Securing IoT Apps

Musard Balliu | KTH Royal Institute of Technology
Iulia Bastys and Andrei Sabelfeld | Chalmers University of Technology

Users increasingly rely on Internet of Things (IoT) apps to manage their digital lives through the overwhelming diversity of IoT services and devices. Are the IoT app platforms doing enough to protect the privacy and security of their users? By securing IoT apps, how can we help users reclaim control over their data?

The world of the Internet of Things (IoT) is fascinating, but who is in charge? Meet Iona, whose story of ups and downs in the IoT world will help us illustrate how the technical aspects of securing IoT apps can have real-life impact on nontechnical users.

Users Lack Control Over Their Digital Lives

Scenarios like this illustrate that users often lack sufficient control over their digital lives. The heterogeneous nature of the IoT implies that although the services and devices might be connected by a network, robust application support is needed so that the interacting services and devices can be controlled by the users. Rather than reinventing new protocols and standards for the IoT, the Web of Things[1] reuses well-known web standards to enable a smooth application layer for IoT applications. Billions of devices, from printers to smart TVs, already routinely run web

Digital Object Identifier 10.1109/MSEC.2019.2914190
Date of publication: 5 June 2019

Take 1: Help!

On her way home, Iona parks her car at a shopping mall, takes a picture of the season's first snow in a nearby park, and heads to the mall for some shopping. However, when her shopping is done, she has a hard time remembering where she parked her car. She realizes she accidentally deleted the first-snow picture as she was fiddling with her phone. To make things worse, she also realizes that she forgot to turn on the thermostat at home, which is unfortunate given the chilly weather. All of this is especially frustrating because her phone is an Internet-connected smartphone; her car is a connected car, with rich Internet and infotainment features; and her thermostat is connected to the Internet through the vendor's portal. Connectivity alone is clearly not enough to manage Iona's digital life through the overwhelming diversity of IoT services and devices.

 Copublished by the IEEE Computer and Reliability Societies

servers and clients, forming a heterogeneous Web of Things. In the automotive domain, HTML5/JavaScript standards enable web connectivity through in-vehicle infotainment systems and vehicle data-access protocols.[2]

Just as the Internet provides network-level connectivity and the web provides application-level connectivity, IoT apps take the main stage for managing users' digital lives. For our general purposes, an *IoT app* is a piece of software that runs on behalf of the user to implement a functionality in the IoT setting.

IoT App Platforms Enable Control...

IoT apps help users manage their digital lives by connecting Internet-connected components ranging from cyberphysical things (such as smart homes, cars, and fitness armbands) to online services (such as Google and Dropbox) and social networks (such as Facebook and Twitter). We focus on two prime examples of IoT app platforms: user-automation apps and in-vehicle apps. These two independently interesting scenarios help us illustrate that although some problems and solutions are common to IoT apps, others are more specific (such as URL-based threats for web-driven IoT apps versus road-safety risks for in-vehicle apps).

Popular user-automation platforms include If This Then That (IFTTT), Zapier, and Microsoft Flow. IFTTT, the most popular of these platforms, supports more than 550 Internet-connected components and services, 11 million users and 54 million apps, and one billion apps run per month.[3] At the core of these platforms are reactive apps, which include triggers, actions, and filter code for customization. Triggers and actions may involve ingredients, enabling app makers to pass parameters to triggers and actions. The filter part is invoked after a trigger has been fired and before an action is dispatched. Filters allow apps to be highly customizable: they are essentially code snippets, often in JavaScript, with application programming interfaces (APIs) pertaining to the services used. Users can make apps and publish them for other users as platforms capitalize on the model of end-user programming. Figure 1 depicts the architecture of a user-automation app, illustrating how triggers act as information sources and actions act as information sinks.

