Thesis for The Degree of Licentiate of Engineering

A Principled Approach to Securing IoT Apps

Iulia Bastys

CHALMERS

Division of Information Security
Department of Computer Science & Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2018

A Principled Approach to Securing IoT Apps

Iulia Bastys

Technical Report No 185L
ISSN 1652-876X
Department of Computer Science & Engineering
Division of Information Security
Chalmers University of Technology
Gothenburg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Reproservice,
Gothenburg, Sweden 2018.

Abstract

IoT apps are becoming increasingly popular as they allow users to manage their digital lives by connecting otherwise unconnected devices and services: cyberphysical “things” such as smart homes, cars, or fitness armbands, to online services such as Google or Dropbox, to social networks such as Facebook or Twitter. IoT apps rely on end-user programming, such that anyone with an active account on the platform can create and publish apps, with the majority of apps being created by third parties.

We demonstrate that the most popular IoT app platforms are susceptible to attacks by malicious app makers and suggest short and longterm countermeasures for securing the apps. For short-term protection we rely on access control and suggest the apps to be classified either as exclusively private or exclusively public, disallowing in this way information from private sources to flow to public sinks.

For longterm protection we rely on a principled approach for designing information flow controls. Following these principles we define *projected security*, a variant of noninterference that captures the attacker’s view of an app, and design two mechanisms for enforcing it. A static enforcement based on a flow-sensitive type system may be used by the platform to statically analyze the apps before being published on the app store. This enforcement covers leaks stemming from both explicit and implicit flows, but is not expressive enough to address timing attacks. Hence we design a second enforcement based on a dynamic monitor that covers the timing channels as well.

Keywords: information flow control, Internet of Things, IoT apps, design principles

Acknowledgments

First, I would like to thank my supervisor Andrei for giving me this incredible opportunity to be part of his group, for his guidance and support, and for constantly pushing me out of my comfort zone. *Spasibo!*

I am grateful to Dave, Gerardo, and Wolfgang for their help in the past year and for interesting discussions about books, movies, or teaching.

Several people have made my transition to this new environment smoother and my stay here more enjoyable. Thank you all! Elena, for being the best buddy student one can have; Georgia, for reminding me that there are other great things out there; Thomas, for adding a bit of refinement to my world; Daniel and Evgenii, for bringing reason to my sometimes emotion-grounded arguments; Alexander and Jeff, for making the work space a fun space; Max, for teaching me not to care so much sometimes.

To others, new and old, past and present: Alejandro, Benjamin, Carlo, Danielito, Elisabet, Hamid, Marco, Mauricio, Musard, Sandro, Simon, Sólrún, Steven and others, thank you for making (and having made) Chalmers such a welcoming and friendly environment.

My deepest gratitude is directed towards my family, for giving me strength and for making me who I am today.

Last, but not least, a special token of appreciation goes to Tomas, for always believing in me. *Ya lyublyu tebya!*

Contents

Introduction

Motivation

By their nature, IoT apps have access to a diverse set of user sensitive information: location, fitness data, private feed from social networks, private documents, or private images. Other IoT apps are given sensitive controls over burglary alarms, thermostats, or baby monitors. In addition, the apps rely on end-user programming, such that anyone can create and publish IoT apps, with the majority of apps being created by third parties. With the increase in popularity of IoT apps, concerns have been raised about keeping user information private or assuring the integrity and availability of data manipulated by the apps. These concerns are not unfounded, as we demonstrate the most popular IoT app platforms to be vulnerable to attacks by malicious app makers.

Background

Starting in 1982 with a single Internet-connected appliance—a drinks vending machine that was only able to report its inventory [39]—the number of IoT devices increased to 8.4 billion in 2017 [16], with, e.g., smart locks, virtual assistants, home appliances, emergency notification systems, or surveillance systems that perform more complex tasks and from longer distances. The number of IoT devices is estimated to grow to 30 billion by 2020 [31].

IoT stands for Internet of Things and, as the name suggests, it defines a network of diverse physical devices embedded amongst others with electronics, software, and sensors that allow for interconnections and data exchange.

