Towards Usable Security Analysis Tools for Trigger-Action Programming

McKenna McCall[1] *mckennak@cmu.edu*
Eric Zeng[1] *ericzeng@cmu.edu*
Faysal Hossain Shezan[2] *fs5ve@virginia.edu*
Mitchell Yang[1] *mfy@andrew.cmu.edu*
Lujo Bauer[1] *lbauer@cmu.edu*
Abhishek Bichhawat[3] *abhishek.b@iitgn.ac.in*
Camille Cobb[4] *camillec@illinois.edu*
Limin Jia[1] *liminjia@cmu.edu*
Yuan Tian[5] *yuant@ucla.edu*

[1]*Carnegie Mellon University* [2]*University of Virginia* [3]*IIT Gandhinagar* [4]*University of Illinois Urbana-Champaign* [5]*University of California, Los Angeles*

Abstract

Research has shown that trigger-action programming (TAP) is an intuitive way to automate smart home IoT devices, but can also lead to undesirable behaviors. For instance, if two TAP rules have the same trigger condition, but one locks a door while the other unlocks it, the user may believe the door is locked when it is not. Researchers have developed tools to identify buggy or undesirable TAP programs, but little work investigates the usability of the different user-interaction approaches implemented by the various tools.

This paper describes an exploratory study of the usability and utility of techniques proposed by TAP security analysis tools. We surveyed 447 Prolific users to evaluate their ability to write declarative policies, identify undesirable patterns in TAP rules (anti-patterns), and correct TAP program errors, as well as to understand whether proposed tools align with users' needs. We find considerable variation in participants' success rates writing policies and identifying anti-patterns. For some scenarios over 90% of participants wrote an appropriate policy, while for others nobody was successful. We also find that participants did not necessarily perceive the TAP anti-patterns flagged by tools as undesirable. Our work provides insight into real smart-home users' goals, highlights the importance of more rigorous evaluation of users' needs and usability issues when designing TAP security tools, and provides guidance to future tool development and TAP research.

USENIX Symposium on Usable Privacy and Security (SOUPS) 2023.
August 6–8, 2023, Anaheim, CA, USA

1 Introduction

Platforms like IFTTT [13], SmartThings [20], and Home Assistant [10] allow users to create *home automations* to configure their smart homes. Home automations are often expressed as *trigger-action programming* (TAP) rules, which are an accessible way for people without programming experience to customize their smart-home devices [22]. A typical TAP rule format is: "IF *trigger* THEN *action*", where the *trigger* is an event that causes the *action*. For example, "IF Alice leaves home THEN lock door" locks the door whenever Alice leaves her home. More complex TAP rules can include conditional triggers or trigger multiple actions.

Users can accomplish more complex goals by writing multiple TAP rules, which we call TAP *programs*. When TAP rules are executed, they can generate events or alter the home environment, which may trigger other rules. For instance, consider the rules: "IF the user nears home THEN unlock door" and "IF door is unlocked THEN turn off security camera". Once the user reaches home, the first rule is triggered, and the resulting action (unlock door) triggers the second rule, causing the camera to stop recording.

Security and privacy risks of TAP Research has shown that users struggle to write complex TAP programs [12, 28] and reason about their behavior [12, 26], leading to problems ranging from safety risks like leaving doors unlocked [4, 5] to privacy risks like leaking sensitive data [3, 21]. Consider a situation where Bob has installed the rules "IF the time is 7PM THEN lock door" and "IF Bob arrives at home THEN unlock door". Here, Bob might mistakenly believe that the first rule will re-lock his door after he comes home at 8PM, a misunderstanding that could pose a safety risk. Further, even if Bob realizes there is a problem, he might have trouble finding a solution.

TAP security analysis tools To address these concerns, researchers have proposed tools to diagnose TAP program problems. Some tools detect *TAP anti-patterns*: recurring structures of TAP rules that can lead to unexpected or problematic behaviors [4, 5, 7, 16, 18, 24, 27]. Examples of anti-patterns

include rules that trigger conflicting behaviors in the same device, rules that might trigger each other (loops), and multiple rules triggering the same action.

Other tools verify that TAP programs adhere to *declarative policies* specified by users: the user specifies how they want their smart home to behave, (e.g., the front door should be locked at night), and the tool checks that no TAP rules violate the policy [2, 4, 5, 14, 15]. A proposed approach to specify policies are fill-in-the-blanks *policy templates* [16, 28]. For instance, Bob might write the policy "Door is unlocked should never be active while the time is after 7PM", where "_ should [always/never] be active while _" is the policy template.

However, there has been little work on whether these tools are easy to use and whether they satisfy users' actual needs. For example, it is not known which of the TAP anti-patterns identified by prior work are of concern to users, or which of the proposed templates for specifying declarative policies are easiest for users to understand and fill out correctly.

Research questions To gain insight into which TAP analysis tool user-interface approaches work well and which may need more refinement, we conducted an exploratory survey. We recruited 447 smart-home users from Prolific to study their perceptions of the problems that TAP security analysis tools address and the usability of the approaches proposed by these tools. In particular, we investigate the following questions:

- **RQ1**: What are smart-home users' motivations and goals for using TAP rules?
- **RQ2**: Can users successfully write *declarative policies* using the templates proposed in prior work? Which templates do people use most successfully, and which ones do people most prefer?
- **RQ3**: Given a TAP program, can users identify the *TAP anti-patterns* from prior work? Do users perceive anti-patterns as undesirable, and would they be interested in using a tool to find them?
- **RQ4**: What information should a tool provide that would be most helpful to aid users fixing policy violations or removing anti-patterns found in their TAP programs?

Overall, we show that smart-home users have the necessary skills to use TAP security analysis tools, but there is room for improvement. Participants were moderately successful at specifying declarative policies using templates, but struggled with some template formats. Participants also varied in their abilities to identify TAP anti-patterns. While some anti-patterns were viewed as desirable, participants generally agreed that it would be helpful to have a tool to identify them. Participants were most successful at repairing buggy TAP programs and completing partial programs that violated declarative policies when they were given some additional context about *the state of the home* when the policy violation would occur. Based on our results, some important shortcomings of these tools include confusingly worded policy templates and inconsistencies between user mental models of TAP rules and how they are modeled in the tools. Our findings provide guidance for both developers of TAP security analysis tools and researchers investigating the security and usability of TAP.

2 Background and Related Work

We broadly categorize TAP tools as ones that verify custom *declarative policies*, and ones that look for high-level *TAP anti-patterns*; several tools do both [4, 5, 16, 18, 27].

Tools with declarative policies Tools such as SIFT [15], Soteria [4], Salus [14], and AutoTap [28] typically require the user to specify which devices and TAP rules they have in their smart home, as well as how they want their devices to behave (a *policy*). The tool then checks the policy against a formal model of the user's home setup. Some tools perform this verification at run time [5], notifying users of potential violations as they interact with devices and trigger TAP rules.

The interfaces to specify policies vary in complexity between tools. Some tools require policies to be specified in a formal logic [4, 18, 27]. Others allow the user to write a policy as a series of conditions that should never happen [15] or via fill-in-the-blanks templates [16, 28], which are then translated to a more formal language. We focus on templates, as they require less training than writing policies in formal logic.

Chaining together smart devices through TAP programs leads to diverse and complex functionality, but also means that finding the source of a problem with a TAP program can be difficult [25]. Upon finding a policy violation, some tools supply a counterexample to help with debugging [4]. Others give hints about what led to the violation, like the rules responsible for the problem [16, 18]. In our study, we investigate what type of feedback is most helpful to the user trying to fix problems: identifying rules involved in the violation, supplying a full trace leading up to the violation, or something in between. Some tools go further and automatically synthesize fixes to suggest to the user [14, 28]. To narrow the scope of our study, we don't evaluate synthesis techniques but focus instead on what information tools can give users to help them understand the causes of violations and how to solve them. For tools that automatically fix problems, our results could still be relevant because users still need to specify a property (RQ2) and may benefit from additional feedback to help them understand what problem is being addressed (RQ4).

