A Survey on Detecting Logical Errors of Trigger-Action Programming Rules on IoT Applications

Yicheng Zhen, *Yeonjoon Lee

Major in Bio Artificial Intelligence, Department of Computer Science and Engineering, Hanyang University, * Department of Computer Science and Engineering, Hanyang University

zyc0928 @hanyang.ac.kr, yeonjoonlee@hanyang.ac.kr

IoT 애플리케이션에서의 트리거-액션 프로그래밍 규칙의 논리적 오류 검출 연구동향

전이정, *이연준

한양대학교 컴퓨터공학과 바이오인공지능융합전공, *한양대학교 컴퓨터공학과

Abstract

The Internet of Things (IoT) is rapidly growing, and there is a dramatic increase in the number of smart interconnected devices. A large number of smart connected devices make IoT systems increasingly penetrating all aspects of human society, and the smart home is one of the representatives. However, while it facilitates life, it also brings some risks. So far, many studies have discovered serious threats to user security and the physical environment in which IoT devices are located due to different types of logical errors in TAP rules. As a result, it is crucial to consider the seriousness of the risks and to understand the research trends. In this paper, we propose research trends for work on detecting the different types of logical errors in TAP rules.

Ⅰ. Introduction

In the last few years, the Internet of Things (IoT) has been rapidly developing in all life sciences. IoT provides automation and smart control for users in different fields, such as smart homes [1].

With the diversification of smart home scenarios, personalization needs are increasing. Most of the programming tools that support the implementation of these personalized needs use Trigger-Action Programming (TAP).

The most popular automation model in the IoT is TAP [2], which enables users to link devices and services based on cause-and-effect relationships. It takes the form of." If a trigger is triggered, perform the corresponding action" (i.e. IF <trigger> THEN <action>). However, with the widespread use of TAP rules, the serious threat that a TAP rule with logical vulnerabilities may pose to user security and the physical environment in which IoT devices are set up deserves careful consideration.

In this paper, we explain the different types of logical errors in the Trigger-Action programming rules and the possible dangers to the device when these logical errors occur and present the methods for detecting logical errors.

Ⅱ. Method

2.1 Background

We can classify logical errors in TAP rules into two categories: logical errors between TAP rules of two applications and logical errors generated when non-programmer users define TAP rules.

2.1.1 Logical errors between TAP rules of two applications. The first is Conflict Trigger. Conflicting commands issued to the same device at the same time can result in conflicting triggers [4]. The attacker can set malicious rules or patterns and use action conflicts to prevent the normal execution of the device. The second is Continuous Trigger. When the action of rule 1 and the trigger of rule 2 is the same, it leads to continuous triggering. The last is Invalid Triggers. Invalid triggers result when the action of the first rule in a rule chain is opposite to the action of the last rule [2]. However, most devices may become unstable if the status is changed immediately.

2.1.2 Logical errors generated when non-programmer users define TAP rules. The first is Loop.

When a set of trigger action rules is activated continuously without reaching a steady state, the loop will occur. The second is Inconsistency. When several rules are executed simultaneously, a certain two rules with different triggers but opposite actions will produce two inconsistent actions because of the implicit triggering of other rules. The last is Redundancy. When several rules are executed at the same time, a certain two rules, which have different triggers but have the same action, will produce two redundant actions because of the implicit triggering of other rules.

2.2 Detecting logical errors between TAP rules of two applications

Chi et al [3] proposed the HOMEGUARD system and selected 90 applications for detection from 146 applications in the SmartThings application repository. HOMEGUARD uses a "symbolic execution" based rule extractor to extract triggers, actions, and conditions from each SmartThings application. A configuration collector is also used to collect configuration information for each SmartThings app, such as the app name, the device bound to the SmartThings app, and the static values within the app. After that, the system constructs constraints on the triggers and conditions and uses whether the constraints are satisfied to perform overlap detection. HOMEGUARD combines all constraints of both rules with additional device constraints; if the constraints are satisfied, the rules are logically incorrect. The result shows that HOMEGUARD found 663 instances of logical errors, and 101/146 SmartThings applications contained at least one error.

2.3 Detecting logical errors generated when non-programmer users define TAP rules

Corno et al.[4] proposed EUDebug. As a non-programmer user defines any troublesome or potentially dangerous behavior, EUDebug can assist the non-programmer user in identifying rule conflicts and performing step-by-step simulations of problematic rules to further examine and understand the problem. Corno et al. defined a novel Semantic Colored Petri Net (SCPN) formalism, which can model and check trigger-action rules. In particular, EUDebug adds triggers and actions in the form of high-level semantic representations of triggers and actions EUDebug adds to the SCPN (Semantic Colored Petri Net) graphs to semantically classify triggers and actions. By traversing the SCPN graph, conflicts can be detected if actions/triggers executed on the same IoT device or web service are categorized under different classes. The effectiveness of the approach insists evaluated by tracking TAP rules made by non-programmer users in the field. The result shows that "issue checking" helped non-programmer users to correct 77.81% of rules with errors, and when it comes to "step-by-step explanations" of errors, this percentage increased to 83.73%.

Ⅲ. Conclusion

With the rapid development of smart homes, end-user programming in the smart home field is gradually becoming popular. To cope with this trend, TAP has become the primary choice for users nowadays.

In this paper, we briefly explain the different types of logical errors in the TAP rules and introduce the role and main principles of HOMEGUARD and EUDebug. The experimental results of the experiments done by Chi et al. and Corno et al. prove that both can detect logical errors between rules with the help of HOMEGUARD and EUDebug. The differences between the two methods as TABLE I.

ACKNOWLEDGMENT

This work was partly supported by Institute of Information & communications Technology Planning &Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2022-00155885, Artificial Intelligence Convergence Innovation Human Resources Development (Hanyang University ERICA)) and Institute of Information & communications Technology Planning& Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2020-0-01343, Artificial Intelligence Convergence Research Center(Hanyang University ERICA)).

REFERENCES

[1] Abuserrieh L, Alalfi M H. A Survey of Analysis Methods for Security and Safety verification in IoT Systems[J]. arXiv preprint arXiv:2203.01464, 2022.

[2] Wang Q, Datta P, Yang W, et al. Charting the attack surface of trigger-action IoT platforms[C]//Proceedings of the 2019 ACM SIGSAC conference on computer and communications security. 2019: 1439-1453.

[3] Corno F, De Russis L, Monge Roffarello A. Empowering end users in debugging trigger-action rules[C]//Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. 2019: 1-13.

[4] Chi H, Zeng Q, Du X, et al. Cross-app interference threats in smart homes: Categorization, detection and handling[C]//2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2020: 411-423.

	Platform	Pro-processing	Method
HOMEGUARD	SmartThings application	The static code analysis	Based on semantics
EUDebug	IFTTT applet	Semantic representation translation algorithm	Graph traversing

TABLE I: Differences between the two methods