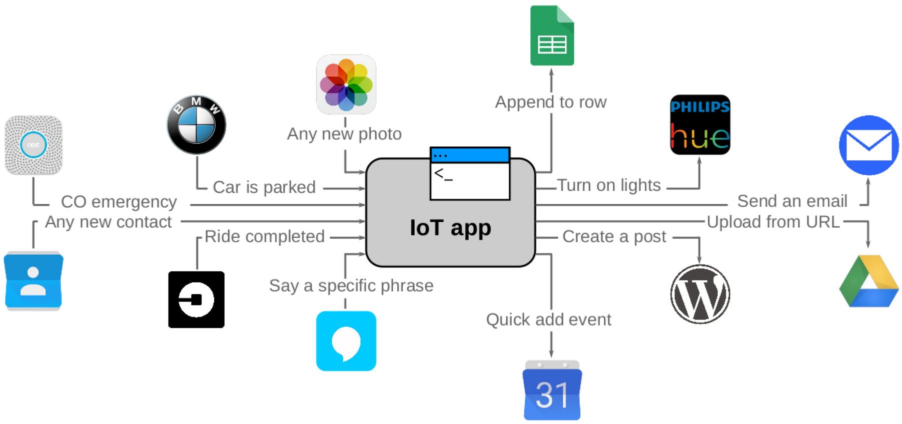

Figure 1: IoT app platform (simplified)

IoT system architecture IoT systems are used for performing a wide range of tasks, from simple ones that control light switches based on motion, to more complex ones that assist in transportation systems. However sophisticated the task to be performed is, the structure of an IoT system is roughly the same. It mainly comprises devices, connectivity protocols, and programming platforms.

Devices are equipped with sensors, which collect data and send events to other devices, the hub or the cloud, and actuators, which process these events and allow the devices to perform an action. For example, when a presence sensor detects movement, it communicates with a switch, in this case the actuator, which will turn on the light. Gateways connect devices with the cloud, while cloud gateways ensure secure communication between the two. Cloud gateways are also responsible for the communication protocols between heterogeneous devices. IoT programming platforms provide users with applications that allow them to monitor and control their devices.

IoT app platforms Provider-specific programming platforms abound on the market: Android Things [4] and Google Fit [19] (from Google), HomeKit [5] (from Apple), SmartThings [34] (from Samsung), or AWS IoT [3] (from Amazon) are just few examples. Other platforms allow building automations that connect devices and services originating from different providers, with IFTTT [25], Zapier [42], and Microsoft Flow [28] being the most popular IoT platforms of this kind.

All platforms offer web-based environments and tools (with some providing smartphone clients as well) that enable creating custom automations, referred to as applications or apps. Most platforms allow not only the service providers, but also both experienced developers and uninitiated users to create such apps, with the majority of IoT apps being created by third parties. Each platform provides (potentially) a different language for specifying these apps (e.g., JavaScript for IFTTT [26] and Zapier [43], Python for Zapier [44], or Groovy for SmartThings [36]) and uses (potentially) a different environment for executing them (e.g., the cloud for IFTTT [26] or a local hub for SmartThings [35]). Additionally, for performance and security reasons, some IoT platforms execute the apps in a sandbox (e.g., IFTTT [26], Zapier [43, 44], or SmartThings [37]).

IoT apps rely on a trigger-action paradigm: when an event takes place (the trigger), such as "Carbon monoxide emergency", another event is produced (the action), such as "Turn on the lights". Platforms allow for specifying JavaScript, Python, or Groovy code, depending on the case, for action refinement, such as "to red color". This refinement is optional, e.g., on IFTTT and Zapier platforms.

IoT platforms provide automations beyond physical environments, with online services such as Google and Dropbox, or social networks such as Facebook and Twitter, added to the equation (Fig. 1). Any combination between "things", online services, and social networks is possible. Figure 2 displays an (IFTTT) app that uploads any new iOS photo taken by the user to their Google Drive.