TAP anti-patterns Research has identified *TAP anti-patterns* [7, 11, 24] that can lead to unsafe or unpredictable behavior [1]. For instance, Bob's program from Section 1 exhibits an anti-pattern (*Opposite Behaviors* in Table 2) frequently identified in TAP research as problematic: if two rules trigger simultaneously with conflicting actions (door unlock and door lock) the result becomes unclear, which could lead to confusing or unsafe situations (such as Bob's door being unlocked when he believes it is locked). Other research iden-

tified potential confidentiality and integrity violations in TAP programs [3, 9, 21] and anti-patterns that could be leveraged in attacks to force devices into an insecure or unsafe state [6].

Research has also examined the prevalence of these potentially harmful anti-patterns. Some examined random or hand-crafted programs built from publicly available TAP rules [9, 21, 24], while others set up real devices to evaluate tools and test their attacks [6, 7, 15, 24]. Some prior work collected TAP programs from a small number of real users to evaluate [8, 11]. While many tools are capable of detecting various patterns, it is unclear whether users would understand what these patterns are, or if they find them undesirable. Therefore, we design tasks to measure understanding and perception of a selection of anti-patterns from prior work.

Usability of TAP tools The usability of a small number of proposed TAP analysis tools has been evaluated individually [15, 16, 28]; in contrast, we compare *techniques* employed by various tools from the TAP literature, examining the potential utility and efficacy of different approaches to specifying and debugging TAP programs and their properties.

3 Study Goals and Survey Design

We next describe the motivation for our study and how it informed the design of our surveys. We describe the specific procedures for administering the surveys in Section 4. Our goal is to examine the user-interaction approaches suggested by existing TAP analysis tools to determine what works well, where refinement is needed, and how tools might better serve users. To explore the breadth of the design space, we conduct a survey-based user study. We divided our study into four parts, which correspond to our research questions (Section 1).

3.1 Part 1: Identifying user needs

What are users trying to accomplish with their TAP programs? Since the space of tools and problems is large and varied, knowledge of users' goals can help inform priorities for tool design. For instance, if users are generally installing TAP rules without much consideration for what they expect their devices to do (or not do), they might find writing declarative policies difficult, and might instead benefit from a tool that looks for programming patterns that may cause problems.

We asked users about their high-level priorities for their TAP programs by rating the importance of several goals — Home Safety, Home Security, Privacy, Comfort and Convenience, Understanding Failures, and Fun — on a four-point Likert scale. We also asked them to describe any goals they had for their smart home in a free-response field.

3.2 Part 2: Writing declarative policies

Some tools allow users to specify how they want their devices to operate and check for violations of these declarative policies. There are a few ways to specify policies; an approach suggested by tools like AutoTap [28] and SafeTAP [16] is fill-in-the-blanks policy *templates*. However, these templates could be confusing to users, because they use complicated constructions such as "[state] should [always/never] be active while [state]" (see Table 1 for a full list).

So, we ask, can users correctly express their goals using this kind of interface? And, if not, what factors might be contributing to their problems? We break these questions down into the following three survey tasks.

Task 2a: Picking templates Can users pick a template format that matches a high-level goal? There is some variation between AutoTap [28] and SafeTAP [16] templates, but both tools have multiple templates that users need to pick from to specify their policies. While some templates are equivalent (like AutoTap's "_ should [always/never] be active" and SafeTAP's "I [always/never] want _"), there are important differences between others, like whether the policy is conditional or includes timing constraints, and other subtleties between (instantaneous) events and (persistent) states.

To investigate this, we presented participants with a scenario and goal, and asked them to choose which of the AutoTap/SafeTAP templates were most appropriate for the goal. Participants could choose any template from Table 1. Participants repeated this task for two scenarios.

Task 2b: Filling out templates Can users correctly fill out a policy template, once one has been correctly selected? To write policies understandable by a tool, a user may have to think of the goals they have for their home in terms of specific device behaviors or conditions since the tool likely will not understand what it means for a home to be "warm" or "safe." Some devices have simple boolean states, like "dryer on" or "dryer off," but others may require more specificity, like temperature or time.

Participants were presented a scenario describing a goal and a template, and their task was to fill in the blanks for template using drop-down menus of states or events. This task was repeated three times, for the following scenarios: closing the windows when it rains, setting the thermostat at night, and keeping the dryer off during work hours.

Task 2c: Clarity of templates Because there may be several equivalent ways to write a policy, we want to know which templates make most sense to users.

Policy templates themselves might be logically equivalent, like "_ should [always/never] be active" and "I [always/never] want _", or people might prefer one way of filling out a particular template over another, like "Dryer on should never be active" and "Dryer off should always be active." We are interested to know whether there are any templates that are especially popular, or if our participants tend to prefer equivalent AutoTap templates or SafeTAP templates, or the positive ("always") or negative ("never") form of the templates.

In this task, participants were shown a goal and four poli-

	Name	Template
AutoTap [28]	One-State Unconditional	[state] should [always/never] be active
	One-State Duration	[state] should [always/never] be active for more than [duration]
	Multi-State Unconditional	[state] and [state] should [always/never] occur together
	State-State Conditional	[state] should [always/never] be active while [state]
	Event-State Conditional	[event] should [only/never] happen when [state]
	Event-Event Conditional	[event] should [always/never] happen within [duration] after [event]
SafeTAP [16]	Whenever	Whenever [event] make sure that [state]
	Only When	[event] only when [state]
	Always/Never	I [always/never] want [state]

Table 1: Name and format of each policy template evaluated in the study.

cies describing the goal and were asked to pick the policy they felt was most natural. Participants were randomly shown three of five possible scenarios.

3.3 Part 3: Identifying TAP anti-patterns

Another class of TAP security analysis tools check for anti-patterns in TAP programs. An example of an anti-pattern is a loop, where one rule triggers a second rule, the second rule triggers the first, and so on.

We ask: Would users find these tools useful? How well can users identify anti-patterns on their own? Do they want assistance from a tool? And are there situations where users actually want to use these patterns, despite researchers having identified them as undesirable?

In this survey component, we investigate users' perceptions of anti-patterns identified by prior work, which we define in Table 2. We selected 12 anti-patterns from prior work and re-named them to reduce any bias the names and descriptions might introduce when evaluating their desirability (e.g., from "Action Conflict" [24] to "Opposite Behaviors"). We also added one that looks for any rules with different triggers and different actions (we call this "Different Triggers, Different Behaviors") to have something benign to compare against the anti-patterns identified by prior work as undesirable.

Participants were randomly assigned four anti-patterns. For each anti-pattern, we first provided participants with a definition and an example of the anti-pattern. Then, we asked participants to complete the following three tasks.

Task 3a: Identifying anti-patterns First, can users understand anti-patterns well enough to identify them in a TAP program, and do they have a good sense of which anti-patterns are more difficult to understand? In this task, we presented participants with four TAP programs, and asked them to select the program that was an instance of their assigned anti-pattern. We also asked them to rate the difficulty of understanding the anti-pattern on a five-point scale (Very Difficult, Difficult, Neither Difficult Nor Easy, Easy, Very Easy).

Task 3b: Perceptions of anti-patterns Do users believe anti-patterns to be problematic? Or, do they think there might be situations where people would want anti-patterns in their TAP programs? In this task, we asked participants four questions about their assigned anti-pattern: if they think their own TAP rules would contain the anti-pattern, if they would want the anti-pattern in their TAP rules, if they think others would want to use the anti-pattern, and if they want to avoid the anti-pattern. Participants responded on a four-point scale (Never, Rarely, Sometimes, Always).

Task 3c: Perceptions of tools for anti-patterns Would tools that detect anti-patterns be useful to users? We asked participants four questions about their assigned anti-pattern: whether they need help identifying it, whether they want help identifying it, whether they would use a tool that finds it, and whether they would be annoyed by a tool looking for the anti-pattern. Participants responded on a four-point scale (Definitely not, Probably not, Probably, Definitely).