Cars are now equipped with so-called infotainment systems, the abilities of which have developed over the years from basic radio and navigation units to powerful Internet-connected devices comparable to tablets and smartphones. Recently, several car manufacturers, including Volvo, Renault, Nissan, and Mitsubishi, announced the upcoming possibility of installing third-party apps onto these infotainment systems. These manufacturers leverage a special version of Android for use in cars, called *Android Automotive*,[4] an open platform where third parties can publish their apps. Similar to user-automation apps, in-vehicle car apps offer a variety of features. At the same time, these apps have access to sensitive information, such as car location, and have some capability of affecting the vehicle while it is on the road.

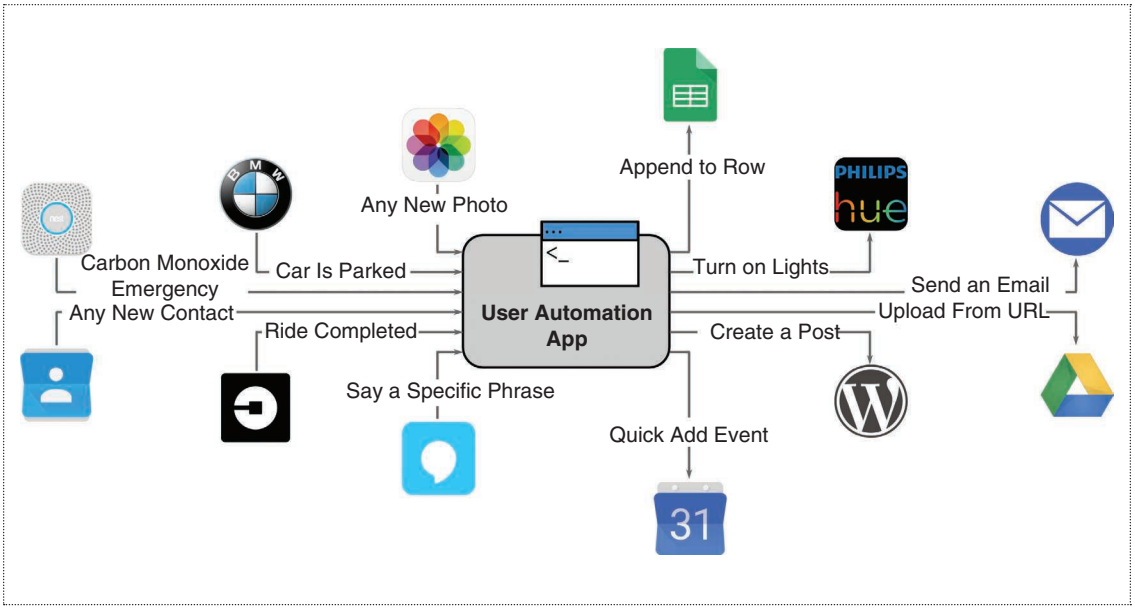

Figure 1. User-automation apps connect trigger and action services.

Thanks to these developments, Iona's digital life is now in the hands of powerful IoT apps.

Take 2: Feeling in Control?

On her way home, Iona parks her car at a shopping mall, takes a picture of the season's first snow in a nearby park, and heads to the mall for some shopping. She receives an email with the map where her car is parked. Her picture is automatically backed up on Google Drive. Her thermostat turns on automatically, based on her proximity to home.

Is Iona's problem now solved?

...And Weaponize the Attacker...

Unfortunately, the power of IoT apps can be abused by attackers. Although app stores boost innovation and market potential (as seen in successful examples like Google Play), they also create possibilities for attacks by malicious developers. In the area of web and mobile security, the recent breach of personal data of more than 50 million Facebook users by Cambridge Analytica's malicious Facebook app[5] provides alarming evidence that threats from malicious third-party apps are real. IoT apps, such as those on the IFTTT platform, access sensitive user location, fitness information, content of private files, or private feed from social networks. This sensitive information can be compromised by insecure or buggy apps.

