grapple with hong kong's new security law. <https://www.datacenterdynamics.com/en/news/tech-companies-grapple-hong-kongs-new-security-law/>. Accessed August 2022.
- [12] Threat analysis apis. <https://www.apivoid.com/>. Accessed August 2022.
- [13] Virustotal. <https://www.virustotal.com/gui/home/upload>. Accessed August 2022.
- [14] RICHMOND ALAKE. Understanding cosine similarity and its applications. <https://builtin.com/machine-learning/cosine-similarity>. Accessed June 2024.
- [15] ALARM GRID. Introducing the "Privacy When Disarmed" Feature for Total Connect 2.0 HD Cameras - Alarm Grid. <https://www.alarmgrid.com/blog/introducing-the-privacy-when-disarmed-feature-for-total-connect->, Accessed Feb 2021.
- [16] ALEXANDRA PARKER. Personal data leaked from fulton county, according to election officials. <https://www.atlantaneWSfirst.com/2022/09/23/personal-data-leaked-fulton-county-according-election-officials/>. Accessed Nov 2022.

- [17] BENJAMIN ANDOW, SAMIN YASEER MAHMUD, WENYU WANG, JUSTIN WHITAKER, WILLIAM ENCK, BRADLEY REAVES, KAPIL SINGH, AND TAO XIE. PolicyLint: Investigating Internal Privacy Policy Contradictions on Google Play. In *Proceedings of the USENIX Security Symposium*, 2019.
- [18] BENJAMIN ANDOW, SAMIN YASEER MAHMUD, JUSTIN WHITAKER, WILLIAM ENCK, BRADLEY REAVES, KAPIL SINGH, AND SERGE EGELMAN. Actions speak louder than words: Entity-sensitive privacy policy and data flow analysis with polichack. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 985–1002. USENIX Association, August 2020.
- [19] ANONYMOUS. HomeEndorser Online Appendix. <https://sites.google.com/view/homeendorser>, 2020.
- [20] BALLOTPEDIA. Special elections to the 116th united states congress (2019-2020). [https://ballotpedia.org/Special\\_elections\\_to\\_the\\_116th\\_United\\_States\\_Congress\\_\(2019-2020\)](https://ballotpedia.org/Special_elections_to_the_116th_United_States_Congress_(2019-2020)). Accessed August 2022.
- [21] BALLOTPEDIA. State legislators who have switched political party affiliation. [https://ballotpedia.org/State\\_legislators\\_who\\_have\\_switched\\_political\\_party\\_affiliation](https://ballotpedia.org/State_legislators_who_have_switched_political_party_affiliation). Accessed May 2023.
- [22] BALLOTPEDIA. What should you have on your campaign website? <https://ballotpedia.org/Ballotpedia:About>. Accessed August 2022.
- [23] BETTERCAP. BetterCAP stable documentation. <https://www.bettercap.org/legacy///>, Accessed June 2018.
- [24] JASPREET BHATIA, TRAVIS D. BREAU, JOEL R. REIDENBERG, AND THOMAS B. NORTON. A Theory of Vagueness and Privacy Risk Perception. In *Proceedings of the IEEE International Requirements Engineering Conference (RE)*, 2016.

- [25] K. J. BIBA. Integrity Considerations for Secure Computer Systems. Technical Report MTR-3153, MITRE, April 1977.
- [26] SIMON BIRNBACH, SIMON EBERZ, AND IVAN MARTINOVIC. Peeves: Physical event verification in smart homes. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1455–1467, 2019.
- [27] JASMINE BOWERS, BRADLEY REAVES, IMANI N. SHERMAN, PATRICK TRAYNOR, AND KEVIN BUTLER. Regulators, Mount Up! Analysis of Privacy Policies for Mobile Money Services. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS)*, 2017.
- [28] JASMINE BOWERS, IMANI N SHERMAN, KEVIN BUTLER, AND PATRICK TRAYNOR. Characterizing Security and Privacy Practices in Emerging Digital Credit Applications. In *Proceedings of the ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2019.
- [29] ALLISON J BROWN. “should i stay or should i leave?”: Exploring (dis) continued facebook use after the cambridge analytica scandal. *Social media+ society*, 6(1):2056305120913884, 2020.
- [30] ONUR CATAKOGLU, MARCO BALDUZZI, AND DAVIDE BALZAROTTI. Automatic extraction of indicators of compromise for web applications. In *Proceedings of the 25th international conference on world wide web*, pages 333–343, 2016.
- [31] CBS4 NEWS. How political campaigns are able to text you with personal information. <https://cbs4indy.com/news/how-political-campaigns-are-able-to-text-you-with-personal-information/>. Accessed May 2023.
- [32] ETHAN CECCHETTI, ANDREW C. MYERS, AND OWEN ARDEN. Nonmalleable in-

- formation flow control. In *Proceedings of the 2017 ACM Conference on Computer and Communications Security*, pages 1875–1891, 2017.
- [33] Z. BERKAY CELIK, PATRICK MCDANIEL, AND GANG TAN. Soteria: Automated IoT Safety and Security Analysis. In *2018 USENIX Annual Technical Conference (USENIX ATC)*, pages 147–158, 2018.
- [34] Z. BERKAY CELIK, GANG TAN, AND PATRICK MCDANIEL. IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT. In *Proceedings of the NDSS 2019 Symposium*, 2019.
- [35] ERIKA CHIN, ADRIENNE PORTER FELT, KATE GREENWOOD, AND D. WAGNER. Analyzing Inter-Application Communication in Android. In *Proceedings of the 9th Annual International Conference on Mobile Systems, Applications, and Services*, 2011.
- [36] CISCO. Building consumer confidence through transparency and control. [https://www.cisco.com/c/dam/en\\_us/about/doing\\_business/trust-center/docs/cisco-cybersecurity-series-2021-cps.pdf?CCID=cc000742&DTID=odicdc000016&OID=rptsc027438](https://www.cisco.com/c/dam/en_us/about/doing_business/trust-center/docs/cisco-cybersecurity-series-2021-cps.pdf?CCID=cc000742&DTID=odicdc000016&OID=rptsc027438). Accessed May 2023.
- [37] D. D. CLARK AND D. WILSON. A Comparison of Military and Commercial Security Policies. In *Proceedings IEEE Symposium on Security and Privacy*, May 1987.
- [38] CLEMFROMSPACE. scrapy-selenium. <https://github.com/clemfromspace/scrapy-selenium>. Accessed August 2022.
- [39] CNET. Google fined \$57 million under new european data privacy law. <https://www.cnet.com/tech/tech-industry/google-fined-57-million-under-european-privacy-law/>. Accessed May 2023.