3.4 Part 4: TAP program repair

TAP security analysis tools typically notify users of bugs or potential problems, but don't necessarily tell them how to fix them. Some tools describe the trace of events leading to the problem (e.g., Soteria [4]), others report only which rules are involved in the violation [16, 18], and yet others synthesize patches to suggest to the user [14, 28].

We ask, what types of information provided by analysis tools about an error or policy violation are most helpful for users trying to understand and correct the issue? In this survey component, we compare three forms of feedback.

Task 4a: Fixing a buggy rule Here we asked participants to fix a policy violation caused by a buggy rule. First, we showed participants a scenario and goal, and set of TAP rules, and asked them to pretend that a tool found a problem. In one scenario, the participants were told the user wants their door to be locked when they aren't home ("door lock" scenario), in another the user wants their lights to blink only to indicate that there is smoke in their home ("smoke", which is based on

Anti-Pattern Name	Description
Different Triggers, Same Behavior [4]	This pattern looks for rules that are triggered by different events, but lead to the same action.
Same Except No Condition [24]	This pattern looks for rules that are identical except that one has the WHILE condition and the other one doesn't.
Same Triggers, Different Behavior & Conditions [6, 7, 24, 27]	This pattern looks for two rules that are triggered by the same event and one rule has a WHILE condition and the other rule turns off the WHILE condition.
Chains with Opposite Behaviors [7, 24]	This pattern looks for rules that trigger other rules (i.e., form chains) and have different behaviors.
Chain [6, 7, 9, 19, 27]	This pattern looks for rules that may trigger other rules (i.e., form chains).
Different Triggers, Different Behaviors	This pattern looks for rules with different triggers and different behaviors.
Loops [1, 5, 7, 16, 24]	Triggering any rule in the loop will cause another rule to be triggered, which then causes the first rule to trigger again, leading to a loop.
Opposite Behaviors [1, 4, 5, 7, 15, 18, 19, 24, 27]	This pattern looks for rules that may trigger at the same time and cause opposite behaviors.
Same Behaviors [4, 5, 18, 24, 27]	This pattern looks for situations where multiple rules trigger the same behavior.
Un-Paired Rules [1, 12, 19]	Some rules form pairs (one rule might turn a device "on", while another turns it "off"). This pattern looks for rules that are missing their natural pair.
Extended Behavior [1, 12, 27]	This pattern looks for rules that do not account for behaviors that do not happen instantaneously, i.e., they are "extended" over a period of time.
Privacy [3, 5, 11, 18, 19, 21]	This pattern looks for rules that may allow people to learn private things about you.
Trust [5, 11, 18, 19, 21]	This pattern looks for rules that do things that require your trust.

Table 2: TAP anti-patterns included in S2, how we described them to participants, and the prior work inspiring them.

prior work [16]), and in the last scenario, the user never wants their house to be warmer than 72 degrees ("temperature").

Participants were randomly assigned to one of three conditions that determines how much additional information the "tool" gives them: a) which rule is involved, b) which rule is involved and the state of the home when the bug occurs, and c) a full trace that describes the series of events leading to the violation, including the initial state of all of the devices.

We then asked participants three questions to emulate the debugging process: (1) whether the problem is due to a missing rule, a (single) misbehaving rule, or interactions between multiple rules; (2) if the fix involves adding, modifying, or deleting a rule; and (3) to perform the fix, which involved writing a new TAP rule using an If-Then or If-While-Then template and drop-down menus, or choosing a rule to edit or delete (depending on their answer to the previous question). Participants repeated this task twice for two different scenarios, which were randomly assigned from three scenarios.

Task 4b: Fixing an incomplete program We test whether participants can fix a violation caused by a missing rule. Like the previous task, we showed participants a scenario, goal, and a set of TAP rules. Participants were randomly assigned to one of three conditions that determines how much additional information the "tool" gives them: a) a rule is missing; b) a rule is missing, and the state in which the missing rule is needed; and c) a full trace of events leading to the state in which the missing rule is needed. We then asked participants to write a TAP rule to fix the error, using an If-Then or If-While-Then template and drop-down menus. Everyone repeated this task for the same two scenarios.

4 Methodology

In this section, we describe the specific methods and procedures for conducting the study.

Survey structure Because of the length of the survey questions, we split our survey components across two different surveys to reduce the amount of time it took for individual participants to complete the study (see Figure 1 for an overview). Both surveys were implemented in Qualtrics.

Both surveys begin by asking participants to read and accept a consent form; then proceed to Part 1 (Identifying User Needs), which asks participants about their smart-home usage and their TAP goals. Then, to ensure that all participants have a baseline understanding of TAP, we give participants two practice exercises where we present them with a smart-home scenario, set a goal for them to achieve using home automations, and ask them to pick the appropriate TAP rule from a list of rules. They receive feedback explaining why their selection was or was not correct.

At this point, the survey flows diverge: Survey 1 (S1), proceeds to Tasks 2a-c (Writing Declarative Policies), while Survey 2 (S2) proceeds to Tasks 3a-c (Identifying TAP Anti-Patterns) followed by Tasks 4a-b (TAP Program Repair).

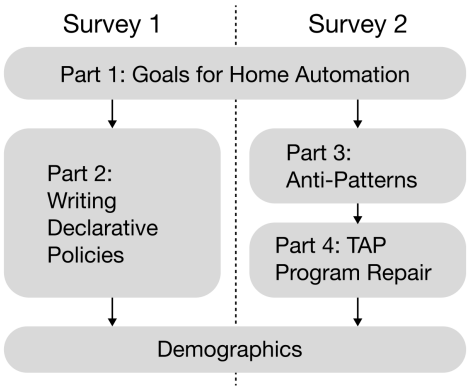

Figure 1: Diagram of the survey structure. Parts 2-4 were split across two surveys to shorten survey length.

At the end of both surveys, we collect participant demographics: age, gender, highest level of education achieved, and whether they have experience in a computing field. Lastly, we asked the participants for permission to publish their anonymized responses (available online [17]). We included two attention-check questions in each survey and discarded responses where both were answered incorrectly.

We revised the surveys over several pilot studies, which involved participants from varied technical backgrounds to help tune the difficulty of the tasks. The study was approved by the IRB at Carnegie Mellon University. Participants who completed a survey and passed at least one attention check received $10.00. The full text of both surveys is in Appendix B.

Recruitment We recruited participants with Prolific. Based on a rule-of-thumb sample-size estimation for an ordinal logistic regression we planned to conduct [23], we estimated we needed 175 participants for S1 and 275 for S2. Participants had to be 18 years or older, located in the US, fluent in English, and have experience with smart-home devices. Participants were only allowed to take one of the two surveys. We used Prolific's gender-balanced sample. S1 was published in October and S2 in November 2022. The median time to complete S1 was around 17 minutes, and 23 minutes for S2.

We received 176 complete responses for S1 and 278 responses for S2. We excluded responses that failed both attention check questions (1 for S1 and 3 for S2) as well as people who skipped more than 2 background questions about their smart-home experiences (1 for each survey). Responses were also evaluated for internal consistency (e.g., did any participants report no smart-home experience after passing the pre-screens?) and nonsensical text responses (1 for S2). In the end, we had 174 usable responses for S1 and 273 for S2.

Participant demographics Table 3 shows participant demographics for both surveys. Participants were balanced across gender, but skewed younger (only 16% and 18% were 46+ years old in S1 and S2, respectively). Most were educated, with 60% holding at least a 2-year degree across both surveys. Most participants did not have experience in a computing field (83%), and had used some form of automation in their own homes (61%).