Figure 2 shows a user's view of a third-party user-automation app, consisting of trigger "Any new photo" (provided by iOS Photos), action "Upload file from URL" (provided by Google Drive), and execution of filter code transparently to the user. The desired expectation is that users explicitly allow the app to access their photos but only to be used on their Google Drive. However, the user cannot inspect the filter code or the ingredient parameters and is not told whether filter code is present at all. Moreover, modifications in the filter code or ingredients can be performed at any time by the app maker, with no user notification. As a result, a third-party maker is granted the opportunity to make and publish malicious apps for all users with the goal of crafting filter code and ingredient parameters to exfiltrate users' photos.

As previously mentioned, several car manufacturers have announced the upcoming possibility of installing third-party apps onto their infotainment systems.

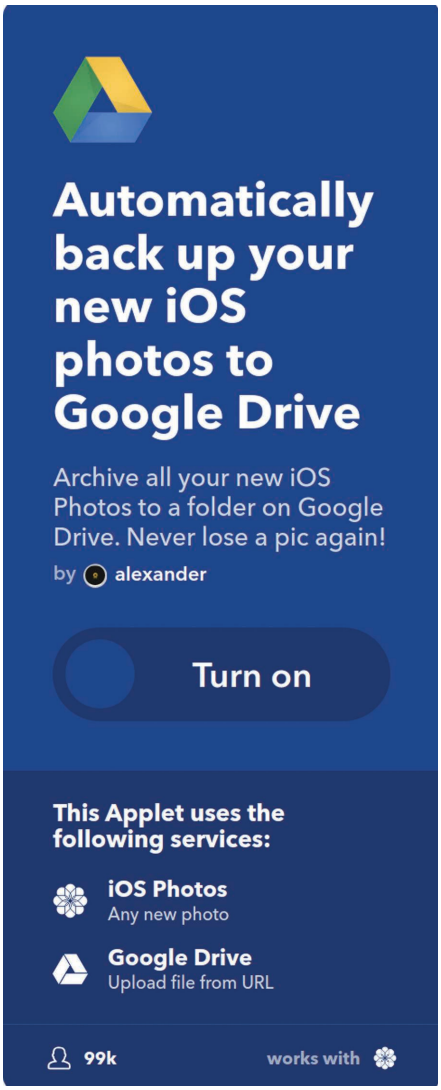

Figure 2. The app view on the IFTTT platform.

Because these apps have access to sensitive resources, such as the car's location, they can also be of interest to malicious app makers. A series of well-publicized attacks[6] in the domain of Internet-connected cars exploited the infotainment software in Jeeps to send commands to the dashboard functions, steering, brakes, and transmission system, gaining full control of the car from a remote laptop. In 2015, Chrysler issued a formal recall for 1.4 million vehicles affected by the vulnerability. This highlights the need to secure IoT apps against third-party makers.

The exposure of safety- and security-critical information to the web via IoT platforms increases the attack surface, enabling different kinds of attackers to take advantage of potential vulnerabilities at different

Table 1. An overview of threat models in IoT apps.

Attacker	Threat and vulnerability	Attack vector	Current defense	Proposed defense
Third-party maker	Malicious/buggy app	URL upload and URL markup SoundBlast, Intent Storm, ForkBomb	Code review, Sandbox Coarse-grained permissions	Fine-grained access control Information flow control
Malicious service/user	Seemingly harmless app Cross-app interaction	Unintended audience Unintended interaction	Auditing	User awareness Program analysis
Cloud attacker	Overprivileges Compromised service Compromised platform	Authorization token misuse	Coarse-grained permissions	Fine-grained access control Decentralization
Malicious platform	Third-party platform	Authorization token misuse	None	Decentralization

components of the IoT app ecosystem: the environment, physical devices, services, communication network, cloud-based IoT platform, and user interfaces. Table 1 (first column) provides an overview of the attackers that arise in the context of web-connected IoT apps.