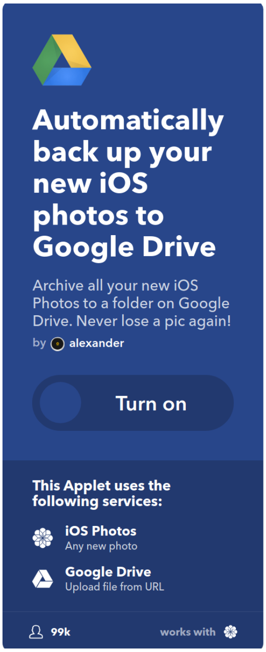

Figure 2: App view on IFTTT platform

Before installing an app, users can see what triggers and actions the given app may use, e.g., trigger "Any new photo" and action "Upload file from URL" for app in Fig. 2. To be able to run the app, users need to provide their credentials to the services associated with its triggers and actions, e.g., iOS Photos and Google Drive for app in Fig. 2. The user can also see the app maker and the number of installs, e.g., third party user `alexander` and 99k installs for app in Fig. 2.

IoT platforms incorporate a basic form of access control. The users explicitly allow the app to access their trigger data (e.g., their iOS photos), *but* only to be used by the action service (e.g., by their Google Drive). In order to achieve this, app code is heavily sandboxed by design, with no blocking or I/O capabilities and access only to APIs pertaining to the services used by the app.

IoT security and privacy While IoT advertises better safety, improved energy and manufacturing efficiency, enhanced health care and crop management, or automation of mundane tasks, concerns about user security and privacy in the IoT ecosystem have been voiced.

In order to provide the user with the expected functionality, IoT apps have access not only to physical functions, which when exploited may lead to safety and security issues, but also to user sensitive data, which when leaked may cause privacy issues. Abusing the smart lock to unlock the door when the user is not at home, or the thermostat to increase the heat to cause the windows to open are a couple of examples of security risks the user may be exposed to. Also, access to data provided by heart rate monitors or smart meters may reveal to unauthorized parties information about the consumer's health, or behavioral patterns and what type of home appliances the consumer is using and when [32].

Attack vectors Unfortunately, these concerns are not entirely unfounded. Recent studies have revealed several vulnerabilities [11, 14, 15, 22, 38, 41] and demonstrated attacks [8, 27] and privacy abuses in IoT devices and on IoT platforms [17].

An infamous example of vendor access privilege abuse is represented by the Xiaomi Mi Robot vacuum cleaner. A recent study [17] revealed the vacuum cleaner was uploading to the cloud not only the names and passwords of the WiFi networks to which the vacuum cleaner connected to, but also the maps of the rooms it cleaned in. Judging by the size of the rooms, information about the user's wealth and social status could be inferred. Pairing with location information (possibly) collected from the user's smartphone via the recommended app, the precise geolocation of the user could be learned. Moreover, since the stored data is never deleted from the cloud, not even after a factory reset, somebody buying a used Xiaomi vacuum cleaner could also get access to the information about previous usages and owners.

Other threat models in IoT focus on 'external' attackers, i.e. different from the vendor. For example, at the hardware level, an attacker can manipulate the IoT device during the fabrication time to maintain the privilege bit of the processor to a target value [41]. At the software level, the range of vulnerabilities and attacks is larger and of more interest. With respect to access control vulnerabilities we have evidence of inappropriate design of granularity in access control on the SmartThings platform [14], over-privileged OAuth tokens on IoT platforms [15], (potentially) illegal intra-flows between different IoT apps [38], limitations of access control and authentication models for Home

IoT [22], or untrusted code accessing sensitive sources [22]. Privacy violations in IoT apps [11], CSRF attacks in IFTTT [27], or programming errors in rule-based smart homes [30] augment the list.

Contributions

In the abundance of threat models in IoT one aspect has been largely overlooked by previous research: the actual inter-flows emitted by the apps and the capabilities of a malicious app maker to exfiltrate user private data.

In this work, we demonstrate that apps may leak user data via URL-based attacks by malicious app makers. To prevent such attacks, we propose short and longterm countermeasures. For short-term protection we rely on access control and suggest the apps to be classified either as exclusively private or exclusively public, disallowing in this way information from private sources to flow to public sinks. This approach is backward-compatible with the current model of IoT platforms. For longterm protection and for securing more complex apps that allow for queries or multiple sources and sinks, we suggest tracking the information flows in IoT apps.