	Survey 1		Survey 2	
	n	%	*n*	%
Gender				
Female	84	48.3%	131	48.0%
Male	87	50.0%	136	49.8%
Non-binary	3	1.7%	5	1.8%
Prefer to self-describe	0	0.0%	1	0.4%
Age				
18-25	39	22.4%	64	23.4%
26-35	63	36.2%	99	36.3%
36-45	44	25.3%	62	22.7%
46+	28	16.1%	48	17.6%
Highest Education Achieved				
Have not completed high school	0	0.0%	1	0.4%
High school or equivalent	68	39.1%	108	39.6%
Bachelor or associate degree	81	46.6%	130	47.6%
Graduate degree	24	13.8%	32	11.7%
Other	1	0.6%	2	0.7%
Experience in Computing?				
Yes	28	16.1%	47	17.2%
No	146	83.9%	226	82.8%
Experience with TAP?				
Yes	109	62.6%	164	60.1%
No	65	37.4%	109	39.9%
Total	174	100%	273	100%

Table 3: Demographics of study participants for both surveys.

We also collected data on the devices participants used in their homes. Almost all participants owned or used smart TVs (88%) and voice assistants (86%), but less than half owned non-entertainment-related devices like smart lights (50%) or thermostats (14%). The full list is available in Appendix C.

5 Results

This section presents our study results. The organization reflects the research questions and tasks from Sections 1 and 3.

5.1 RQ1: Users' Goals for TAP Rules

First, we present results on the goals that smart home users seek to accomplish with home automation.

Home safety and security are the most important high-level goals We asked participants to rate the importance of each of the following high-level goals for home automations: home safety, home security, comfort and convenience, understanding failures, privacy, and "just for fun", rating each on a four-point Likert scale. Figure 2 summarizes the responses.

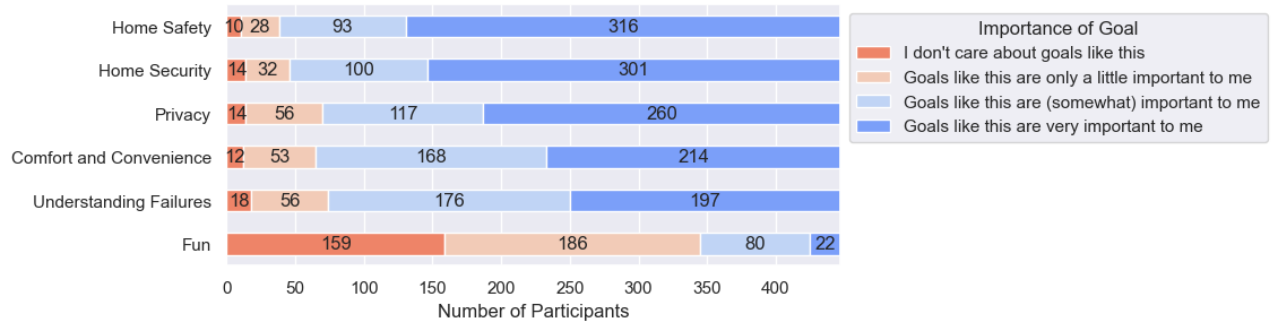

Figure 2: Importance of home automation goals. Safety and security were top goals, while fun was relatively unimportant. Each category included an example, like "Whenever my security camera turns off, I want to know why it happened" for Home Safety.

We found that the goals were divided into three tiers of importance: Home safety and security were the most important, with 92% and 90% of participants rating them as "(somewhat) important" or "very important". Privacy, comfort and convenience, and understanding failures were of secondary importance, with 84%, 86%, and 83% of participants rating them as (very) important. Notably, fun was not a strong motivation for using home automation, with 77% of participants reporting they don't care or that it is only a little important.

There were also 17 participants who described additional goals in a free response field. Some responses rephrased one of the above goals, but other goals included saving money/energy (5), accessibility (1), and health and safety (1).

Users' interest in TAP is primarily for comfort and convenience, home security We also asked participants in S1 to describe, in their own words, the purpose of the TAP programs in their home, or if they didn't have experience with TAP, what they hypothetically would do with TAP programs. We labeled their responses by the categories of goals identified above, and the level of specificity (at the whole home level vs. individual devices).

Contrary to the high-level self-reported goals, we found that the most common use-case for participants' TAP programs was comfort and convenience (mentioned in 69% of the responses) followed by home security (54%). Others mentioned saving money or energy (21%). Home safety, fun, privacy, and health appeared in fewer than 6% of the responses.

The difference between the importance of high-level goals and the actual programs that people write suggests that users want privacy and safety as an implicit property of their smart home, and will automate comfort/convenience or security related functionality using TAP.

Users goals for automation range in sophistication Responses ranged in specificity, indicating that users have different mental models for goals. Some responses (24%) were extremely broad, such as:

> *I want my home to be comfortable and secure. (P130)*

Many (59%) were specific to particular devices:

> *I would want my doors to be locked whenever I am not home. [...] certain devices to be turned on at certain times of the day with smart plugs, like my coffee machine for example. (P1)*

A few participants (16%) described both high-level goals and specific behaviors:

> *My goals are to keep my home safe and secure when I leave/while I am sleeping, so I want my doors locked and secure during these situations. I also love being able to control my thermostat from my phone. I also love using smart lights to help make my apartment look better and feel more like home. (P113)*

5.2 RQ2: Usability of Declarative Policies

Next, we explore the usability of template-based interfaces for specifying declarative policies about the desired state of the home, such as those proposed by AutoTap [28] and SafeTAP [16]. We evaluate whether users can a) pick a correct template format to achieve a goal and b) correctly fill out the fields of a template. We also investigate c) which templates seem most natural to users.

Task 2a: Participants can usually select a suitable template In this task, participants were presented two scenarios, and asked to pick a template that was appropriate for the scenario. In the *smoke detector* scenario, the home's lights should blink only when the smoke detector is triggered. In the second, *neighbors*, some automations cause the home's lights to blink, and the goal is that they should blink for at most 30 minutes to avoid annoying the neighbors. Participants chose from all nine SafeTAP and AutoTAP templates (Table 1).

In the *smoke detector* scenario, 91% of participants selected a valid template while 71% did likewise in the *neighbors* scenario. The correct templates for the first scenario included *Only When* (selected by 51%), *Event-State Conditional* (24%), *Whenever* (12%), or *Multi-State Unconditional*

(4%). Meanwhile, only *One-State Duration* (62%) or *Event-Event Conditional* (9%) could be used to write a policy for the *neighbors* scenario. The one-sample Pearson Chi-Squared tests indicated that the differences in the number of participants who picked each template were significantly different from chance (*smoke detector*: $\chi^2(8, N=174) = 290.37$, $p < .0001$, *neighbors*: $\chi^2(8, N=174) = 420.67$, $p < .0001$).

In general, it appears that participants can match a template to a high-level goal, but their success likely depends on the goal or situation: the more-complex *neighbors* scenario involved duration and had much lower success rates than the simpler *smoke detector* scenario.

Task 2b: Participants' success at filling out templates varies based on complexity of template and goal In this task, participants were assigned a scenario and a template, and were asked to *correctly fill-in-the-blanks* of the template to describe the goal. In the *window* scenario, the goal is to ensure the windows are closed when it rains; in *temperature*, that the room is cooler than 73F at night; and in *dryer*, that the dryer cannot run until after 5PM. In Table 4 we show participants' success at filling out their assigned templates.

We found that template filling success rates varied widely across scenarios and templates. For *window*, 74% of participants correctly filled out templates, compared to 28% for *temperature*, and 51% for *dryer*. Within each scenario, participants found more success with some templates than others. For example, 98% of participants assigned to the *Always/Never* template for *window* filled it out correctly, while only 37% were able to correctly complete the *Multi-State Unconditional* template for the same scenario. An analysis of variance based on mixed binomial logistic regression indicates a significant effect of scenario on correctness ($\chi^2(2, N = 522) = 47.9, p < 0.001$) and template on correctness ($\chi^2(6, N = 522) = 79.1, p < 0.001$).