In addition to third-party makers, a malicious user or service may have access to the source and sink services of an IoT app, for instance, by being part of the user's audience of a social media post or simply by being able to send emails to the user. A benign IoT app that connects such services may enable sensitive information disclosure that a user did not consider as possible at the time of app's installation, for example, by enlarging the audience of a social media post or by receiving malware via email attachments. Moreover, the unique ability of IoT apps to affect the shared (physical and/or logical) environment, such as the room temperature or the cloud storage, enables unintended cross-app interactions between IoT apps that are installed by the same user. Finally, because the interaction between services is materialized on the cloud-based IoT platform via the Internet, IoT apps inherit classical weaknesses with respect to the cloud attacker and malicious platforms.

...In the Face of Current Security Mechanisms

IoT platforms incorporate varying forms of access control and authorization to control the access to sensitive APIs. For instance, the IFTTT platform requires user authentication and authorization on partner services, such as iOS Photos and Google Drive to poll a trigger's service for new data or to push data to a service in response to the execution of an action. This is achieved through the OAuth 2.0 authorization protocol, which, upon app installation, redirects the user to the authentication page hosted by the service provider. An access token is then generated and used by the platform for future executions of any apps that use such services. For the app in Figure 2, the user gives the app access to his/her iOS Photos and Google Drive. Such permissions are coarse-grained, giving access to more user information than what the app requires to perform its functionality. Furthermore, the filter code is run in an isolated environment (called the *sandbox*) with a short timeout. By design, the sandbox allows access only to APIs pertaining to the services used by the app; otherwise, it provides no input–output or blocking capabilities.

Unfortunately, malicious app makers can bypass the access control mechanism of the sandbox by crafting filter code. Platforms often leverage URLs as "universal glue" for connecting different services. iOS Photos and Google Drive, for example, provide URL-based APIs connected to app actions for uploading content. For the photo-backup app in Figure 2, IFTTT uploads a new photo to its server, creates a publicly accessible random URL, and passes it to Google Drive. URLs are also used by apps in other contexts, such as including custom images like logos in email notifications.

Bastys et al.[7] demonstrate two classes of URL-based attacks for stealth exfiltration of private information by apps: URL-upload attacks and URL-markup attacks. Under both attacks, a malicious maker may craft a URL by encoding the private information as a parameter part of a URL linking to a server under the attacker's control, such as https://attacker.com?secret.

In a URL-upload attack, the attacker exploits the capability of uploads via links. In a scenario in Figure 2, IFTTT stores any new photo on its server and passes it to Google Drive using an intermediate URL. Thus, the attacker can pass the intermediate URL to

its own server instead by string processing in the JavaScript code of the filter. For the attack to remain unnoticed, the attacker configures attacker.com to forward the original image in the response to Google Drive so that the image is backed up as expected by the user. This attack requires no additional user interaction because the link upload is (unsuspiciously) executed by Google Drive.

In a URL markup attack, the attacker creates HTML markup with a link to an invisible image with the crafted URL embedding the secret, such as the user's location map. The markup can be part of a post on a social network or the body of an email message. The leak is then executed by a web request upon processing the markup by a web browser or an email reader.

Bastys et al.[7] show that the other common user-automation platforms, Zapier and Microsoft Flow, are both vulnerable to URL-based attacks. URL-based exfiltration attacks are particularly powerful because of their stealth nature. They performed a measurement study on a data set of about 300,000 IFTTT apps from more than 400 services and found that 30% of the apps were susceptible to stealthy privacy attacks by malicious app makers.

In-vehicle apps are also susceptible to attacks by malicious makers.[8] The Android Automotive security architecture inherits much from the regular Android permission model.[9] This model forces the apps to request permissions before using the system resources. Sensitive resources, such as camera and GPS, require the user to explicitly grant them access before the app can use them. Other resources, such as using the Internet or near-field communication (better known as *NFC*), can be granted during installation. From a user's perspective, the security implications of these permissions are often hard to understand.