Design principles for IFC Information flow control (IFC) tracks the data flows in a system and prevents those flows from sensitive sources to public sinks. The policy enforcing this restriction is usually referred to as noninterference [18] and the literature abounds with different variants for it [2, 6, 7, 21, 33, 40] and with as many different enforcement mechanisms [12, 13, 20, 24, 29, 40].

The myriad of existing models and security conditions do not fully cover the privacy concerns raised by the flows in IoT apps and the URL-based attacks. Thus, we require a principled approach for choosing the right security characterization and for selecting the right enforcement mechanism for it.

In this regard, inspired by the seminal work of Abadi and Needham on prudent engineering practice for cryptographic protocols [1], we outline six principles [9] to assist the security designer in tailoring information flow controls for a new application domain, such as intra-flows in IoT apps. Two core principles—attacker-driven security and trust-aware enforcement—refer to properly defining the attacker model and the trusting computing base. Other four principles are secondary and tightly connected to the core principles: separation of policy annotations and code, language-independent security condition and enforcement, justified abstraction when defining the attacker, and permissiveness of enforcement mechanism.

Projected security and enforcement mechanisms Applying these principles when securing IoT apps against the URL-based attacks, we define *pro-*

jected security, a variant of noninterference that takes into account the attacker's view of an app, and we design enforcement mechanisms that provably enforce this condition [8, 10].

Envisioning a platform where the IoT apps are statically analyzed for security before being published, we design a flow-sensitive type system that enforces projected noninterference [10]. The type system can track both explicit and implicit flows and it can be trivially extended to cover presence channels, but it cannot handle information leaks via the timing channel. To capture these flows, we design a dynamic monitor [8] and implement it as an extension of JSFlow [23], an information flow tracker for JavaScript.

Thesis structure

Paper 1: Prudent Design Principles for Information Flow Control [9]

This short paper aims to systematize and structure the plethora of security characterizations and enforcement mechanisms in the literature to assist a security designer when designing information flow controls for new application domains. In this regard, we introduce six design principles. Two main principles roughly refer to defining the attacker model and the trusting computing base: attacker-driven security and trust-aware enforcement. The other four principles are in close connection to the main ones, and refer to separation of policy annotations and code, language-independent security condition and enforcement, justified abstraction when defining the attacker, and permissiveness of enforcement mechanism.

Statement of contributions This paper was in collaboration with Frank Piessens and Andrei Sabelfeld. Iulia was responsible with flashing out the principles and illustrating them with concrete examples in JSFlow.

Appeared in: *Proceedings of the 13th Workshop on Programming Languages and Analysis for Security (PLAS 2018), Toronto, Canada, October 2018.*

Paper 2: If This Then What? Controlling Flows in IoT Apps [8]

This paper demonstrates a new class of vulnerabilities on popular IoT app platforms (IFTTT, Zapier, and Microsoft Flow), this time with the attacker assumed to be a malicious app maker. In order to estimate the impact of the possible attacks, we conduct an empirical study on a set of roughly 300 000 IFTTT apps. We find that 30% of the existing apps may not only violate privacy, but also do it *invisibly* to its users.

One protection mechanism we suggest is based on access control and it disallows flows from private sources to public sinks by classifying the apps either as exclusively public or exclusively private. A second protection mechanism based on information flow control (IFC) covers in addition apps with more complex functionality that deal with flows from several sources and to several sinks.

We implement the latter mechanism as a dynamic monitor that extends JSFlow, a taint tracker for JavaScript, and prove its soundness. We then evaluate the monitor on a set of 60 apps, 30 secure and 30 insecure. We obtain no false negatives and a single false positive on 'artificially'-constructed code, proving that IFC is a suitable enforcement mechanism for securing IoT apps.

Statement of contributions This paper was in collaboration with Musard Balliu and Andrei Sabelfeld. Iulia was responsible for designing the semantics of the dynamic monitor, proving its soundness, implementing it as an extension of JSFlow, and evaluating it.

Appeared in: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS 2018), Toronto, Canada, October 2018.*

Paper 3: Tracking Information Flow via Delayed Output: Addressing Privacy in IoT and Emailing Apps [10]

This paper focuses on tracking information flow in the presence of delayed output in two scenarios with different levels of trust in the computing base: IoT apps and email campaigns. Delayed output is structured output in a markup language generated by a service and subsequently processed by a different service. For example, in the case of HTML, the output is generated by a webserver and later processed by browsers or email readers.