We also observed that specific templates may be misinterpreted in certain situations. For *temperature*, no participants filled out the *Whenever* template correctly, which indicates that this template may be too confusing to use for duration conditions, even though it is technically adequate for the goal. There is also variation between the "always" and "never" forms of the same template. For instance, participants were more successful at filling out the *One-State Unconditional* template when they picked the "always" form (76% correct) than the "never" form (27%). In another case, some participants chose the "always" form of the *Multi-State Unconditional* template, even though it is not possible to correctly write the template in that form, meaning that 100% failed. This is consistent with prior work that observed that users tend to misinterpret the meaning of this template [28].

Task 2c: Template preferences vary Lastly, we investigated which templates participants preferred. We created five scenarios; in each we presented four sets of filled-in templates that satisfied a goal, and asked the participant to pick the one that sounded most natural to them. Within each scenario, we varied several factors, including the use of SafeTAP vs. AutoTap templates; always vs. never forms of templates; and whether multiple conditions were fulfilled via multiple templates or via one template using an AND or OR clause. The templates, their attributes, and the percentage of participants that chose each option are shown in Appendix E.

In each scenario, participants had somewhat clear preferences: the top two choices in each comprised over 75% of the votes. One-sample Pearson Chi-Squared tests confirmed that participants' choices were significantly different from chance.

The specific forms and attributes associated with the more popular templates varied from scenario to scenario. In 3 of 5 scenarios, SafeTAP templates were preferred over AutoTap. The *Whenever* template was relatively popular in each of the scenarios it appeared in (most popular in *security camera* and *forecast*, 2nd most popular in *smoke*), and the *Multi-State Unconditional* template was the least popular in both scenarios it appeared in. Participants did not prefer having fewer templates to more templates, nor was it clear if they preferred the "always" or "never" forms of templates.

5.3 RQ3: Understanding Anti-Patterns

Here, we report whether participants could identify TAP anti-patterns in S2 and whether they perceived them as undesirable. We see a large variation in participants' ability to identify anti-patterns and, surprisingly, that a substantial number of people may actually want to use anti-patterns in their programs.

Task 3a/b: Some anti-patterns are easy to spot, others are hard In this task, we showed participants the definition of an anti-pattern, and asked them to choose which of four example TAP programs contained that anti-pattern. Table 5 summarizes the results.

Overall, participants' success at identifying anti-patterns varies substantially between anti-patterns. Anti-patterns that most participants correctly identified typically involve redundant rules that share a trigger or action: *Same Behaviors* (75% correct), *Different Triggers, Same Behavior* (83%), and *Same Except No Condition* (83%). The anti-patterns participants had the most trouble identifying required understanding how long an action takes to complete (*Extended Behavior*, 34%), whether one rule can trigger another (Chains, 37%; and *Chains with Opposite Behaviors*, 31%), and other nuanced concepts, like the integrity of a trigger/action (*Trust*, 26%).

We also asked participants how difficult they found it to understand the anti-pattern they were tasked to look for. Figure 3 shows participants' responses. We again found a wide spread depending on the template. The easiest anti-pattern to understand was *Different Triggers, Same Behavior*, where 69% reported it was "easy" or "very easy" to understand, and the hardest was *Chains with Opposite Behaviors*, which 48% of people found "very difficult" or "difficult."

Participants' self-reported understanding of anti-patterns roughly correlates with their performance identifying the

Scenario	Template Name	Template	Overall % Correct	"Always" % Correct	"Never" % Correct
Window	Always/Never	I [always/never] want __	98	97	100
	One-State Unconditional	__ should [always/never] be active	91	97	71
	State-State Conditional	__ should [always/never] be active while __	67	64	86
	Multi-State Unconditional	__ and __ should [always/never] occur together	37	0	89
Temperature	One-State Unconditional	__ should [always/never] be active	62	76	27
	State-State Conditional	__ should [always/never] be active while __	39	41	29
	Multi-State Unconditional	__ and __ should [always/never] occur together	7	0	21
	Whenever	Whenever __ make sure that __	0	N/A	N/A
Dryer	State-State Conditional	__ should [always/never] be active while __	75	76	74
	Event-State Conditional	__ should [only/never] happen when __	63	0	96
	Whenever	Whenever __ make sure that __	57	N/A	N/A
	Event-Event Conditional	__ should [always/never] happen within __ after __	12	0	63

Table 4: Percent of participant responses that correctly filled out the blanks in a declarative policy template, across three scenarios. We also report the proportion of correct responses for the always/never form of each template, where applicable (the "only" form of event-state conditional is included under "always"). Success rates varied across scenarios and templates, indicating that people's ability to fill out policy templates is extremely context-specific.

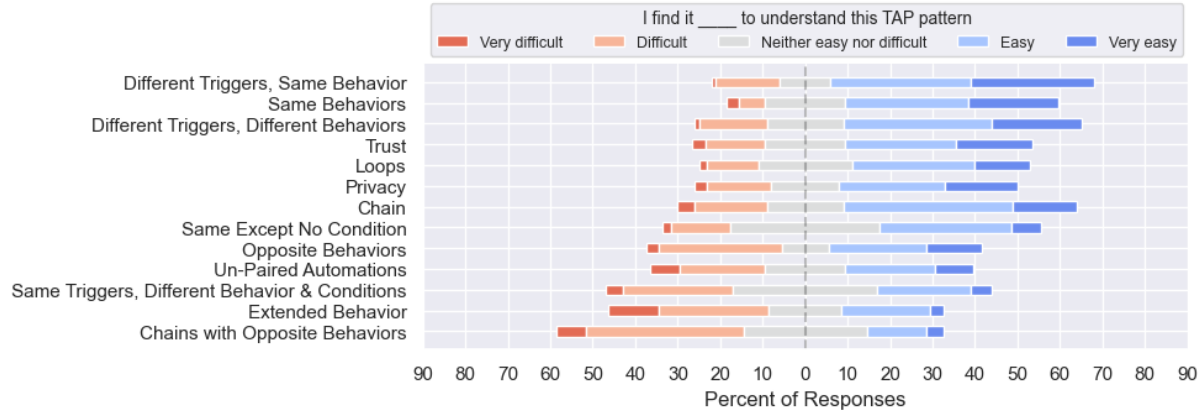

Figure 3: Perceived difficulty of identifying problematic TAP anti-patterns, sorted by average response score for each anti-pattern when converted to numerical values.

anti-pattern: an analysis of variance based on a mixed logistic regression indicated a significant effect of a participants' self-reported understanding of the anti-pattern and whether they correctly identified the template ($\chi^2(4,N=1683)=19.1$, $p<0.001$). However, for some specific anti-patterns, understanding and identification rates don't appear to align: for example, 58% of people thought that *Chain* was "easy" or "very easy" to understand, but only 37% of participants correctly identified the set of TAP rules exhibited that pattern.

Task 3b/c: Participants would accept a tool to find anti-patterns, but also *want* to use TAP anti-patterns We find that across anti-patterns, a majority of participants would "sometimes" or "always" like help identifying them (ranging from 57% to 78%) and would use a tool that helped them (66% to 89%). These responses (summarized in Appendix F) also roughly align with the perceived difficulty.

Surprisingly, participants did not find the anti-patterns universally undesirable. Figure 4 shows participants' perceptions of the desirability each anti-pattern. For all anti-patterns, many said they would "rarely" or "sometimes" *like to have* the anti-pattern in their home (at least 49% for each anti-pattern) and would want to *avoid* the anti-pattern "rarely" or "sometimes" (at least 51%). A majority also reported that their rules may "sometimes" or "rarely" have the anti-patterns (at least 67%).

However, a few anti-patterns were more undesirable than others: 41% of people always wanted to avoid *Opposite Behaviors*, 40% for *Chains with Opposite Behaviors* , and 31% for *Loops*. This suggests the TAP anti-patterns that researchers have identified as problematic might somehow be useful for people, or at least that people may not recognize them as problematic based on the description and example alone.