SoundBlast, a representative of disturbance attacks, demonstrates how a malicious app can shock the driver by excessive sound volume, for example, upon reaching high speed. Malicious apps can also trigger availability attacks, such as ForkBomb and Intent Storm, which render the infotainment system unusable until it is rebooted. Similar to user-automation apps, malicious in-vehicle apps can exfiltrate sensitive information, such as vehicle location and in-vehicle voice sound.

Other in-vehicle privacy threats obtain information from onboard sensors, such as speed, temperature, and engine r/min. Although accessing the current speed requires a permission, accessing the current r/min or gear requires no permission in Android Automotive. The attacker can thus easily approximate speed based on the r/min and gear data.[8]

Surbatovich et al.[10] point out that even benign IoT apps may cause security and privacy risks that a user did not anticipate at the time of the app's installation. For instance, the user-automation app "If I take a new photo, then upload on Flickr as public photo" could leak sensitive or embarrassing information if the user took a picture of a check to send to his/her landlord or a picture of a romantic partner. Furthermore, the interaction of user-automation apps installed by the same user can enable additional risks due to cross-app interactions. For instance, a user may install these two apps for different purposes: "If I leave my work location, turn on the thermostat at home" and "If the room temperature exceeds a threshold, open the windows." Although the user's intention is to use these apps for separate purposes, their interaction may open the window while the user is away, thus clearing the way for burglary. Similarly, a smoke-alarm app ("If smoke is detected, activate the alarm and open the water valve to activate the fire sprinklers") may interact with a water-leak-detector app ("If a water leak is detected, shut off the water valve") and shut off the water valve when a fire is detected, a scenario studied by Celik et al.[11]

Moreover, because billions of users confide their digital lives to a cloud-based IoT platform with powerful access to their services, both the cloud and the services become targets of cloud-based attacks. A compromised service allows an attacker to steal authorization tokens and perform sensitive actions on other user services. Similarly, a compromised IoT platform allows the attacker to affect billions of platform users. Fernandes et al.[12] show that overprivilege is a significant shortcoming of permission models in user-automation apps, despite the efforts of user-automation platforms to constrain dangerous privileges.

The exposure of permissions that are never used by IoT apps further increases the risk for malicious uses. Recently, different IoT platforms have begun to allow for interactions of IoT apps across IoT platforms, with the goal of overcoming the drawbacks of a given platform. For instance, because IFTTT does not allow multiple triggers in apps, platforms, such as apilio.io, have emerged and can be used to mash up different IFTTT apps to implement complex trigger logic. Furthermore, different platforms, such as IFTTT and Stringify, allow their respective apps to interact with each other. From a security perspective, such developments increase the attack surface, opening up opportunities for new breaches. Finally, as apps increasingly rely on AI-powered components, the ability to analyze and address adversarial threats for machine learning[13] is becoming increasingly important.

Iona's life gets harder than ever.

Take 3: Digital Life Hijacked

On her way home, Iona parks her car at a shopping mall, takes a picture of the season's first snow in a nearby park, and heads to the mall for some shopping. As she receives an email with the map where her car is parked, the map is also sent to an attacker, unbeknownst to her. As her picture is backed up on Google Drive, the picture is stealthily uploaded on the attacker's server as well. The attacker sets the thermostat on the highest temperature, causing the windows to open automatically and thus clearing a way for burglars to enter.

How can we secure IoT apps and platforms in the face of these threats?

Giving Control Back to the Users

The root cause of security and privacy violations in malicious and buggy IoT apps is the flow of information from sensitive sources to insensitive sinks. The problem is further exacerbated by the exposure of coarse-grained permissions by platforms and services to IoT apps, thereby increasing the risks whenever these platforms or services get compromised. Moreover, trusting the platforms and services with sensitive user data imposes additional risks whenever these data are used improperly. In the face of current threats and vulnerabilities, we discuss immediate and exploratory countermeasures that either 1) break the insecure flows through tighter access controls and decentralization or 2) track the information flows via information flow control.