- [40] CNET. Google reverses course on cutting off Works with Nest connections. <https://www.cnet.com/home/smart-home/google-reverses-course-on-cutting-off-works-with-nest-connections/>, Accessed May 2021.
- [41] CAMILLE COBB, MILIJANA SURBATOVICH, ANNA KAWAKAMI, MAHMOOD SHARIF, LUJO BAUER, ANUPAM DAS, AND LIMIN JIA. How Risky Are Real Users’{IFTTT} Applets? In *Sixteenth Symposium on Usable Privacy and Security ({SOUPS} 2020)*, pages 505–529, August 2020.
- [42] CONGRESSIONAL RESEARCH SERVICE. Franking privilege: Historical development and options for change. <https://crsreports.congress.gov/product/pdf/RL/RL34274/20>. Accessed May 2023.
- [43] JOAN L CONNERS. Social media use in us senate campaigns: Initial tactics with twitter. *Social Media and Politics: A New Way to Participate in the Political Process [2 volumes]*, page 129, 2016.
- [44] SUNNY CONSOLVO, PATRICK KELLEY, TARA MATTHEWS, KURT THOMAS, LEE DUNN, AND ELIE BURSZTEIN. "why wouldn't someone think of democracy as a target?": Security practices & challenges of people involved with u.s. political campaigns. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, August 2021.
- [45] CPO MAGAZINE. Lessons learned from gdpr fines in 2023. <https://www.cpomagazine.com/data-protection/lessons-learned-from-gdpr-fines-in-2023/>. Accessed May 2023.
- [46] LORRIE FAITH CRANOR, PEDRO GIOVANNI LEON, AND BLASE UR. A Large-Scale Evaluation of US Financial Institutions’ Standardized Privacy Notices. *ACM Transactions on the Web*, 2016.

- [47] RY CRIST. A smart home divided: Can it stand? <https://www.cnet.com/news/a-smart-home-divided-can-it-stand/>, Accessed September 2018.
- [48] MARTIN DEGELING, CHRISTINE UTZ, CHRISTOPHER LENTZSCH, HENRY HOSSEINI, FLORIAN SCHAUB, AND THORSTEN HOLZ. We Value Your Privacy... Now Take Some Cookies: Measuring the GDPR's Impact on Web Privacy. In *Proceedings of the ISOC Network and Distributed Systems Symposium (NDSS)*, 2018.
- [49] DIGITAL TRENDS. Google is ending its Works with Nest system. Here's what that means for you . <https://www.digitaltrends.com/home/google-is-ending-its-works-with-nest-system/>, Accessed May 2021.
- [50] DIGITIZED HOUSE. Fallout Mounts from Impending Works with Nest Sunsetting. <https://digitized.house/fallout-mounts-works-with-nest-sunsetting/>, Accessed May 2021.
- [51] WENBO DING AND HONGXIN HU. On the Safety of IoT Device Physical Interaction Control. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 832–846, October 2018.
- [52] WENBO DING, HONGXIN HU, AND LONG CHENG. IOTS SAFE: Enforcing Safety and Security Policy with Real IoT Physical Interaction Discovery. In *Proceedings of the ISOC Network and Distributed Systems Security Symposium (NDSS)*, February 2021. To Appear.
- [53] KATHARINE DOMMETT. Regulating digital campaigning: the need for precision in calls for transparency. *Policy & Internet*, 12(4):432–449, 2020.
- [54] EFF. Voter privacy: What you need to know about your digital trail during the 2016 election. <https://www.eff.org/deeplinks/2016/02/voter-privacy-what-you-need-know-about-your-digital-trail-during-2016-el> Accessed May 2023.

- [55] PETROS EFSTATHOPOULOS, MAXWELL KROHN, STEVE VANDEBOGART, CLIFF FREY, DAVID ZIEGLER, EDDIE KOHLER, DAVID MAZIERES, FRANS KAASHOEK, AND ROBERT MORRIS. Labels and event processes in the Asbestos operating system. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles*, volume 39, pages 17–30, 2005.
- [56] WILLIAM ENCK, PETER GILBERT, BYUNG-GON CHUN, LANDON P. COX, JAEYEON JUNG, PATRICK MCDANIEL, AND ANMOL N. SHETH. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, October 2010.
- [57] ENCOMPASS. The effect of gdpr on the political industry. <https://encompass-europe.com/comment/the-effect-of-gdpr-on-the-political-industry>. Accessed May 2023.
- [58] ENGADGET. SmartThings shows off the ridiculous possibilities of its connected home system. <https://www.engadget.com/2014/01/11/smartthings-labs/>, Accessed June 2018.
- [59] STEVEN ENGLEHARDT AND ARVIND NARAYANAN. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1388–1401, 2016.
- [60] SASCHA FAHL, MARIAN HARBACH, THOMAS MUDERS, LARS BAUMGÄRTNER, BERND FREISLEBEN, AND MATTHEW SMITH. Why eve and mallory love android: An analysis of android ssl (in)security. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, 2012.
- [61] FEC. Browse data. <https://www.fec.gov/data/browse-data/?tab=candidates>. Accessed August 2022.

