Automatic Trigger Generation for End User Written Rules for Home Automation

Chandrakana Nandi
Department of Computer Science and Engineering
University of Washington, Seattle, USA
cnandi@cs.washington.edu

ABSTRACT

To customize the behavior of a smart home, an end user writes rules. When an external event satisfies a rule's trigger, the rule's action executes; for example, when the temperature is above a certain threshold, then window awnings might be extended. End users often write incorrect rules. This paper's technique prevents a certain category of errors in the rules: *errors due to too few triggers*. It statically analyzes a rule's actions to automatically determine a set of necessary and sufficient triggers.

We implemented the technique in a tool called TrigGen and tested it on 96 end user written rules for openHAB, an open-source home automation platform. It identified that 80% of the rules had fewer triggers than required for correct behavior. The missing triggers could lead to unexpected behavior and security vulnerabilities in a smart home.

CCS Concepts

•**Security and privacy** → **Logic and verification;** *Usability in security and privacy;* •**Software and its engineering** → **Formal software verification;** *Domain specific languages;*

Keywords

Static analysis; Security; Smart homes; Trigger action programs

1. RESEARCH PROBLEM

Most home automation platforms support end user customization components: a user writes rules to determine what actions should be taken by what device under what conditions [12]. Listing 1 shows an example of such a rule. A rule has two main components: triggers that cause a rule to be fired and actions to be executed when a rules fires. This is also called Trigger Action Programming (TAP) [15]. Even though TAP is the most commonly used and practical approach for home automation [15, 5], end users often make errors in writing trigger-action programs [8, 16]. In a smart home with multiple interacting devices, an error in one rule can cause unexpected behavior or security vulnerabilities in another part of the house.

FSE'16, November 13–18, 2016, Seattle, WA, USA
ACM. 978-1-4503-4218-6/16/11...$15.00
http://dx.doi.org/10.1145/2950290.2983965

```
rule "Away rule"
  Item State_Away changed //trigger
then // action block
  if(State_Away.state == ON) {
    if(State_Sleeping.state != OFF) {
      postUpdate(State_Sleeping,OFF)
    }
  }
end
```

Listing 1: End user written rule for ensuring that the Away and Sleeping states in the house are not ON at the same time.

Consider the rule in listing 1 written by an end user. It is supposed to ensure that `State_Away` and `State_Sleeping` are not `ON` simultaneously. If `State_Sleeping` is `ON`, it indicates that the home inhabitants are sleeping in the house. If `State_Away` is `ON`, it indicates that the home inhabitants are away. The rule fires when `State_Away` changes but not when `State_Sleeping` changes. This allows both values to be set at the same time, which deviates from the expected behavior of the rule. The correct version of the rule should include *both* `State_Away` and `State_Sleeping` as triggers.

This example shows that even if the action block of a rule is implemented correctly, not having all the correct triggers can lead to too few firings of the rule and thus, unexpected behavior. This paper addresses such errors in writing triggers for rules. Our approach eliminates a certain category of error in the rules—*errors due to too few triggers*. We have built a static analysis that determines a necessary and sufficient set of event-based triggers in the rules. These inferred triggers can be compared to the user-written triggers to indicate whether there are too few triggers or unnecessary triggers. The approach could also free users from the need to write triggers. We have implemented the technique as a tool— TrigGen—and used it to analyse 96 home automation rules written by end users. TrigGen found that 80% of the rules had insufficient triggers.

2. RELATED WORK

Security loopholes in smart homes have been investigated in several papers [4, 6, 14, 11]. Recently, Fernandes et al. [7] did a security analysis of several apps based on Samsung SmartThings and discovered that many of them unnecessarily granted full access to the devices in the house. While they aimed at identifying security flaws in the SmartThings framework itself, our aim is to assist end users in writing correct automation rules. Further, instead of analysing apps for home automation, we analysed end user written rules.

