Poster: Protecting Remote Controlling Apps of Smart-Home-Oriented IOT Devices

Eric Ruiz, Richard Avelar, Xiaoyin Wang
University of Texas at San Antonio
{eric.ruiz, richard.avelar, xiaoyin.wang}@utsa.edu

ABSTRACT

Smart home systems rely on cloud servers and multiple IOT (Internet Of Things) devices such as smart thermometers, video monitors, and smart appliances to realize convenient remote home control and monitoring. As smart home systems get more and more popular recently, the security protection of smart home systems has become an important problem. Among the components of a smart home system, the remote controlling mobile app is often the most vulnerable part as it is directly exposed to the public network. In this paper, we propose a novel tool, HomeGuard, to detect potential vulnerabilities in remote controlling apps of smart home system. Specifically, HomeGuard first identifies information flows related to sensitive control and data messages, and then checks where such information flows to and whether they are properly encrypted.

CCS CONCEPTS

• **Software and its engineering** → **Software verification and validation**;

KEYWORDS

IOT, Smart Home, Network Analysis

ACM Reference format:
Eric Ruiz, Richard Avelar, Xiaoyin Wang. 2018. Poster: Protecting Remote Controlling Apps of Smart-Home-Oriented IOT Devices. In *Proceedings of 40th International Conference on Software Engineering Companion, Gothenburg, Sweden, May 27-June 3, 2018 (ICSE '18 Companion),* 2 pages.
https://doi.org/10.1145/3183440.3195101

1 INTRODUCTION

According to recent studies [1], the market of smart home devices in USA are valued more than 700 million dollars 2015, and are expected to expand 18.7% per year till 2022. As more homes start to benefit from the convenience of smart homes, they also open a new attack surface of households, leaving them vulnerable to cybercriminals with economic, privacy, and physical security motivations. For example, malicious parties may peep or even hijack the sent / received control requests and find out the normal schedule of a family, whether nobody is at home, or even heat or cool a home to extreme temperatures for ransom [2]. While a study [3] shows that popular smart home devices (e.g., NEST, HoneyWell) are reasonably well secured, it also indicates that remote controlling apps are vulnerable to attacks from other apps on the smart phone and the cyber space.

To protect remote controlling apps, we develop a novel tool called HomeGuard, which examines whether sensitive data are properly encrypted and protected in energy management apps. HomeGuard contains an outbound analyzer that scans the app's code for sensitive information flowing out, an inbound analyzer that identifies sensitive information sent back to users such as monitoring information about home devices, and an encryption analyzer to find out whether data are properly encrypted. In our empirical evaluation, we applied HomeGuard to top 10 free smart home apps in the Google Play market, and it detected 187 sensitive data flows, 15 destinations, and 12 encryption methods.

2 APPROACH

2.1 Outbound Analyzer

HomeGuard's outbound analyzer statically checks for sensitive data flows out of the app. While data flow analysis can be supported by existing data flow analysis tools such as FlowDroid [4], and the sinks can be identified as network APIs (e.g., instantiation of `HTTPGet` objects), the major challenge is to locate the sources of sensitive data. In smartphone apps, user inputs are a major source of sensitive data collected, so we need to link user input fields (e.g., input boxes, time pickers) to sensitive information such as temperature and time. We first enumerate terms related to sensitive information by constructing a lexicon of smart home systems, which including control commands and data types, etc. This lexicon is generated from the user interface of a set of training smart-home apps. Then, for a new smart-home app, we calculate text similarity between terms in the lexicon and identifiers in the app's user interface, including view ids and view labels, to identify the view labels and view ids related to sensitive information. Finally, we apply our prior work GUILeak [9], which extends Gator [7] (a static estimator of runtime Android view hierarchy) with static string analysis to link Android input views with their possible labels, hints, and IDs in view hierarchies. Therefore, using view ids and labels as bridges, we can link user input fields with sensitive information terms in the lexicon, and use such input fields as sources in data flow analysis.

2.2 Inbound Analyzer

The basic idea of the inbound analyzer of HomeGuard is to statically estimate the possible contents of the requests sent to the network, and determine the content of the response based on the request. We first apply NetDroid [5, 6], which estimates possible values of HTTP requests based on static string analysis [10, 11], on the app to estimate all possible values of network requests (i.e., arguments of network API methods). Then, we use the lexicon constructed in Section 2.1 to identify the network requests requesting sensitive

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195101

Table 1: Basic Information of Studied Subjects

App	Package	# Flows	# Dest.	# Enc.
SmartThings	com.smartthings.android	37	3	1
Stringify	com.stringify.stringify	20	2	1
Wink	com.quirky.android.wink.wink	22	1	1
Gideon	mobile.alfred.com.alfredmobile	13	1	1
Yeti	com.netbeast.yeti	25	1	1
Vivint	com.vivint.vivintsky	12	2	1
AllTek	com.alltek.android.smarthome	20	1	2
Yonomi	com.yonomi	5	1	1
Loxone	com.loxone.kerberos	10	1	2
Rogers	com.ucontrol.activity	23	2	1

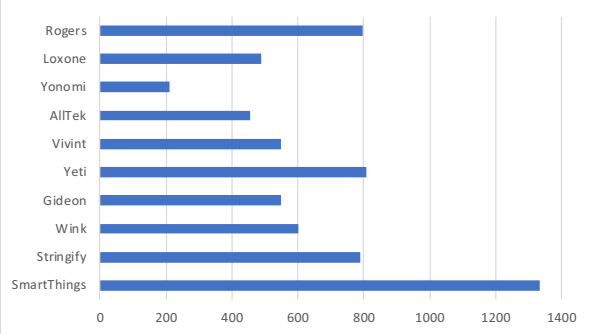

Figure 1: The Time Spent on Analyses for HomeGuard (in seconds)

information and their corresponding response variables (i.e., return values of the API method invocations executing these network requests). Finally we consider these response variables as sources and apply information flow analysis [8] to check whether such responses are sent to insecure destinations (e.g., SD card, system logs).