- [62] FEC. Recording receipts. <https://www.fec.gov/help-candidates-and-committees/keeping-records/recording-receipts/>. Accessed August 2022.
- [63] FEC. Registering as a candidate. <https://www.fec.gov/help-candidates-and-committees/registering-candidate/>. Accessed August 2022.
- [64] ADRIENNE PORTER FELT, ERIKA CHIN, STEVE HANNA, DAWN SONG, AND DAVID WAGNER. Android Permissions Demystified. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2011.
- [65] ADRIENNE PORTER FELT, HELEN J. WANG, ALEXANDER MOSHCHUK, STEVEN HANNA, AND ERIKA CHIN. Permission Re-Delegation: Attacks and Defenses. In *Proceedings of the USENIX Security Symposium*, August 2011.
- [66] EARLENCE FERNANDES, JAEYEON JUNG, AND ATUL PRAKASH. Security analysis of emerging smart home applications. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 636–654, May 2016.
- [67] CENTER FOR DEMOCRACY AND TECHNOLOGY. Election cybersecurity 101 field guides. <https://cdt.org/area-of-focus/cybersecurity-standards/election-security/>. Accessed August 2022.
- [68] BELFER CENTER FOR SCIENCE AND INTERNATIONAL AFFAIRS. Cybersecurity campaign playbook. <https://www.belfercenter.org/publication/cybersecurity-campaign-playbook>. Accessed August 2022.
- [69] TIMOTHY FRASER, LEE BADGER, AND MARK FELDMAN. Hardening COTS Software with Generic Software Wrappers. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 2–16, 1999.

- [70] GALLUP. Partisan differences growing on a number of issues. <https://news.gallup.com/opinion/polling-matters/215210/partisan-differences-growing-number-issues.aspx>. Accessed Nov 2022.
- [71] GARTNER. Gartner Says 8.4 Billion Connected Things Will Be in Use in 2017, Up 31 Percent From 2016. <https://www.gartner.com/newsroom/id/3598917>, Accessed June 2018.
- [72] MANISH GAURAV, AMIT SRIVASTAVA, ANOOP KUMAR, AND SCOTT MILLER. Leveraging candidate popularity on twitter to predict election outcome. SNAKDD '13, New York, NY, USA, 2013. Association for Computing Machinery.
- [73] GDPR EU. What are the 7 main principles of gdpr? <https://www.gdpreu.org/7-main-data-protection-principles-under-gdpr/>. Accessed May 2023.
- [74] GDPR-INFO. Art. 9 gdpr - processing of special categories of personal data. <https://gdpr-info.eu/art-9-gdpr/>. Accessed May 2023.
- [75] ROBERTO J. GONZÁLEZ. Hacking the citizenry?: Personality profiling, 'big data' and the election of donald trump. *Anthropology Today*, 33(3):9–12, 2017.
- [76] GOOGLE. Nest x Yale Lock. [https://store.google.com/us/product/nest\\_x\\_yale\\_lock](https://store.google.com/us/product/nest_x_yale_lock), 2020.
- [77] GOOGLE. Local Home SDK. <https://developers.google.com/actions/smarthome/local-home-sdk>, Accessed June 2019.
- [78] GOOGLE. Google Nest: welcome to the helpful home. <https://blog.google/products/google-nest/helpful-home/>, Accessed May 2021.
- [79] GOOGLE. We hear you: updates to Works with Nest. <https://blog.google/products/google-nest/updates-works-with-nest/>, Accessed May 2021.

- [80] GOOGLE PLAY. Kasa App. [https://play.google.com/store/apps/details?id=com.tplink.kasa\\_android](https://play.google.com/store/apps/details?id=com.tplink.kasa_android), Accessed June 2018.
- [81] GOOGLE PLAY. Keen Home. <https://play.google.com/store/apps/details?id=com.hipo.keen//>, Accessed June 2018.
- [82] GOOGLE PLAY. Nest. <https://play.google.com/store/apps/details?id=com.nest.android>, Accessed June 2018.
- [83] GOOGLE PLAY. Philips Hue. <https://play.google.com/store/apps/details?id=com.philips.lighting.hue2>, Accessed June 2018.
- [84] GOOGLE PLAY. WeMo. <https://play.google.com/store/apps/details?id=com.belkin.wemoandroid>, Accessed June 2018.
- [85] GOVERNMENT TECHNOLOGY. Voter data modeling: Does it threaten our privacy? <https://www.govtech.com/data/voter-data-modeling-does-it-threaten-our-privacy.html>. Accessed May 2023.
- [86] YASSINE HAMDAOUI. A guide to tf-idf. <https://builtin.com/articles/tf-idf>. Accessed June 2024.
- [87] CATHERINE HAN, IRWIN REYES, ÁLVARO FEAL, JOEL REARDON, PRIMAL WIJESEKERA, NARSEO VALLINA-RODRIGUEZ, AMIT ELAZARI, KENNETH A. BAMBERGER, AND SERGE EGELMAN. The Price is (Not) Right: Comparing Privacy in Free and Paid Apps. In *Proceedings on Privacy Enhancing Technologies (PETS)*, 2020.
- [88] HAMZA HARKOUS, KASSEM FAWAZ, RÉMI LEBRET, FLORIAN SCHAUB, KANG G. SHIN, AND KARL ABERER. Polisis: Automated analysis and presentation of privacy policies using deep learning. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 531–548, Baltimore, MD, August 2018. USENIX Association.

- [89] HAMZA HARKOUS, KASSEM FAWAZ, RÉMI LEBRET, FLORIAN SCHAUB, KANG G. SHIN, AND KARL ABERER. Polis: Automated Analysis and Presentation of Privacy Policies Using Deep Learning. In *Proceedings of the USENIX Security Symposium*, 2018.
- [90] HELION. Helion Dataset. [https://github.com/helion-security/helion/tree/master/data/generated\\_data/policy/predictions\\_for\\_policy](https://github.com/helion-security/helion/tree/master/data/generated_data/policy/predictions_for_policy), Accessed September 2021.
- [91] STEPHAN HEUSER, ADWAIT NADKARNI, WILLIAM ENCK, AND AHMAD-REZA SADEGHI. ASM: A Programmable Interface for Extending Android Security. In *Proceedings of the USENIX Security Symposium*, August 2014.
- [92] JOANNE HINDS, EMMA J WILLIAMS, AND ADAM N JOINSON. "it wouldn't happen to me": Privacy concerns and perspectives following the cambridge analytica scandal. *International Journal of Human-Computer Studies*, 143:102498, 2020.
- [93] HOME ASSISTANT. <https://www.home-assistant.io>, Accessed July 2019.
- [94] HOME ASSISTANT AUTOMATIONS. Cookbook. <https://www.home-assistant.io/cookbook/>, Accessed July 2019.
- [95] HOMEKIT. Homekit helps to control devices using Home App. <https://www.apple.com/ios/home/>, Accessed July 2019.
- [96] GENG HONG, ZHEMIN YANG, SEN YANG, LEI ZHANG, YUHONG NAN, ZHIBO ZHANG, MIN YANG, YUAN ZHANG, ZHIYUN QIAN, AND HAIXIN DUAN. How you get shot in the back: A systematical study about cryptojacking in the real world. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1701–1713, 2018.
- [97] HOW TO GEEK. How to Use Your Nest Thermostat as a Motion Detector. <https://www.howtogeek.com/249093/>