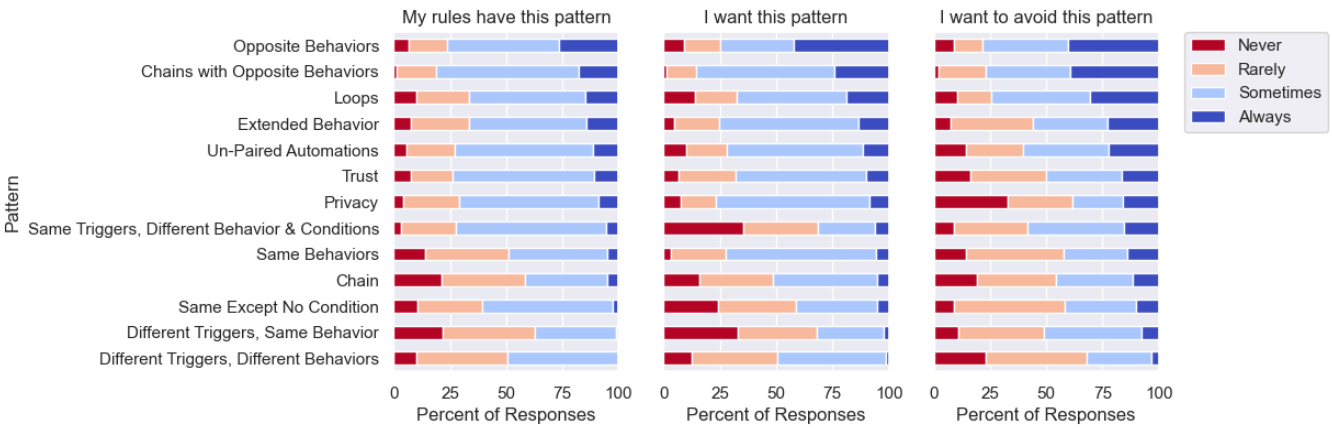

Figure 4: Participants' perceptions of the desirability of TAP patterns.

TAP Pattern Name	% Correctly Identified Round 1	Round 2
Loops	81%	47%
Same Behaviors	71%	80%
Privacy	67%	71%
Extended Behavior	52%	15%
Opposite Behaviors	51%	48%
Trust	48%	4%
Un-Paired Automations	30%	46%
Different Triggers, Same Behavior	83%	—
Same Except No Condition	83%	—
Different Triggers, Different Behaviors	53%	—
Same Triggers, Different Behavior & Conditions	57%	—
Chain	37%	—
Chains with Opposite Behaviors	31%	—

Table 5: Percent of participants that were able to identify the program that exhibited a TAP anti-pattern. For half of the anti-patterns, we tested participants a second time on an additional set of programs, shown in the rightmost column.

5.4 RQ4: Fixing TAP Programs

Many security tools, regardless of the types of properties they check, help users identify problems but do not fix them. Rather, tools typically generate some feedback, like a counterexample, to help the user in their efforts to make a fix. In this set of tasks, participants are asked to pretend a tool has identified some problem and they need to repair the program. In Task 4a, participants must identify and make modifications to fix a misbehaving rule, while in Task 4b, participants must add a missing rule. We vary the type of information the "tool" provides, showing either 1) which rule is misbehaving/that a rule is missing, 2) the rule and the state causing the issue, or 3) a full execution trace leading to the error, to investigate which form of feedback is mostly likely to help users.

Task 4a: Multiple types of feedback can help users repair broken rules We created three scenarios (*lock*, *temperature*, and *smoke*) where there is a violation of a declarative policy (worded like the goals in Section 5.2). In all of them, the error is caused by one incorrect rule and can be fixed by modifying or deleting it. We measure three stages of fixing the bug: identifying the problem, identifying how to fix it, and implementing the fix by writing a new rule or choosing the rule to modify/delete. Table 6 shows success rate of each fix stage for the feedback conditions across all three scenarios.

We found that the type of feedback had no effect on participants' ability to identify and fix bugs. For identifying the cause of the error, the difference in success rates ranged from 4-9% across conditions. The difference between conditions were also small for identifying how to fix (1-5%), and slightly larger for implementing a fix (9-15%). Analyses of variance based on mixed logistic regressions found no significant effect of condition on any of the three metrics ($\chi^2(2, N=546)=2.47$, *n.s.*, $\chi^2(2, N=546)=0.22$, *n.s.*, $\chi^2(2, N=546)=5.54$, *n.s.*).

However, the scenario did affect the success rate; namely, the *temperature* scenario was harder than the others. Across conditions, only 53% of participants successfully implemented a fix for *temperature*, versus 80% and 84% in for *lock* and *smoke*. The previous regressions indicated that scenario had a significant effect on success rates for all three metrics, ($\chi^2(2, N=546)=30.8$, $p<0.001$, $\chi^2(2, N=546)=41.2$, $p<0.001$, $\chi^2(2, N=546)=52.4$, $p<0.001$) and pairwise comparisons using Z-tests between scenarios in all three metrics showed *temperature* was significantly different from the others.

These results suggest that when a program is broken due to a single buggy rule, it is equally effective for tools to either indicate the rule causing the error (with or without additional information about the state of the home), or provide an execution trace of the error. They also suggest that some errors may be harder to fix than others, regardless of feedback provided.

Task 4b: State information helped participants write missing rules Unlike fixing bugs, we find that the type of

	Identified Problem			Identified Fix			Implemented Fix		
Condition	Lock	Smoke	Temp	Lock	Smoke	Temp	Lock	Smoke	Temp
Rule	84%	88%	71%	91%	95%	74%	84%	89%	63%
Rule w/State	86%	89%	68%	92%	93%	75%	75%	86%	48%
Full Trace	82%	88%	77%	92%	90%	77%	82%	77%	48%

Table 6: Percentage of participants that identified and implemented fixes successfully, across each scenario and tool condition. Different types of feedback did not have an effect on fix rates, but the temperature scenario was more difficult.

feedback does affect participants' success rate for adding missing rules. For both scenarios, participants were more likely to write the correct TAP rule when told that a rule is missing, and the state in which the program fails, (73% for *temperature*, 50% for *window*), rather only being told that a rule is missing (62% and 19%, respectively), or being given a full trace (57% and 20%).

An analysis of variance based on a mixed logistic regression found a significant effect of condition on success rate (χ^2(2,N=546)=23.5, p<0.001) and scenario on success rate (χ^2(2,N=546)=54.7, p<0.001). Post-hoc Z-tests indicate that the Missing Rule with State condition was significantly different from the other two conditions.

These results suggest that for the case where an error is caused by a missing rule, identifying the state which causes the issue makes it more obvious what rule to add, perhaps because it provides the user with a starting point.

5.5 Effect of TAP Experience

Surprisingly, we found that previous experience with TAP or computing, the number of smart home devices owned, and demographic factors had little correlation with participants' performance on the above tasks.

For Tasks 2b, 3a, 3b, 3c, 4a, and 4b, we conducted mixed logistic regressions to test if prior experience or demographics were correlated with the correctness of participants' answers or perceptions of the difficulty of a task. In each regression, we modeled the experimental conditions, participants' experience with TAP, experience with computing, number of devices owned, and demographic factors as predictors.

We only found a statistically significant effect for experience or demographics in three cases: In Task 4a, the success rate for implementing fixes to a template was 10 percentage points higher for participants with TAP experience (χ^2(1,N=546)=7.52,p=0.006), and the success rate for identifying fixes was 11 percentage points higher for male participants (χ^2(1,N=546)=5.61,p=0.017). In Task 2b, counterintuitively, the success rate for picking templates decreased by 4 percentage points for each smart home device a participant owned (χ^2(1,N=348)=4.61,p=0.032). The lack of consistent findings across tasks suggests that users that lack expertise in smart homes will not necessarily find security analysis tools harder to use, and that improving the usability of such tools is important even for expert users.