Both IoT apps and email campaigns are vulnerable to exfiltrations via delayed output, with the distinction that IoT apps can be written by endusers and are potentially malicious, while email campaigns are written by the service providers and are non-malicious, but potentially buggy. We develop a formal framework to reason about secure information flow with delayed output in both settings and design static enforcement mechanisms based on type systems. The enforcement for malicious code entails a type system that tracks both explicit and implicit flows, while the type system for the non-malicious code only tracks (explicit) data flows. Both type systems are formally proven to be sound.

Statement of contributions This paper was in collaboration with Frank Piessens and Andrei Sabelfeld. Iulia was responsible with designing the type systems and proving their soundness, and for verifying the exfiltrations via delayed output on other platforms.

To appear in: *The 23rd Nordic Conference on Secure IT Systems (NordSec 2018), Oslo, Norway, November 2018.*

Bibliography

[1] M. Abadi and R. M. Needham. Prudent Engineering Practice for Cryptographic Protocols. *IEEE Trans. Software Eng.*, 22(1):6–15, 1996.

[2] J. Agat. Transforming out timing leaks. In *Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2000, Boston, MA, USA, January 19-21, 2000*, pages 40–53. ACM, 2000.

[3] Amazon Web Services (AWS) IoT. `https://aws.amazon.com/iot/`, 2018.

[4] Android Things. `https://developer.android.com/things/`, 2018.

[5] Apple HomeKit. `https://www.apple.com/ios/home/`, 2018.

[6] A. Askarov, S. Hunt, A. Sabelfeld, and D. Sands. Termination-insensitive noninterference leaks more than just a bit. In *Computer Security - ESORICS 2008 - 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings*, volume 5283 of *Lecture Notes in Computer Science*, pages 333–348. Springer, 2008.

[7] A. Askarov and A. Sabelfeld. Gradual release: Unifying declassification, encryption and key release policies. In *28th IEEE Symposium on Security and Privacy, S&P 2007, Oakland, CA, USA, May 20-23, 2007*, pages 207–221. IEEE Computer Society, 2007.

[8] I. Bastys, M. Balliu, and A. Sabelfeld. If This Then What? Controlling Flows in IoT Apps. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1102–1119, 2018.

[9] I. Bastys, F. Piessens, and A. Sabelfeld. Prudent Design Principles for Information Flow Control. In *Proceedings of the 13th Workshop on Programming Languages and Analysis for Security*, pages 17–23. ACM, 2018.

[10] I. Bastys, F. Piessens, and A. Sabelfeld. Tracking Information Flow via Delayed Output: Addressing Privacy in IoT and Emailing Apps. In *23rd Nordic Conference on Secure IT Systems (NordSec 2018), Oslo, Norway, November 28-30, 2018*, To appear.

[11] Z. B. Celik, L. Babun, A. K. Sikder, H. Aksu, G. Tan, P. D. McDaniel, and A. S. Uluagac. Sensitive Information Tracking in Commodity IoT. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018.*, pages 1687–1704, 2018.

[12] D. E. Denning and P. J. Denning. Certification of programs for secure information flow. *Commun. ACM*, 1977.

[13] D. Devriese and F. Piessens. Noninterference through secure multi-execution. In *31st IEEE Symposium on Security and Privacy, S&P 2010, Oakland, CA, USA, May 16-19, 2010*, pages 109–124. IEEE Computer Society, 2010.

[14] E. Fernandes, J. Jung, and A. Prakash. Security Analysis of Emerging Smart Home Applications. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 636–654, 2016.

[15] E. Fernandes, A. Rahmati, J. Jung, and A. Prakash. Decentralized Action Integrity for Trigger-Action IoT Platforms. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*, 2018.

[16] Gartner Says 8.4 Billion Connected “Things” Will Be in Use in 2017, Up 31 Percent From 2016. `https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016`. Accessed on November 3rd, 2018.

[17] D. Giese and D. Wegemer. `https://github.com/dgiese/dustcloud`, 2018.