- how-to-use-your-nest-thermostat-as-a-motion-detector/, Accessed May 2021.
- [98] CHRISTOPHER HUNTER. Political privacy and online politics: How e-campaigning threatens voter privacy. *First Monday*, 7(2), Feb. 2002.
- [99] IAPP. Closing in on the us election with voter privacy and election security. <https://iapp.org/news/a/closing-in-on-the-u-s-election-with-voter-privacy-and-election-security/> Accessed May 2023.
- [100] IBM. What is the Internet of Things (IoT)? <https://www.ibm.com/topics/internet-of-things>, Accessed June 2024.
- [101] IFTTT. IFTTT helps your apps and devices work together. <https://ifttt.com/>, Accessed June 2018.
- [102] IFTTT. Important update about the Nest services . <https://help.ifttt.com/hc/en-us/articles/360022524734-Important-update-about-the-Nest-services>, Accessed May 2021.
- [103] INFORMATION COMMISSIONER'S OFFICE. Guidance for the use of poersonal data in political campaigning. <https://ico.org.uk/for-organisations/direct-marketing-and-privacy-and-electronic-communications/guidance-for-the-use-of-personal-data-in-political-campaigning-1/>. Accessed May 2023.
- [104] ELECTION CYBERSECURITY INITIATIVE. Use protect elections - our candidate is democracy. <https://www.electionsecurity.usc.edu>. Accessed August 2022.

- [105] INTERNET SOCIETY. Online trust alliance (ota). <https://www.internetsociety.org/ota/>. Accessed May 2023.
- [106] INTERNET SOCIETY. Online trust audit – 2020 u.s. presidential campaigns. <https://www.internetsociety.org/resources/ota/2019/online-trust-audit-2020-u-s-presidential-campaigns/>. Accessed Nov 2022.
- [107] IOTIVITY. IoTivity Wiki: IoTivity Initialization and Setting. [https://wiki.iotivity.org/initialize\\_setting](https://wiki.iotivity.org/initialize_setting), Accessed June 2019.
- [108] JIM ISAAK AND MINA J. HANNA. User data privacy: Facebook, cambridge analytica, and privacy protection. *Computer*, 51(8):56–59, 2018.
- [109] CARLOS JENSEN AND COLIN POTTS. Privacy Policies as Decision-Making Tools: An Evaluation of Online Privacy Notices. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems (CHI)*, 2004.
- [110] YUNHAN JACK JIA, QI ALFRED CHEN, SHIQI WANG, AMIR RAHMATI, EARLENCE FERNANDES, Z MORLEY MAO, ATUL PRAKASH, AND SHANGHAI JIAO-TONG UNVIERSITY. ContextIoT: Towards providing contextual integrity to appified IoT platforms. In *Proceedings of the 2017 Network and Distributed System Security Symposium (NDSS)*, 2017.
- [111] ARUN CYRIL JOSE AND REZA MALEKIAN. Improving smart home security: Integrating logical sensing into smart home. *IEEE Sensors Journal*, 17(13):4269–4286, 2017.
- [112] KAUSHAL KAFLE, KEVIN MORAN, SUNIL MANANDHAR, ADWAIT NADKARNI, AND DENYS POSHYVANYK. A Study of Data Store-based Home Automation. In *Proceedings of the 9th ACM Conference on Data and Application Security and Privacy (CODASPY)*, March 2019.

- [113] KAUSHAL KAFLE, KEVIN MORAN, SUNIL MANANDHAR, ADWAIT NADKARNI, AND DENYS POSHYVANYK. Security in Centralized Data Store-based Home Automation Platforms: A Systematic Analysis of Nest and Hue. *ACM Transactions on Cyber-Physical Systems (TCPS)*, 5(1), December 2020.
- [114] KAMI. How to use Home and Away mode in YI Home app . <https://help.yitechnology.com/hc/en-us/articles/360041767214-How-to-use-Home-and-Away-mode-in-YI-Home-app>, Accessed May 2021.
- [115] DANIEL KREISS AND PHILIP N HOWARD. New challenges to political privacy: Lessons from the first us presidential race in the web 2.0 era. *International Journal of Communication*, 4:19, 2010.
- [116] MAXWELL KROHN, ALEXANDER YIP, MICAH BRODSKY, NATAN CLIFFER, M. FRANS KAASHOEK, EDDIE KOHLER, AND ROBERT MORRIS. Information Flow Control for Standard OS Abstractions. In *Proceedings of ACM Symposium on Operating Systems Principles (SOSP)*, pages 321–334, October 2007.
- [117] LAMBDATEST. Different Types of Mobile Apps: A Beginner’s Guide. <https://www.lambdatest.com/blog/types-of-mobile-apps/>, Accessed June 2024.
- [118] CARL E. LANDWEHR, ALAN R. BULL, JOHN P. MCDERMOTT, AND WILLIAM S. CHOI. A Taxonomy of Computer Program Security Flaws. *ACM Computing Surveys (CSUR)*, 26(3), September 1994.
- [119] SANGHAK LEE, JIWON CHOI, JIHUN KIM, BEUMJIN CHO, SANGHO LEE, HANJUN KIM, AND JONG KIM. FACT: Functionality-centric access control system for IoT programming frameworks. In *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies*, pages 43–54, 2017.