The usefulness of TAP for customizing smart homes has been studied by Ur et al. [15] and Dey et al. [5]. They showed by conducting user studies that about 80% of the automation requirements of the users could be represented by trigger action programs [5]

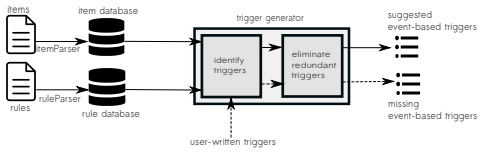

Figure 1: Design of TrigGen.

and even non-programmers could easily learn TAP [15]. Huang et al. [8] conducted user studies on TAP to identify inconsistencies in interpreting the behavior of trigger action programs and errors in writing them. They found that often the interpretation of a rule by a user is different from the semantics of the rule.

Some work has been done on detecting conflicts in trigger-action programs [9, 10, 13]. These approaches assume that the triggers are already provided by the user while TrigGen is capable of automatically generating the correct triggers for a rule which makes it unique and different from previous work.

3. APPROACH

We have developed a static technique to automatically generate correct event-based triggers from the actions written by the end users. Figure 1 shows the design of our tool. It takes as input a *rule* file which contains all the end user written rules and an *item* file which contains a list of all the smart devices (or items) installed in the home. Both rules and items have their own domain specific language (DSL).

The rules are parsed to extract the action block and the items are parsed to extract all the item names, types (`Switches`, `Shutters`, `Dimmers` etc.), and the groups to which they belong—for example all the light switches in the living room may be grouped together.

By statically analysing the abstract syntax tree (AST) of the action block A of rule R, TrigGen first identifies all the items that appear in R. This is an exhaustive list of all potential event-based triggers. Let this list be T. Naively adding all $t \in T$ as triggers of R would unnecessarily fire R too many times. Hence, we apply an elimination technique to get rid of triggers that are not required. For that, we define the following terms.

Definition 1. An item is *live* [3] in R if its value may be read before it is written to, in the action block, A.

Definition 2. An event-based trigger is *redundant* if inclusion of the trigger in a rule *never* changes the state of any item or the value of any variable involved in A when the rule is fired due to it.

Trigger elimination strategy. **An item d which is not *live* in R is not included in T because it is *redundant*.**

Enumerating groups. Items can be grouped so that they can be controlled together if needed—the action block of the rule can perform operations on all the member items of a group. Listing 2 shows examples of groups in an *item* file and a group being used in a *rule* file.

```
Group All
Group GF(All)
Group GF_Living(GF)
Group Lights(All)
Dimmer Light_GF_Living(GF_Living,Lights)

---

Lights.members.forEach(light| postUpdate(light,ON))
```

Listing 2: Code snippet from an items file (above) showing grouping of items (GF stands for Ground Floor) and use of a group in a rule (below).

A group can have multiple items and one item can belong to one or more groups. Groups can also be contained in other groups. For such rules, TrigGen generates triggers by enumerating all members of a group. It performs a *depth first search* to determine all the groups to which an item belongs. Figure 2 shows the graph representation of the groups in listing 2.

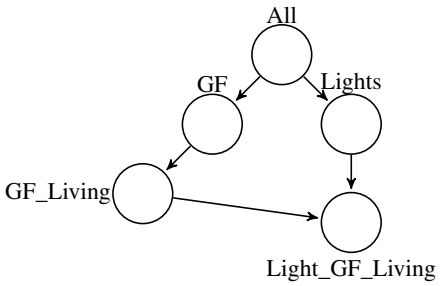

Figure 2: Graph representing the item grouping shown in listing 2. The arrows point from a parent group to a child group or a member item.

4. EXPERIMENTAL RESULTS

We evaluated TrigGen on 96 end user written rules. We obtained the rules from links provided on the openHAB website [1, 2]. For 91 out of the 96 rules (95%), TrigGen suggested a list of all required event-based triggers. TrigGen also compared its output with the user-written event-based triggers and found that for 77 rules (80%), the user-written triggers were insufficient—it identified the missing ones for these rules. Figure 3 shows the output of TrigGen for suggesting triggers (in white) and detecting missing triggers (in black).

Scope. Since TrigGen uses the AST of the action block to infer the triggers, if there is no reference to an item in the script, TrigGen cannot detect that trigger. We observed this for 5 out of the 96 rules (5%).