[18] J. A. Goguen and J. Meseguer. Security policies and security models. In *1982 IEEE Symposium on Security and Privacy, S&P 1982, Oakland, CA, USA, April 26-28, 1982*, pages 11–20. IEEE Computer Society, 1982.

[19] Google Fit: Coaching you to a healthier and more active life. `https://www.google.com/fit/`, 2018.

[20] G. L. Guernic. Automaton-based confidentiality monitoring of concurrent programs. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium, CSF 2007, Venice, Italy, 6-8 July, 2007*, pages 218–232. IEEE Computer Society, 2007.

[21] J. Y. Halpern and K. R. O'Neill. Secrecy in multiagent systems. *ACM Trans. Inf. Syst. Secur.*, 12(1):5:1–5:47, 2008.

[22] W. He, M. Golla, R. Padhi, J. Ofek, M. Dürmuth, E. Fernandes, and B. Ur. Rethinking access control and authentication for the home internet of things (iot). In *27th USENIX Security Symposium (USENIX Security 18)*, Baltimore, MD, 2018. USENIX Association.

[23] D. Hedin, A. Birgisson, L. Bello, and A. Sabelfeld. JSFlow: Tracking Information Flow in JavaScript and its APIs. In *SAC*, 2014.

[24] S. Hunt and D. Sands. On flow-sensitive security types. In *Proceedings of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2006, Charleston, SC, USA, January 11-13, 2006*, pages 79–90. ACM, 2006.

[25] IFTTT (IF This Then That). `https://ifttt.com`, 2018.

[26] IFTTT: Maker guide. `https://platform.ifttt.com/maker/guide`, 2018.

[27] E. Kang, A. Milicevic, and D. Jackson. Multi-representational Security Analysis. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2016, pages 181–192, 2016.

[28] Microsoft Flow. `https://flow.microsoft.com/`, 2018.

[29] S. Moore and S. Chong. Static analysis for efficient hybrid information-flow control. In *Proceedings of the 24th IEEE Computer Security Foundations Symposium, CSF 2011, Cernay-la-Ville, France, 27-29 June, 2011*, pages 146–160. IEEE Computer Society, 2011.

[30] C. Nandi and M. D. Ernst. Automatic trigger generation for rule-based smart homes. In *Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security*, PLAS '16, pages 97–102, 2016.

[31] Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated. `https://spectrum.ieee.org/tech-`

talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated. Accessed on November 3rd, 2018.

[32] E. L. Quinn. Privacy and the new energy infrastructure. 2009.

[33] A. Sabelfeld and A. C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003.

[34] Samsung SmartThings: Add a little smartness to your things. https://www.smartthings.com/, 2018.

[35] SmartThings Classic Documentation: Architecture. https://docs.smartthings.com/en/latest/architecture/, 2018.

[36] SmartThings Classic Documentation: Groovy basics. https://docs.smartthings.com/en/latest/getting-started/groovy-basics.html, 2018.

[37] SmartThings Classic Documentation: Groovy with SmartThings. https://docs.smartthings.com/en/latest/getting-started/groovy-for-smartthings.html, 2018.

[38] M. Surbatovich, J. Aljuraidan, L. Bauer, A. Das, and L. Jia. Some recipes can do more than spoil your appetite: Analyzing the security and privacy risks of IFTTT recipes. In *WWW*, 2017.

[39] The "Only" Coke Machine on the Internet. https://www.cs.cmu.edu/~coke/history_long.txt. Accessed on November 3rd, 2018.

[40] D. M. Volpano, C. E. Irvine, and G. Smith. A sound type system for secure flow analysis. *Journal of Computer Security*, 4(2/3):167–188, 1996.

[41] K. Yang, M. Hicks, Q. Dong, T. M. Austin, and D. Sylvester. A2: Analog Malicious Hardware. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 18–37, 2016.

[42] Zapier. https://zapier.com/, 2018.

[43] Zapier: How to Get Started with Code (JavaScript) on Zapier. https://zapier.com/help/code/, 2018.

[44] Zapier: How to Get Started with Code (Python) on Zapier. https://zapier.com/help/code-python/, 2018.