Poster: Attack Surface Modelling in Trigger Action Platforms

Pubali Datta, Adam Bates
Department of Computer Science
University of Illinois at Urbana-Champaign
{pdatta2, batesa}@illinois.edu

I. PROBLEM DESCRIPTION

The use of end-user programming in form of trigger-action rules is becoming increasingly popular with the advancement of Internet of Things (IoT) and smart technologies. Home automation platforms like IFTTT, Zapier, Microsoft Flow [1] are popular amongst users for their ease of use. These platforms let users create new functions in their homes by stitching different IoT devices and online services in the form of simple rules in natural language, e.g., “If humidity goes over 55% send notification to open window”. While these platforms can be used easily, they also pose great security risks through interplay of several rules installed in a smart home.

Predominantly mentioned platforms are closed where rule behaviors are described in only natural language making it difficult to observe rule interactions leading to security issues. Previous works [2] have focussed on detecting integrity and secrecy violation at rule level and user level with a significant amount of manual work to process natural language rule descriptions. However, the attack surface of rules installed in a smart home using such platforms were not discussed previously. This work aims to process natural language rule descriptions automatically, without any manual effort, into an information flow (IF) graph that describes the attack surface of the smart home. Moreover, our technique enables an user to perform reachability queries over the IF graph to find sources of flows to a specific device or service installed in the smart home.

II. PRELIMINARY EXPERIMENTS

To get an idea of how trigger-action rules interact with each other, we obtained a dataset (2016) of trigger-action rules from the popular platform IFTTT [3]. Individual rules in this dataset are in form of “If Trigger, Then Action”, where each trigger and action is linked to some channel. As an example, in “If humidity goes over 55% send notification to open window”, trigger ‘humidity goes over 55%’ is linked to channel Netatmo Weather Station and action ‘send notification to open window’ is linked to channel IFTTT notifications. In IFTTT platform, each rule is called a recipe (from 2017, each rule is called an applet and channel is renamed as service).

We parsed individual recipes in an information flow (IF) graph as following: Channel → Trigger → Recipe → Action → Channel. Then we combined IF graphs for all recipes in a single IF graph. As evident from this single recipe graph, tracking flow from trigger to action is trivial, because the flow lies in the rule itself. But tracking flow from an action to other possible triggers it could invoke is non-trivial since the descriptions of actions and triggers are in natural language. So, the simplest way to model inter-rule flows is to link a service’s inbound actions to all outbound triggers, which leads to state explosion and spurious flows in the graph (Figure 1). Unfortunately, many of the flows in this graph shows false flows that do not describe actual rule interaction. For example, Smartthings’s "Activate Device” action would not trigger “Presence Detected”; device activation and presence detection are actually two independent features that can be manipulated through this service. It is thus apparent that in order to obtain an accurate and precise attack surface graph, we must decompose channels into their underlying components in order to identify true inter-rule flows.

III. SYSTEM DESIGN

We present the system architecture of NLP-aided information flow analysis system for trigger-action platforms in Figure 2. We strive to automatically find precise information flow from action to trigger using existing natural language techniques [5]. IFTTT platform website was scraped to collect recent dataset of applets and service descriptions, 512 Services and 18892 applets were obtained. These text descriptions act as an input to our NLP engine.

Syntactic Analysis. Using NLP techniques, text descriptions of actions and triggers are tagged using parts-of-speech.
Fig. 2. NLP-aided Information Flow Analysis of Trigger-Action Platforms.

(POS) tags and subsequently dependency parsing is employed to create a dependency tree for the sentence representing grammatical relationships. Using these dependency relationships and predefined heuristics, syntactic elements can be extracted from the description, which are - root verb (task), the main object and the properties of the object. Due to similarity in descriptions of actions and triggers inside a single service, syntactic element matching results in detection of flow from action to trigger.

Semantic Analysis. Unfortunately, syntactic analysis alone is insufficient to detect all inter-rule flows. This is because trigger and action text descriptions often refer to semantically-related objects (e.g., temperature and thermostat) without referencing the exact same object. This calls for semantic analysis technique like word to vector embedding [6] to compute semantic similarity between words. If the similarity surpasses a predefined threshold between main objects of an action and a trigger, then we determine a flow from said action to trigger.

Using the detected flows in NLP engine, the spurious flows in IF graph in Figure 1 can be pruned leading to a highly precise graph capturing correct flows among the set of automation rules. This IF graph can be leveraged by an user to perform queries e.g., “Which services can affect my door lock?” and retrieve the attack surface of the specified door lock.

IV. RESULTS

We manually inspected action and trigger descriptions of services scraped from the IFTTT platform website to obtain ground truth. Then we employed our tool and compared the number of flows detected to the manually computed results, as shown in Table I. The semantic analyzer is still under development causing our system to miss several semantic flows leading to low number of true positives. The false positive occurs when object properties are specified using grammar rules our system do not monitor, leading to a false match.

TABLE I
SUMMARY OF ERROR RATES.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Observed Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#True Positive</td>
<td>122</td>
</tr>
<tr>
<td>#False Negative</td>
<td>152</td>
</tr>
<tr>
<td>#False Positive</td>
<td>1164</td>
</tr>
<tr>
<td>#True Negative</td>
<td>5251</td>
</tr>
<tr>
<td>PP rate $\alpha$ (type I error)</td>
<td>0.18</td>
</tr>
<tr>
<td>FN rate $\beta$ (type II error)</td>
<td>0.26</td>
</tr>
<tr>
<td>#Total flows (naive strategy)</td>
<td>6089</td>
</tr>
<tr>
<td>#Total flows (NLP-aided strategy)</td>
<td>1580 (23%)</td>
</tr>
</tbody>
</table>

V. FUTURE WORK

Refining the semantic analyzer to improve the accuracy for flow detection is ongoing. Additionally, to test the effectiveness of our system to find security vulnerabilities in real environment, representative smart home configurations are required. Since this information is proprietary to the platforms, generating practical models of rule configurations in user homes is another important thread of work.

ACKNOWLEDGMENT

The authors would like to thank Qi Wang and Dr. Carl Gunter from University of Illinois, Urbana- Champaign for their helpful suggestions and resources.

REFERENCES