Immediate Countermeasures

The granularity of access control in IoT apps varies from all-or-nothing to coarse-grained permissions. The attacks described underscore the need for fine-grained access control. This can be achieved by defining a security architecture that enables service providers to expose finer-grained APIs to IoT platforms, possibly with information about the sensitivity level of the data. IoT platforms, on the other hand, can improve their security mechanisms to enforce fine-grained access control, for example, by providing safe output encoding through APIs such that the only way to include links or image markup on a sink is through API constructors generated by the platform.

Users are entitled to have the final say on defining security policies over their data. At the same time, a security mechanism is practical only if it does not burden users with settings, notifications, or popups. Luckily, in many cases, fine-grained permissions can be automatically derived from the context. Thus, the overall workflow can be accommodated with minimal user effort. Service providers already need to register on the IoT platform as partner services. Hence, permissions can either be derived from the services used in a given app or checked by the users in a way similar to dynamic permissions in Android apps. Therefore, the app shown in Figure 2 will not necessarily need additional user interaction. The trigger ("Any new photo") can be automatically classified as sensitive. On the other hand, triggers like "Astronaut enters space" can be automatically classified as public.

Exploratory Countermeasures

IoT platforms are centralized entities that have privileged access to sensitive data and the devices of billions of users. As such, both platforms and services become an attractive target for attackers. If they are compromised, attackers can learn sensitive user data and arbitrarily manipulate user data and devices. Decentralization allows us to reduce the trust placed on the IoT platforms by full mediation of the communication between service providers via fine-grained authorization tokens, such as per-app tokens, and trusted client apps, such as mobile apps. Fernandes et al.[12] proposed a decentralized architecture that enforces the integrity of an app's actions with negligible performance overhead. However, protecting the privacy of user data in the context of a malicious platform remains an open problem. One solution is to build a decentralized peer-to-peer system between all service providers that are involved in a user-automation app and to implement and execute the app's functionality on one of the trigger's or action's services. By eliminating the cloud platform, this solution improves security and privacy at the expense of more complex services and business relationships among service providers. Another solution is to leverage the recent advances in homomorphic encryption and use the IoT platform only as a means of computing over encrypted data. For example, computing the proximity to a given location without revealing the actual user location requires a simple comparison over encrypted data, which is within reach for homomorphic encryption techniques.

A promising approach for protecting against third-party apps is tracking the flow of information from sensitive sources to insensitive sinks. Fernandes et al.[14] proposed a framework in which information flow control is combined with sandboxing. Bastys et al.[7] leveraged state-of-the-art information-flow trackers to control flows in JavaScript-driven user-automation

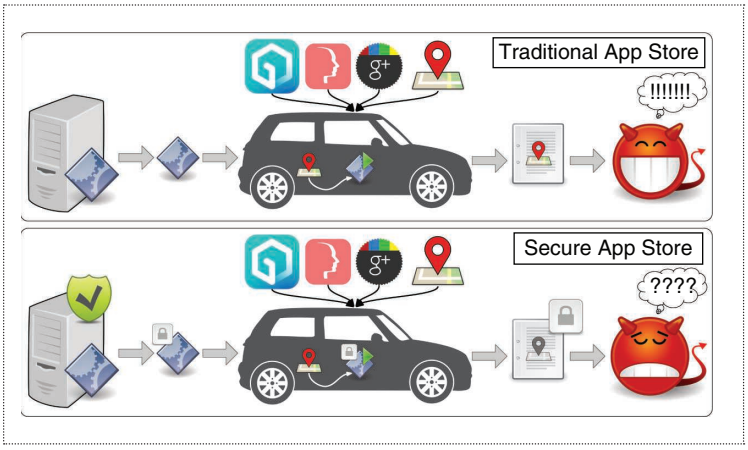

Figure 3. A traditional versus secure app store.