- [120] LI LI, DAOYUAN LI, TEGAWENDÉ F BISSYANDÉ, JACQUES KLEIN, YVES LE TRAON, DAVID LO, AND LORENZO CAVALLARO. Understanding android app piggybacking: A systematic study of malicious code grafting. *IEEE Transactions on Information Forensics and Security*, 12(6):1269–1284, 2017.
- [121] THOMAS LINDEN, RISHABH KHANDELWAL, HAMZA HARKOUS, AND KASSEM FAWAZ. The privacy policy landscape after the gdpr. *Proceedings on Privacy Enhancing Technologies*, 2020(1):47–64, 2020.
- [122] MADISON TROYER. How political campaigning has changed throughout US history. <https://stacker.com/politics/how-political-campaigning-has-changed-throughout-us-history>, Accessed June 2024.
- [123] TARIQ MAHMOOD, TASMIYAH IQBAL, FARNAZ AMIN, WAJEETA LOHANNA, AND ATIKA MUSTAFA. Mining twitter big data to predict 2013 pakistan election winner. In *INMIC*, pages 49–54. IEEE, 2013.
- [124] SUNIL MANANDHAR, KEVIN MORAN, KAUSHAL KAFLE, RUHAO TANG, DENYS POSHYVANYK, AND ADWAIT NADKARNI. Towards a Natural Perspective of Smart Homes for Practical Security and Safety Analyses. In *Proceedings of the 41st IEEE Symposium on Security and Privacy (Oakland)*, San Fransisco, CA, USA, May 2020.
- [125] MASTERCLASS. A guide to the primary election cycle in the united states. <https://www.masterclass.com/articles/a-guide-to-the-primary-election-cycle-in-the-united-states#how-do-primary-elections-work>. Accessed August 2022.
- [126] ARUNESH MATHUR, ANGELINA WANG, CARSTEN SCHWEMMER, MAIA HAMIN, BRANDON M STEWART, AND ARVIND NARAYANAN. Manipulative tactics are the norm in political emails. 2022.

- [127] MATOMO. Google analytics privacy issues: Is it really that bad? <https://matomo.org/blog/2022/06/google-analytics-privacy-issues/>. Accessed Nov 2022.
- [128] ALEECIA M. McDONALD AND LORRIE FAITH CRANOR. The Cost of Reading Privacy Policies. *I/S Journal of Law and Policy for the Information Society (ISJLP)*, 4, 2008.
- [129] M. DOUGLAS MCILROY AND JAMES A. REEDS. Multilevel security in the UNIX tradition. *Software: Practice and Experience*, 1992.
- [130] MCKINSEY & COMPANY. The consumer-data opportunity and the privacy imperative. <https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/the-consumer-data-opportunity-and-the-privacy-imperative>. Accessed May 2023.
- [131] ADWAIT NADKARNI, BENJAMIN ANDOW, WILLIAM ENCK, AND SOMESH JHA. Practical DIFC Enforcement on Android. In *Proceedings of the 25th USENIX Security Symposium*, August 2016.
- [132] ADWAIT NADKARNI AND WILLIAM ENCK. Preventing Accidental Data Disclosure in Modern Operating Systems. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, November 2013.
- [133] NATIONAL ARCHIVES. Code of federal regulations. <https://www.ecfr.gov/current/title-45/subtitle-A/subchapter-A/part-46/subpart-A/section-46.102>. Accessed Nov 2022.
- [134] NEST. Nest API Explorer. <https://developers.nest.com/documentation/api-reference>, 2020.

- [135] NEST DEVELOPERS. How to Choose Permissions and Write Descriptions. <https://developers.nest.com/documentation/cloud/permissions-overview>, Accessed June 2018.
- [136] NEST DEVELOPERS. OAuth 2.0 Authentication and Authorization. <https://developers.nest.com/documentation/cloud/how-to-auth>, Accessed June 2018.
- [137] NEST DEVELOPERS. Product Review Guidelines. <https://developers.nest.com/documentation/cloud/product-review-guidelines>, Accessed June 2018.
- [138] NEST LABS. Meet the Nest app. <https://nest.com/app/>, Accessed Feb 2019.
- [139] NEST LABS. Turn your Nest camera on or off - Google Nest Help. <https://bit.ly/331kfPE>, Accessed Feb 2021.
- [140] NEST LABS. Kasa - Works with Nest Store. <https://workswith.nest.com/company/tp-link-research-america-corp/kasa>, Accessed June 2018.
- [141] NEST LABS. Nest Developers. <https://developers.nest.com///>, Accessed June 2018.
- [142] NEST LABS. Nest Simulator. <https://developers.nest.com/documentation/cloud/home-simulator>, Accessed June 2018.
- [143] NEST LABS. Works with Nest. <https://nest.com/works-with-nest//>, Accessed June 2018.
- [144] DANG TU NGUYEN, CHENGYU SONG, ZHIYUN QIAN, SRIKANTH V. KRISHNAMURTHY, EDWARD J. M. COLBERT, AND PATRICK MCDANIEL. IotSan: Fortifying the Safety of IoT Systems. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, pages 191–203, 2018.

- [145] NIQ. Consumer Data - Unveiling the Power of Market Insights. <https://nielseniq.com/global/en/info/consumer-data/>, Accessed June 2024.
- [146] NIST. Software security in supply chains. <https://www.nist.gov/itl/executive-order-14028-improving-nations-cybersecurity/software-supply-chain-security-guidance>. Accessed June 2024.
- [147] SUKHVIR NOTRA, MUHAMMAD SIDDIQI, HASSAN HABIBI GHARAKHEILI, VIJAY SIVARAMAN, AND ROKSANA BORELI. An experimental study of security and privacy risks with emerging household appliances. In *Proceedings of the 2014 IEEE Conference on Communications and Network Security (CNS)*, pages 79–84, 2014.
- [148] OCF. Open Connectivity Foundation - OCF. <https://openconnectivityfoundation.github.io/devicemodels/docs/index.html>, 2020.
- [149] OCF. Open Connectivity Foundation Github. <https://github.com/openconnectivityfoundation/devicemodels>, 2020.
- [150] FEDERAL BUREAU OF INVESTIGATION. Protected voices. <https://www.fbi.gov/investigate/counterintelligence/foreign-influence/protected-voices>. Accessed August 2022.
- [151] EHIMARE OKOYOMON, NIKITA SAMARIN, PRIMAL WIJESKERA, AMIT ELAZARI BAR ON, NARSEO VALLINA-RODRIGUEZ, IRWIN REYES, ÁLVARO FEAL, AND SERGE EGELMAN. On the Ridiculousness of Notice and Consent: Contradictions in App Privacy Policies. In *Workshop on Technology and Consumer Protection (ConPro)*, 2019.
- [152] LUCKY ONWUZURIKE AND EMILIANO DECRISTOFARO. Danger is my middle name: experimenting with SSL vulnerabilities in Android apps. In *Proceedings of the 8th*