6 Discussion

Our study shows that trigger-action programming and security analysis tools for TAP have promise to be broadly accessible: participants worked with abstractions proposed in prior work successfully, e.g., to specify declarative policies for desired smart-home states and to identify anti-patterns in TAP programs. We also found that a high level of expertise in TAP or technology generally was not required to perform well at these tasks. At the same time, the results also suggest the need for more research: the features of some proposed tools were *much* less approachable than others (e.g., confusing declarative policy templates), and some of users' preferences clash with researchers' expectations (e.g., some users wanting to use anti-patterns). In this section, we make concrete recommendations based on our results, both for building tools and for future research.

6.1 Recommendations for Tool Developers

Templates are a promising approach for writing policies Encouragingly, participants were generally able to pick appropriate policy templates and fill them out, at least for some template formats in some scenarios (Section 5.2). This suggests that these template-based approaches to writing policies will work well for tools. The greatest confusion seems related to particularly confusing template formats or tricky scenarios, rather than to any fundamental misunderstanding of the interface, suggesting that with some refinement, these fill-in-the-blanks templates might be approachable to most users.

Policy tools should use context to guide users through template selection and filling Some of the more spectacular failures in the template-based tasks (Section 5.2) occurred when the template was not the most suitable choice for the context of the scenario, e.g., when non-duration based templates were assigned for duration-based scenarios. Though, more general-purpose templates technically can be used to satisfy duration-based goals, participants struggled with filling those templates. Thus, tools of this category could interac-

tively guide people into selecting more appropriate templates based on some contextual clues.

Analysis tools should provide specific information for debugging In many cases, participants were able to fix bugs in TAP programs when given sufficient information (Section 5.4). When a rule was broken, simply highlighting that specific rule was sufficient, but if a rule was missing, it was most helpful to also provide some context about when the rule would be needed. For security analysis tools, if automatically synthesizing a fix to a problem is not possible, providing information about the specific states and rules relating to the problem appears to be the best approach.

Tools should indicate how rules are modeled Some of the more confusing anti-patterns from Section 5.3, like *Extended Behavior*, require users to understand how TAP rules will behave in the real world. These results are consistent with prior work that identifies gaps in users' mental models of TAP programs and rules, especially when distinguishing instant (e.g., send email), extended (e.g., brew coffee), and sustained (e.g., turn lights on) behaviors [12]. In these cases, tools may need to provide more information about how the rule is modeled or interactivity to help people understand and fix the problem.

6.2 Recommendations for Researchers

Policy templates need refinement for comprehension In Section 5.2, we found several examples where specific templates had confusing wordings that negatively affected participants' ability to use them correctly. Among the most challenging to use templates are the "always" form of the "_ and _ should [always/never] occur together" template and the "Whenever _ make sure that _" template. The first was often used when it wasn't appropriate (which is consistent with prior findings [28]), and the confusion seemed to come from a mismatch between how the template is phrased (people appear to read it as "if-then") and the underlying formalism ("if-and-only-if"). Adjusting the wording of this template would likely improve its performance. On the other hand, the *Whenever* template is so popular (Table 13) and so frequently misused (Table 4), that it may be worth avoiding altogether.

Researchers developing tools to enforce declarative policy should carefully test the usability of the templates that they provide as an interface to users. It would also be helpful to dedicate some research to identify common characteristics of the most (and least) usable policy templates and develop guidelines for writing new policy templates. We can additionally leverage some of the insights from Section 5.1 to identify scenarios that research participants are more likely to find relevant and useful.

Users have an uneven grasp of anti-patterns, and tools could help TAP anti-patterns were tricky for people to understand and identify. Some were relatively simple and easy for people to understand, like *Different Triggers, Same Behavior*, which simply compare triggers an actions. In these cases, warnings with simple definitions and examples of the anti-pattern, like in our survey, could be sufficient.

Surprisingly, we also found that even anti-patterns which seem objectively undesirable (like *Loops* or *Conflicting Actions*, described in Section 1) were considered desirable (at least "sometimes") by many of our participants. This highlights the need for more research about how to communicate the threats posed by these anti-patterns, especially when users may overestimate their understanding of these anti-patterns.

6.3 Limitations

Our survey primarily used quantitative measures of usability, like task performance or rating scales for ease of use. We did not capture qualitative feedback on the usability of policy templates, anti-patterns, or template repair tools, which we leave to future work. We used vignettes to enable a controlled evaluation of templates and anti-patterns. However, the scenarios varied in difficulty and may not have been familiar to all participants, which could have contributed to participants' poor performance on some tasks. Because these tasks were presented as vignettes in a survey interface, the findings may not generalize to a live smart-home setting, due to differences in the user interfaces and the scenarios in which users would encounter anti-patterns or create templates in practice. Not all participants in our study had prior experience with TAP (39% did not), and for those that did, we did not characterize their level of expertise with TAP. Though we did not find an effect of prior experience on most tasks, in real deployments familiarity with smart-home configuration and TAP may impact the usability of security analysis tools.

7 Conclusion

In this paper, we presented the results of our exploratory survey of TAP security analysis approaches. We found considerable variation in the success and perceived utility of various approaches. Participants were generally capable of picking the correct templates for implementing specified high-level policies; however, while they generally filled out some templates very accurately, there were other templates where almost all participants struggled. We found that participants were more successful at debugging TAP programs when they knew some of the relevant state conditions involved in the violation, compared to only telling them which rule was involved or sharing all of the events leading up to the violation. Participants had more difficulty identifying some anti-patterns than others, didn't always seem to realize when they were having difficulty understanding them, and didn't find any of them wholly undesirable. More research is needed to determine how to best describe anti-patterns to facilitate understanding and communicate the threats they pose.

Acknowledgments We would like to express our gratitude to the numerous students, staff, faculty, and friends who helped pilot our studies and offered helpful feedback to improve their design. This work was supported in part by Carnegie Mellon CyLab, Cisco Research through the Carnegie Mellon CyLab partnership program, NSF award CNS2114148, and a CyLab Presidential Fellowship.

References

[1] Will Brackenbury, Abhimanyu Deora, Jillian Ritchey, Jason Vallee, Weijia He, Guan Wang, Michael L. Littman, and Blase Ur. How users interpret bugs in trigger-action programming. In *Proc. of CHI*, 2019.

[2] Lei Bu, Wen Xiong, Chieh-Jan Mike Liang, Shi Han, Dongmei Zhang, Shan Lin, and Xuandong Li. Systematically ensuring the confidence of real-time home automation IoT systems. *ACM Trans. on Cyber-Physical Systems*, 2, 2018.

[3] Z. Berkay Celik, Leonardo Babun, Amit K. Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A. Selcuk Uluagac. Sensitive information tracking in commodity IoT. In *Proc. of USENIX Security*, 2018.

[4] Z. Berkay Celik, Patrick McDaniel, and Gang Tan. SOTERIA: Automated IoT safety and security analysis. In *Proc. of USENIX ATC*, 2018.

[5] Z Berkay Celik, Gang Tan, and Patrick D McDaniel. IoTGuard: Dynamic enforcement of security and safety policy in commodity IoT. In *Proc. of NDSS*, 2019.

[6] Haotian Chi, Chenglong Fu, Qiang Zeng, and Xiaojiang Du. Delay wreaks havoc on your smart home: Delay-based automation interference attacks. In *Proc. of IEEE SP*, 2022.

[7] Haotian Chi, Qiang Zeng, Xiaojiang Du, and Jiaping Yu. Cross-app interference threats in smart homes: Categorization, detection and handling. In *Proc. of IEEE/IFIP DSN*, 2020.

[8] Camille Cobb, Milijana Surbatovich, Anna Kawakami, Mahmood Sharif, Lujo Bauer, Anupam Das, and Limin Jia. How risky are real users' IFTTT applets? In *Proc. of USENIX SOUPS*, 2020.

[9] Wenbo Ding and Hongxin Hu. On the safety of IoT device physical interaction control. In *Proc. of ACM CCS*, 2018.