Take 4: Saving the Day

On her way home, Iona parks her car at a shopping mall, takes a picture of the season's first snow in a nearby park, and heads to the mall for some shopping. She receives an email with the map where her car is parked. The map is securely confined to her email. Her picture is securely backed up on Google Drive. Her thermostat turns on automatically, based on her proximity to home, maintaining a safe temperature range.

platforms. Moreover, program analysis techniques can be used to explore interactions among different IoT apps and uncover insecure cross-app interactions.

Other Countermeasures

Securing user-automation apps goes beyond the purely technical solutions that we have discussed so far. A user-automation app may simply describe a desirable functionality in the app's description text, such as "Automatically back up your new iOS Photos to Google Drive," while implementing a different functionality, such as "Unlock the door." As a result, a user may be tricked into installing an app the does not meet his/her original intentions. Similarly, even benign apps can yield unintended consequences whenever they are used in contexts that the user did not anticipate at the time of app's installation. Hence, it is important to raise the user awareness on the security and privacy risks that come with the apps. Recent techniques that use natural language processing to match the app's textual descriptions with the actual API are a promising approach in this direction; however, the context-dependent nature of user-automation apps ultimately requires the end user to evaluate the risks.

In-Vehicle Apps

Figure 3 contrasts the secure app store architecture with the traditional one. The latter allows apps to act on a user's behalf, implying risks, such as leaking user location to third parties. The former can instead analyze an app for insecure information flows, detecting such insecurities as tracking by third-party components. This can be achieved through robust permissions, API control, and information-flow control. A secure version of the app can be cleared for shipment to vehicles, enabling secure location-based services, such as finding nearby points of interest without leaking the driver's location to unauthorized parties.

At last, there is peace and quiet in Iona's life.

The Road Ahead

IoT security is hard, in general, because of the combination of heterogeneity, connectivity, limited resources, and device longevity.[15] The area of IoT apps brings additional challenges. Although users entrust their sensitive information to IoT apps, the IoT platforms thrive on third-party code. In the area of in-vehicle apps, safety challenges need to be addressed, as third-party apps are trusted resources and in control of the infotainment units.

Although the latest developments boost innovation and business potential, they also open up the possibility of large-scale, high-impact attacks by malicious app makers. The Cambridge Analytica privacy breach in the area of web and mobile security has demonstrated that threats by malicious third-party apps are real. Similar to the Facebook app for personality testing from Cambridge Analytica, users might be tempted to install, for example, an IoT app for carbon dioxide emission detection, which can maliciously exfiltrate user location information. This type of exfiltration can be extended to attack the service itself. For example, Uber's IFTTT APIs expose not only pickup and drop-off locations for each trip but also the driver's name, phone number, and photo as well as the car's license plate number. This opens up opportunities for stealthy profiling of Uber as a company, by building a detailed database of its drivers and vehicle fleet. Scenarios like this call for a principled approach to security, safety, and privacy of IoT apps.

We identify the following key challenges for securing IoT apps. Based on the limited current technology, there is a high demand to develop the following concepts and mechanisms.

- *Fine-grained access control* can regulate safe and secure use of sensitive resources. This needs to include fully

mediated mechanisms against bypassing by advanced attacks, such as resource exhaustion. Fine-grained access control connects to application- and user-level permissions as well as to API control.

- *Robust and usable permission models* are an important challenge. Regulating the granularity is especially important for location information. Bundling and automatically deriving user-level permissions is important to relieve users of the burden of understanding the technical inner workings of IoT apps.
- *API control* helps regulate safe and secure use of sensitive app functionalities. This goes beyond permission models to, for example, enforcing safe ranges for APIs, such as the sound volume in a vehicle and sharing the location only under certain temporal conditions.
- *Information tracking* mechanisms can keep track of how sensitive information is used by apps. This can, for example, help detect URL-based leaks. Such a mechanism can be used either during the vetting process or as a security monitor of a deployed app.
- *Secure architectures for app stores* can leverage the mentioned program analysis technology to automatically flag suspicious apps before they are released in the app store.