- ACM Conference on Security & Privacy in Wireless & Mobile Networks*, page 15, 2015.
- [153] OPENHAB. OpenHAB: Empowering the Smart Home. <https://www.openhab.org>, Accessed September 2020.
- [154] FEARGUS PENDLEBURY, FABIO PIERAZZI, ROBERTO JORDANEY, JOHANNES KINDER, AND LORENZO CAVALLARO. {TESSERACT}: Eliminating experimental bias in malware classification across space and time. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 729–746, 2019.
- [155] ANDREW PERRIN. Americans are changing their relationship with facebook. *Pew Research Center*, 5, 2018.
- [156] GIUSEPPE PETRACCA, YUQIONG SUN ANDN AHMAD-ATAMLI REINEH, JENS GROSSKLAGS, PATRICK MCDANIEL, AND TRENT JAEGER. Entrust: Regulating sensor access by cooperating programs via delegation graphs. In *Proceedings of the 28th USENIX Security Symposium*, August 2019.
- [157] GIUSEPPE PETRACCA, AHMAD-ATAMLI REINEH, YUQIONG SUN, JENS GROSSKLAGS, AND TRENT JAEGER. Aware: Preventing abuse of privacy-sensitive sensors via operation bindings. In *Proceedings of the 26th USENIX Security Symposium*, August 2017.
- [158] PEW RESEARCH. Voters rarely switch parties, but recent shifts further educational, racial divergence. <https://www.pewresearch.org/politics/2020/08/04/voters-rarely-switch-parties-but-recent-shifts-further-educational-racial> Accessed May 2023.
- [159] PHILIPS. Philips Hue: Personal Wireless Lighting. <https://www2.meethue.com/en-us/about-hue>, Accessed June 2018.

- [160] PHILIPS HUE DEVELOPERS. Philips hue API. <https://developers.meethue.com/philips-hue-api>, Accessed June 2018.
- [161] CALLUM PILTON, SHAMAL FAILY, AND JANE HENRIKSEN-BULMER. Evaluating privacy-determining user privacy expectations on the web. *computers & security*, 105:102241, 2021.
- [162] DAVE PISCITELLO. What is Privilege Escalation. <https://www.icann.org/news/blog/what-is-privilege-escalation>, 2016.
- [163] ANDREW PODOB, BENJAMIN W CAMPBELL, AND JANET M BOX-STEFFENSMEIER. Collaboration among congressional campaigns: The sharing of donor and supporter information. In *Political Networks Workshops & Conference*, 2018.
- [164] POLITICO. Inside the 2016 black market for donor emails. <https://www.politico.com/story/2015/12/inside-the-2016-black-market-for-donor-emails-216761>. Accessed May 2023.
- [165] POLITYZER. Polityzer data and code. <https://github.com/polityzer/polityzer>. Accessed January 2022.
- [166] POPULAR SCIENCE. Stop shouting at your smart home so much and set up multi-step routines. <https://www.popsci.com/smart-home-routines-apple-google-amazon/>, Accessed June 2018.
- [167] PROTON BLOG. Political campaigns and your personal data. <https://proton.me/blog/political-campaigns-and-your-personal-data>. Accessed May 2023.

- [168] YU PU AND JENS GROSSKLAGS. Valuating {Friends’} privacy: Does anonymity of sharing personal data matter? In *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*, pages 339–355, 2017.
- [169] PULITZER CENTER. Consumer data privacy in politics. <https://pulitzercenter.org/stories/consumer-data-privacy-politics>. Accessed May 2023.
- [170] EMILEE RADER. Awareness of behavioral tracking and information privacy concern in facebook and google. In *10th Symposium On Usable Privacy and Security (SOUPS 2014)*, pages 51–67, 2014.
- [171] AMIR RAHMATI, EARLENCE FERNANDES, KEVIN EYKHOLT, AND ATUL PRAKASH. Tyche: A Risk-Based Permission Model for Smart Homes. In *2018 IEEE Cybersecurity Development (SecDev)*, pages 29–36, 2018.
- [172] BRADLEY REAVES, NOLEN SCAIFE, ADAM BATES, PATRICK TRAYNOR, AND KEVIN R.B. BUTLER. Mo(bile) money, mo(bile) problems: Analysis of branchless banking applications in the developing world. In *Proceedings of the 24th USENIX Security Symposium (USENIX Security 15)*, pages 17–32, 2015.
- [173] REUTERS. How political campaigns use your data. <https://graphics.reuters.com/USA-ELECTION/DATA-VISUAL/yxmvjjgojvr/>. Accessed August 2022.
- [174] RING. Control All your Ring Cameras with Modes - Ring Help. <https://support.ring.com/hc/en-us/articles/360036107792-Control-All-your-Ring-Cameras-with-Modes>, Accessed Feb 2021.
- [175] TALIA RINGER, DAN GROSSMAN, AND FRANZISKA ROESNER. AUDACIOUS: user-