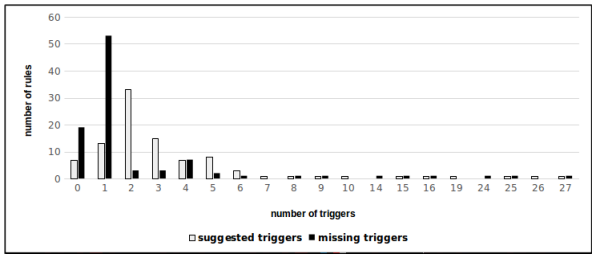

Figure 3: Summary of the output of TrigGen.

5. CONCLUSIONS AND FUTURE WORK

End users have a major role in home automation—they decide the automation rules for the devices. Not surprisingly, our research shows that these end user written rules are often error-prone. We observed that a common error made by end users while writing the rules is having insufficient number of triggers, leading to fewer firings of the rules than necessary. To prevent this problem, we developed TrigGen that can automatically generate a set of necessary and sufficient triggers based on the actions in a rule. This also reduces the burden of the end users by assisting them in writing the rules correctly.

We evaluated TrigGen on real home automation rules written by end users and found that out of 96 rules, 77 (80%) had insufficient number of triggers. By manually inspecting the suggestions provided by the tool, we found that each one of them was relevant and including them in the rules ensured consistent behavior.

6. REFERENCES

[1] Configs, Tools & Icons. http://www.intranet-of-things.com/software/downloads. Accessed: May 2016.

[2] openhab downloads. http://www.openhab.org/getting-started/downloads.html. Accessed: May 2016.

[3] F. E. Allen and J. Cocke. A program data flow analysis procedure. *Commun. ACM*, 19(3):137–, Mar. 1976.

[4] T. Denning, T. Kohno, and H. M. Levy. Computer security and the modern home. *Commun. ACM*, 56(1):94–103, Jan. 2013.

[5] A. K. Dey, T. Sohn, S. Streng, and J. Kodama. icap: Interactive prototyping of context-aware applications. In *Proceedings of the 4th International Conference on Pervasive Computing*, PERVASIVE'06, pages 254–271, Berlin, Heidelberg, 2006. Springer-Verlag.

[6] N. Dhanjani. *Abusing the Internet of Things: Blackouts, Freakouts, and Stakeouts*. O'Reilly Media, Incorporated, 2015.

[7] E. Fernandes, J. Jung, and A. Prakash. Security Analysis of Emerging Smart Home Applications. In *Proceedings of the 37th IEEE Symposium on Security and Privacy*, May 2016.

[8] J. Huang and M. Cakmak. Supporting mental model accuracy in trigger-action programming. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, pages 215–225, New York, NY, USA, 2015. ACM.

[9] H. Luo, R. Wang, and X. Li. A rule verification and resolution framework in smart building system. In *Parallel and Distributed Systems (ICPADS), 2013 International Conference on*, pages 438–439, Dec 2013.

[10] C. Maternaghan and K. J. Turner. Policy conflicts in home automation. *Comput. Netw.*, 57(12):2429–2441, Aug. 2013.

[11] S. Mennicken, J. Vermeulen, and E. M. Huang. From today's augmented houses to tomorrow's smart homes: New directions for home automation research. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14, pages 105–115, New York, NY, USA, 2014. ACM.

[12] M. W. Newman. Now we're cooking: Recipes for end-user service composition in the digital home, 2006. Position Paper–CHI 2006 Workshop IT@Home.

[13] Y. L. Sun, X. Wang, H. Luo, and X. Li. Conflict detection scheme based on formal rule model for smart building systems. *IEEE Trans. Human-Machine Systems*, 45(2):215–227, 2015.

[14] B. Ur, J. Jung, and S. Schechter. The current state of access control for smart devices in homes. In *Workshop on Home Usable Privacy and Security (HUPS)*. HUPS 2014, July 2013.

[15] B. Ur, E. McManus, M. Pak Yong Ho, and M. L. Littman. Practical trigger-action programming in the smart home. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 803–812, New York, NY, USA, 2014. ACM.

[16] B. Ur, M. Pak Yong Ho, S. Brawner, J. Lee, S. Mennicken, N. Picard, D. Schulze, and M. L. Littman. Trigger-action programming in the wild: An analysis of 200,000 ifttt recipes. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 3227–3231, New York, NY, USA, 2016. ACM.