With these concepts and mechanisms in place, an important practical goal is to provide an open platform for standardization and technology transfer of secure app and web technologies to the IoT[1] and automotive[2] industries. ■

Acknowledgments

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program funded by the Knut and Alice Wallenberg Foundation. It was also partly funded by the Swedish Foundation for Strategic Research and the Swedish Research Council.

References

1. W3C, "W3C begins standards work on Web of Things to reduce IoT fragmentation," 2017. [Online]. Available: https://www.w3.org/WoT/
2. W3C, "Automotive and web at W3C," 2016. [Online]. Available: https://www.w3.org/auto
3. IFTTT, "One connection, countless possibilities," 2019. [Online]. Available: https://platform.ifttt.com/lp/learn_more
4. Google LLC, "Automotive." Accessed on: 2019. [Online]. Available: https://source.android.com/devices/automotive/
5. M. Rosenber, N. Confessore, and C. Cadwalladr, "How Trump consultants exploited the Facebook data of millions," *NY Times*, Mar. 17, 2018. [Online]. Available: https://www.nytimes.com/2018/03/17/us/politics/cambridge-analytica-trump-campaign.html
6. A. Greenberg, "The Jeep hackers are back to prove car hacking can get much worse," *Wired*, Aug. 1, 2016. [Online]. Available: https://www.wired.com/2016/08/jeep-hackers-return-high-speed-steering-acceleration-hacks/
7. I. Bastys, M. Balliu, and A. Sabelfeld, "If this then what? Controlling flows in IoT apps," in *Proc. 2018 ACM SIGSAC Conf. Computer and Communications Security (CCS)*, 2018, pp. 1102–1119. doi: 10.1145/3243734.3243841.
8. B. Eriksson, J. Groth, and A. Sabelfeld, "On the road with third-party apps: Security analysis of an in-vehicle app platform," in *Proc. 5th Int. Conf. Vehicle Technology and Intelligent Transport Systems (VEHITS)*, 2019, pp. 64–75.
9. Google LLC, "Permissions overview." Accessed on: 2019. [Online]. Available: https://developer.android.com/guide/topics/permissions/overview
10. 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 *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 1501–1510. doi: 10.1145/3038912.3052709.
11. Z. B. Celik, P. D. McDaniel, and G. Tan, "Soteria: Automated IoT safety and security analysis," in *Proc. USENIX ATC*, 2018, pp. 147–158.
12. E. Fernandes, A. Rahmati, J. Jung, and A. Prakash, "Decentralized action integrity for trigger-action IoT platforms," in *Proc. Network and Distributed System Security Symp. (NDSS)*, 2018. doi 10.14722/ndss.2018.23119.
13. P. D. McDaniel, N. Papernot, and Z. B. Celik, "Machine learning in adversarial settings," *IEEE Security Privacy*, vol. 14, no. 3, pp. 68–72, May 2016.
14. E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash, "Flowfence: Practical data protection for emerging IoT application frameworks," in *Proc. USENIX Security Symp.*, 2016, pp. 531–548.
15. P. C. van Oorschot, "Internet of Things security: Is anything new?" *IEEE Security Privacy*, vol. 16, pp. 3–5, Sept./Oct. 2018.

Musard Balliu is an assistant professor at the School of Electrical Engineering and Computer Science at KTH Royal Institute of Technology, Sweden. Balliu received a Ph.D. in computer science from KTH Royal Institute of Technology in Stockholm, Sweden, in 2014. Contact him at musard@kth.se.

Iulia Bastys is a doctoral student in the Department of Computer Science and Engineering at Chalmers University of Technology, Sweden. Bastys received an M.Sc. in computer science from the University of Saarland in Saarbrücken, Germany, in 2016. Contact her at bastys@chalmers.se.

Andrei Sabelfeld is a professor in the Department of Computer Science and Engineering at Chalmers University of Technology, Sweden. Sabelfeld received a Ph.D. in computer science from the University of Gothenburg, Sweden, in 2001. Contact him at andrei@chalmers.se.