2.3 Encryption Analyzer

HomeGuard's encryption analyzer identifies encrypting API methods in Android framework, as well as other encryption methods based on method names. After the encryption methods are identified, HomeGuard uses conditional data flow analysis to check whether all data flows go though encryption [12]. Conditional information flow analysis will perform static analysis to examine whether the condition holds for all information flows from the source to the sink. Specifically, HomeGuard will further check the predicates guarding the information flows detected by FlowDroid and compare them with the condition defined in the set of encryption methods.

3 EMPIRICAL EVALUATION

In our evaluation, we apply HomeGuard to top 10 free smart home apps (the 10 highest ranked apps searching with "Smart Home") in Google Play market, while using the top 11-20 apps as the training set for lexicon construction. The basic information of the studied subject apps are shown Table 1. The columns 1-5 presents the app name, the app's package name, the number of detected sensitive information flows, the number of detected destinations, and the number of detected encryption methods.

To study the efficiency of HomeGuard, we further studied the time HomeGuard spent on analyzing each apps, and present the results in Figure 1. From the figure, we can see that, HomeGuard can finish its analysis on all real world smart home apps within 1400 seconds or about 25 minutes.

In our study we further studied the breakdown of sensitive information flows detected in the 10 top apps, and the result is presented

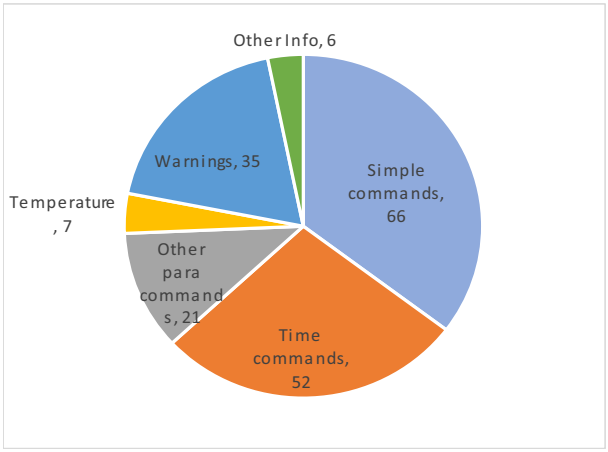

Figure 2: The Breakdown of Sensitive Information Flows Detected by HomeGuard

in Figure 2. From the figure, we can see that, the most common sensitive data flows are simple commands, such as turning on the light, and time commands, which represents a command sent to the smart home system with time, such as turn on the air-conditioning within 10 minutes or at 6pm. The third largest category includes warnings which are various information sent back from the smart home system to the user about the status of smart home systems such as the cook is done, or certain appliance is not working. The fourth time of sensitive information flows includes other commands with parameters such as set the room temperature to 70 Fahrenheit degree or set the washing machine to work with heavy load option.

REFERENCES

[1] Smart home market, http://www.businessinsider.com/the-us-smart-home-market-report-adoption-forecasts-top-products-and-the-cost-and-fragmentation-problems-that-could-hinder-growth-2015-9, 2017.
[2] Smart home security, https://motherboard.vice.com/article/internet-of-things-ransomware-smart-thermostat, 2017.
[3] Smart thermostat security: Turning up the heat, http://www.burrough.org/documents/thermostat-final-paper.pdf, 2017.
[4] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. *SIGPLAN Not.*, 49(6):259–269, 2014.
[5] S. Mostafa, R. Rodriguez, and X. Wang. Netdroid: Summarizing network behavior of android apps for network code maintenance. In *International Conference on Program Comprehension*, pages 165–175, 2017.
[6] R. Rodriguez, S. Mostafa, and X. Wang. Ntapps: A network traffic analyzer of android applications. In *SACMAT*, pages 199–206, 2017.
[7] A. Rountev and D. Yan. Static reference analysis for GUI objects in Android software. In *International Symposium on Code Generation and Optimization*, pages 143–153, 2014.
[8] H. Tang, X. Wang, L. Zhang, B. Xie, L. Zhang, and H. Mei. Summary-based context-sensitive data-dependence analysis in presence of callbacks. In *POPL*, pages 83–95, 2015.
[9] X. Wang, X. Qin, M. B. Hosseini, R. Slavin, T. D. Breaux, and J. Niu. Guileak: Tracing privacy policy claims on user input data for android applications. In *Proceedings of ICSE, To Appear,*, 2018.
[10] X. Wang, L. Zhang, T. Xie, H. Mei, and J. Sun. Transtrl: An automatic need-to-translate string locator for software internationalization. In *ICSE*, pages 555–558, 2009.
[11] X. Wang, L. Zhang, T. Xie, H. Mei, and J. Sun. Locating need-to-translate constant strings in web applications. In *FSE*, pages 87–96, 2010.
[12] H. Zhang, H. B. K. Tan, L. Zhang, X. Lin, X. Wang, C. Zhang, and H. Mei. Checking enforcement of integrity constraints in database applications based on code patterns. *The Journal of Systems & Software*, 12(84):2253–2264, 2011.