- driven access control with unmodified operating systems. In *Proceedings of the 2016 ACM Conference on Computer and Communications Security*, 2016.
- [176] FRANZISKA ROESNER, TADAYOSHI KOHNO, ALEXANDER MOSHCHUK, BRYAN PARNO, HELEN J. WANG, AND CRISPIN COWAN. User-Driven Access Control: Rethinking Permission Granting in Modern Operating Systems. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2012.
- [177] IRA RUBINSTEIN. Voter privacy in the age of big data. *SSRN Electronic Journal*, 2014, 01 2014.
- [178] SAMSUNG. Samsung SmartThings SmartApp Public Repository. <https://github.com/SmartThingsCommunity/SmartThingsPublic>, 2018.
- [179] KANTHASHREE MYSORE SATHYENDRA, SHOMIR WILSON, FLORIAN SCHAUB, SEBASTIAN ZIMMECK, AND NORMAN SADEH. Identifying the Provision of Choices in Privacy Policy Text. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.
- [180] CHRISTOPHE OLIVIER SCHNEBLE, BERNICE SIMONE ELGER, AND DAVID SHAW. The cambridge analytica affair and internet-mediated research. *EMBO reports*, 19(8):e46579, 2018.
- [181] ROEI SCHUSTER, VITALY SHMATIKOV, AND ERAN TROMER. Situational access control in the internet of things. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1056–1073, 2018.
- [182] UMESH SHANKAR, TRENT JAEGER, AND REINER SAILER. Toward Automated Information-Flow Integrity Verification for Security-Critical Applications. In *Proceedings of the ISOC Network and Distributed Systems Security Symposium (NDSS)*, 2006.

- [183] AMIT KUMAR SIKDER, GIUSEPPE PETRACCA, HIDAYET AKSU, TRENT JAEGER, AND A. SELCUK ULUAGAC. A Survey on Sensor-based Threats and Attacks to Smart Devices and Applications. *IEEE Communications Surveys and Tutorials*, 2021.
- [184] SIMPLISAFE. What is the privacy shutter? How does it work? <https://support.simplisafe.com/hc/en-us/articles/360029760591>, Accessed Feb 2021.
- [185] SIMPLISAFE. Everything You Need to Know About Home Mode. <https://simplisafe.com/blog/home-mode>, Accessed May 2021.
- [186] PRABHSIMRAN SINGH, YOGESH K DWIVEDI, KARANJEET SINGH KAHLON, ANNIE PATHANIA, AND RAVINDER SINGH SAWHNEY. Can twitter analytics predict election outcome? an insight from 2017 punjab assembly elections. *Government Information Quarterly*, 37(2):101444, 2020.
- [187] VIJAY SIVARAMAN, DOMINIC CHAN, DYLAN EARL, AND ROKSANA BORELI. Smart-phones attacking smart-homes. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 195–200. ACM, 2016.
- [188] SLICKTEXT. One year after cambridge analytica, survey reveals strong consumer privacy fears remain. <https://www.slicktext.com/blog/2019/02/survey-consumer-privacy-fears-after-cambridge-analytica/>. Accessed April 2022.
- [189] SMARTTHINGS DEVELOPER DOCUMENTATION. Capabilities Reference. <https://docs.smarthings.com/en/latest/capabilities-reference.html>, Accessed December 2018.
- [190] SMARTTHINGS DEVELOPERS. Hosting with AWS Lambda. <https://smarthings.developer.samsung.com/docs/guides/smartapps/aws-lambda.html>, Accessed July 2019.

- [191] SMARTTHINGS DEVELOPERS. Self-hosting (WebHook). <https://smarththings.developer.samsung.com/docs/guides/smartapps/webhook-apps.html>, Accessed July 2019.
- [192] SMARTTHINGS DEVELOPERS. Documentation. <http://developer.smarththings.com/>, Accessed June 2018.
- [193] SMARTTHINGS DEVELOPERS. The SmartThings Ecosystem. <https://smarththings.developer.samsung.com/docs/index.html>, Accessed June 2019.
- [194] SMARTTHINGS SUPPORT. Routines in the SmartThings Classic app. <https://support.smarththings.com/hc/en-us/articles/205380034-Routines-in-the-SmartThings-Classic-app>, Accessed June 2018.
- [195] DAVID SOUNTHIRARAJ, JUSTIN SAHS, GARRET GREENWOOD, ZHIQIANG LIN, AND LATIFUR KHAN. Smv-hunter: Large scale, automated detection of ssl/tls man-in-the-middle vulnerabilities in android apps. In *In Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS'14)*, 2014.
- [196] DAVID SOUNTHIRARAJ, JUSTIN SAHS, ZHIQIANG LIN, LATIFUR KHAN, AND GARRETT GREENWOOD. SMV-Hunter: Large Scale, Automated Detection of SSL/TLS Man-in-the-Middle Vulnerabilities in Android Apps. In *Proceedings of the ISOC Network and Distributed Systems Symposium (NDSS)*, February 2014.
- [197] SPEAKEASY POLITICAL. Political digital fact of the day: The importance of your campaign's website. <https://www.speakeasypolitical.com/12-facts-political-campaign-website/>. Accessed May 2023.
- [198] STATISTA. Revenue of the smart home market in the united states from 2019 to 2028. <https://www.statista.com/forecasts/887552/>

- revenue-in-the-smart-home-market-in-the-united-states. Accessed June 2024.
- [199] STATISTA. Smart Home - Worldwide. <https://www.statista.com/outlook/279/100/smart-home/worldwide>, 2020.
- [200] STRINGIFY. Stringify | Change Your Life by Connecting Everything. <https://www.stringify.com//>, Accessed June 2018.
- [201] MILIJANA SURBATOVICH, JASSIM ALJUR AidAN, LUJO BAUER, ANUPAM DAS, AND LIMIN JIA. Some Recipes Can Do More Than Spoil Your Appetite: Analyzing the Security and Privacy Risks of IFTTT Recipes. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1501–1510, April 2017.
- [202] TACTICAL TECH. Personal data: Political persuasion. [https://cdn.ttc.io/s/tacticaltech.org/Personal-Data-Political-Persuasion-How-it-works\\_print-friendly.pdf](https://cdn.ttc.io/s/tacticaltech.org/Personal-Data-Political-Persuasion-How-it-works_print-friendly.pdf). Accessed Nov 2022.
- [203] TECHCRUNCH. Donor data is the new currency of presidential candidates. <https://techcrunch.com/2016/01/18/donor-data-is-the-new-currency-of-presidential-candidates/>. Accessed May 2023.
- [204] TECHCRUNCH. Google Assistant is adding Routines and location-based reminders. <https://techcrunch.com/2018/02/23/google-assistant-is-adding-routines-and-location-based-reminders/>, Accessed June 2018.
- [205] THE AMBIENT. Life360 guide: How to smarten up your home with the geolocation app. <https://www.the-ambient.com/guides/life360-complete-guide-804>, Accessed May 2021.

- [206] THE GUARDIAN. 'i made steve bannon's psychological warfare tool': meet the data war whistleblower. <https://www.theguardian.com/news/2018/mar/17/data-war-whistleblower-christopher-wylie-faceook-nix-bannon-trump>. Accessed May 2023.
- [207] THE NATIONAL ARCHIVES. Data protection act 1998. <https://www.legislation.gov.uk/ukpga/1998/29>. Accessed May 2023.
- [208] THE NEW YORK TIMES. Ted cruz's new data strategy: Here comes santa claus. [https://archive.nytimes.com/www.nytimes.com/politics/first-draft/2015/12/17/ted-cruzs-new-data-strategy-here-comes-santa-claus/?\\_r=0](https://archive.nytimes.com/www.nytimes.com/politics/first-draft/2015/12/17/ted-cruzs-new-data-strategy-here-comes-santa-claus/?_r=0). Accessed May 2023.
- [209] THE VERGE. You can soon activate multi-step routines in Alexa with a single command. <https://www.theverge.com/2017/9/27/16375050/alexa-routines-echo-amazon-2017/>, Accessed June 2018.
- [210] THE VERGE. Google clarifies Works with Nest shutdown, provides extension on existing connections. <https://www.theverge.com/2019/5/16/18627719/google-works-with-nest-shutdown-clarification-statement-update/>, Accessed May 2021.
- [211] THE VERGE. Google's Nest changes risk making the smart home a little dumber. <https://www.theverge.com/2019/5/9/18537331/google-nest-cancellation-assistant-smart-home-integration-report>, Accessed May 2021.
- [212] YUAN TIAN, NAN ZHANG, YUEH-HSUN LIN, XIAOFENG WANG, BLASE UR, XI-ANZHENG GUO, AND PATRICK TAGUE. Smartauth: User-centered authorization for the internet of things. In *Proceedings of the 26th USENIX Security Symposium*, 2017.

- [213] TP-LINK. Does Nest work with TP-Link Kasa. <https://www.tp-link.com/us/support/faq/2014/>, Accessed May 2021.
- [214] UPROXX. How ted cruz’s app is taking privacy invasion to all new levels. <https://uproxx.com/technology/ted-cruz-app/>. Accessed May 2023.
- [215] U.S. CONGRESS. S. 2398 - a bill to amend the federal election campaign act of 1971 to ensure privacy with respect to voter information. <https://www.congress.gov/bill/116th-congress/senate-bill/2398/text>. Accessed May 2023.
- [216] HAOYU WANG, ZHE LIU, JINGYUE LIANG, NARSEO VALLINA-RODRIGUEZ, YAO GUO, LI LI, JUAN TAPIADOR, JINGCUN CAO, AND GUOAI XU. Beyond google play: A large-scale comparative study of chinese android app markets. In *Proceedings of the Internet Measurement Conference 2018*, pages 293–307, 2018.
- [217] QI WANG, WAJIH UL HASSAN, ADAM BATES, AND CARL GUNTER. Fear and Logging in the Internet of Things. In *Network and Distributed Systems Symposium*, 2018.
- [218] WATCHGUARD. Https content inspection. <https://www.watchguard.com/wgrd-solutions/security-topics/https-inspection>. Accessed April 2022.
- [219] WEB TRIBUNAL. 21 ssl statistics that show why security matters so much. <https://webtribunal.net/blog/ssl-stats/>. Accessed April 2022.
- [220] PAUL WEBB AND TIM BALE. Shopping for a better deal? party switching among grassroots members in britain. *Journal of Elections, Public Opinion and Parties*, 33(2):247–257, 2023.
- [221] CRAIG E WILLS AND CAN TATAR. Understanding what they do with what they know. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, pages 13–18, 2012.

- [222] WORKS WITH NEST STORE. Keen Home. <https://workswith.nest.com/company/leaksmart/leaksmart//>, Accessed June 2018.
- [223] WORKS WITH NEST STORE. Nest x Yale Lock. <https://workswith.nest.com/company/yale/nest-x-yale-lock//>, Accessed June 2018.
- [224] WORKS WITH NEST STORE. Somfy Protect. <https://workswith.nest.com/company/somfy-protect-by-myfox-sas/works-with-somfy-protect/>, Accessed June 2018.
- [225] CHRIS WRIGHT, CRISPIN COWAN, STEPHEN SMALLEY, JAMES MORRIS, AND GREG KROAH-HARTMAN. Linux Security Modules: General Security Support for the Linux Kernel. In *Proceedings of the 11th USENIX Security Symposium*, August 2002.
- [226] YETI. Yeti - Simplify the control of your smart home. <https://getyeti.co/>, Accessed June 2018.
- [227] YONOMI. Yonomi app – Yonomi. <https://www.yonomi.co>, Accessed June 2018.
- [228] LE YU, XIAPU LUO, XULE LIU, AND TAO ZHANG. Can We Trust the Privacy Policies of Android Apps? In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2016.
- [229] STEVE ZDANCEWIC, LANTIAN ZHENG, NATHANIEL NYSTROM, AND ANDREW C. MYERS. Secure program partitioning. *ACM Transactions on Computing Systems*, 20(3):283–328, 2002.
- [230] NICKOLAI ZELDOVICH, SILAS BOYD-WICKIZER, EDDIE KOHLER, AND DAVID MAZIÈRES. Making Information Flow Explicit in HiStar. In *Proceedings of the 7th symposium on Operating Systems Design and Implementation (OSDI)*, pages 263–278, 2006.

- [231] SEBASTIAN ZIMMECK, ZIQI WANG, LIEYONG ZOU, ROGER IYENGAR, BIN LIU, FLORIAN SCHAUB, SHOMIR WILSON, NORMAN SADEH, STEVEN M. BELLOVIN, AND JOEL REIDENBERG. Automated Analysis of Privacy Requirements for Mobile Apps. In *Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS)*, 2017.