[10] Home Assistant. Home Assistant: Awaken your home. https://www.home-assistant.io, 2023.

[11] Kai-Hsiang Hsu, Yu-Hsi Chiang, and Hsu-Chun Hsiao. SafeChain: Securing trigger-action programming from attack chains. *IEEE Transactions on Information Forensics and Security*, 14(10), 2019.

[12] Justin Huang and Maya Cakmak. Supporting mental model accuracy in trigger-action programming. In *Proc. of ACM UbiComp*, 2015.

[13] IFTTT. IFTTT: Every thing works better together. https://ifttt.com, 2023.

[14] Chieh-Jan Mike Liang, Lei Bu, Zhao Li, Junbei Zhang, Shi Han, Börje F. Karlsson, Dongmei Zhang, and Feng Zhao. Systematically debugging IoT control system correctness for building automation. In *Proc. of ACM BuildSys*, 2016.

[15] Chieh-Jan Mike Liang, Börje F. Karlsson, Nicholas D. Lane, Feng Zhao, Junbei Zhang, Zheyi Pan, Zhao Li, and Yong Yu. SIFT: Building an internet of safe things. In *Proc. of IPSN*, 2015.

[16] McKenna McCall, Faysal Hossain Shezan, Abhishek Bichhawat, Camille Cobb, Limin Jia, Yuan Tian, Cooper Grace, and Mitchell Yang. SafeTAP: An efficient incremental analyzer for trigger-action programs, 2021. CMU Technical Report.

[17] McKenna McCall, Eric Zeng, Faysal Hossain Shezan, Mitchell Yang, Lujo Bauer, Abhishek Bichhawat, Camille Cobb, Limin Jia, and Yuan Tian. Supplementary material. https://kilthub.cmu.edu/articles/dataset/Towards_Usable_Security_Analysis_Tools_for_Trigger-Action_Programming_--_Dataset/23100482, 2023.

[18] Dang Tu Nguyen, Chengyu Song, Zhiyun Qian, Srikanth V. Krishnamurthy, Edward J. M. Colbert, and Patrick McDaniel. IotSan: Fortifying the Safety of IoT Systems. In *Proc. of CoNEXT*, 2018.

[19] Mitali Palekar, Earlence Fernandes, and Franziska Roesner. Analysis of the susceptibility of smart home programming interfaces to end user error. In *Proc. of IEEE SPW*, 2019.

[20] SmartThings Inc. SmartThings: One simple home system. A world of possibilities. https://www.smartthings.com, 2023.

[21] Milijana Surbatovich, Jassim Aljuraidan, Lujo Bauer, Anupam Das, and Limin Jia. Some Recipes Can Do More Than Spoil Your Appetite: Analyzing the Security and Privacy Risks of IFTTT Recipes. In *Proc. of WWW*, 2017.

[22] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L. Littman. Practical trigger-action programming in the smart home. In *Proc. of SIGCHI*, 2014.

[23] Maarten van Smeden, Karel GM Moons, Joris AH de Groot, Gary S Collins, Douglas G Altman, Marinus JC Eijkemans, and Johannes B Reitsma. Sample size for binary logistic prediction models: Beyond events per variable criteria. *Statistical Methods in Medical Research*, 28(8), 2019.

[24] Qi Wang, Pubali Datta, Wei Yang, Si Liu, Adam Bates, and Carl A. Gunter. Charting the attack surface of trigger-action IoT platforms. In *Proc. of ACM CCS*, 2019.

[25] Qi Wang, Wajih Ul Hassan, Adam Bates, and Carl Gunter. Fear and logging in the internet of things. In *Proc. of NDSS*, 2018.

[26] Svetlana Yarosh and Pamela Zave. Locked or not? mental models of IoT feature interaction. In *Proc. of CHI*, 2017.

[27] Yinbo Yu and Jiajia Liu. TAPInspector: Safety and liveness verification of concurrent trigger-action IoT systems. *IEEE Trans. on Information Forensics and Security*, 17, 2022.

[28] Lefan Zhang, Weijia He, Jesse Martinez, Noah Brackenbury, Shan Lu, and Blase Ur. AutoTap: Synthesizing and repairing trigger-action programs using LTL properties. In *Proc. of IEEE/ACM ICSE*, 2019.

A Terminology

The terminology differs slightly between our paper and the surveys. We summarize the relevant terminology in Table 7.

B Survey Instrument

This section includes the survey instrument for both S1 and S2. Both surveys include the following sections:

- Consent form, pre-screen questions, and Prolific ID collection
- Background questions on smart home devices; warm-up exercises introducing TAP rules; home automation goals
- Main survey tasks
- Demographic and other wrap-up questions

The surveys are the same except where indicated. We include question text (shortened for brevity) and describe the survey flow decisions and other details about the survey using *italics*, fields from tables using **bold text**, and list answer choices next to circles: ○

Background and warm-up (RQ1) *Introduction to the background questions.*

1. For each of the following categories, tell us if you have used and/or have heard of the device. If you own or have set up a device not listed here, you can use the box at the bottom to describe the device. *Device categories are shown in Figure 12*
   ○ I have used smart features on one of these ○ I have used one of these, but not the smart features ○ I have heard of these, but never used one ○ I have never heard of these

2. *For the devices the participant reports they have used/heard of in Question 1:* For each of the following categories, tell us if you own and/or have set up the device. If you own or have set up a device not listed here, you can use the box at the bottom to describe the device.
   ○ I own and have set up one of ○ I own one of these, but have not set it up ○ I do not own one of these but I have set one up ○ I do not own, nor have I set up one of these

Practice exercises to introduce home automations

3. Have you ever tried to use home automations with your own smart home devices? (Please select "yes" even if you tried to set up home automations without success.) *Allows multiple answers*
   ○ Yes, using a platform like IFTTT, SmartThings, HomeKit, or openHAB ○ Yes, using built-in settings like schedules or based on my location ○ No

4. Which of the following reasons stop you from using more smart home devices than you currently do? (either installing more home automations with your current devices, buying more devices, or using more features on your current devices).
   ○ This is not a factor ○ This is somewhat a factor ○ This is definitely a factor

   (a) Learning to use a new device/feature/home automation is too difficult
   (b) Cost is too high
   (c) Not interesting to me
   (d) Privacy and/or security concerns
   (e) Other (please describe below) *Free response*

5. *S1 only. Wording depends on whether the participant indicated they have used home automations before in Question 3.* Which of the following describes your goals for your home automation? If you have different goals for different types of home automation, you can select multiple options. *Allows multiple answers*

Term	Survey term	Definition
TAP rule	Home automation	Trigger-action programming (TAP) rules allow users to customize their smart home devices. A simple TAP rule format is "IF *trigger* THEN *action*" which causes *action* to happen in response to the *trigger*. In the surveys, we often use the word *behavior* instead of *action*.
TAP program	Multiple home automations	A set of TAP rules which may (or may not) interact with each other when one rule triggers another rule.
(Declarative) policy template	Template	Tools which allow users to check that their smart homes perform specific behaviors use policy templates to help the user communicate their goals to the tool.
TAP patterns	Patterns	Tools which look for trigger-action programming patterns are checking for potentially dangerous/undesirable interactions between TAP rules rather than asking users to specify their own custom property.

Table 7: Terms used in this paper and their survey counterparts.

○ I have goals which some of my home automations work together to achieve ○ I have independent goals which my home automations would achieve independently ○ I have no specific goals in mind for some of my home automations

6. *S1 only.* Can you describe the goals you had in mind (if any) when you were answering the last question? *Free response*

7. Please select from the choices below to tell us how important these goal categories are to you.
   ○ Goals like this are very important to me ○ Goals like this are (somewhat)[1] important to me ○ Goals like this are only a little important to me ○ I don't care about goals like this
   Categories are listed in Figure 2

8. Did you think of any other goal(s) which did not fit into the categories we identified, but would be important to you?
   ○ Yes ○ No

9. *If the participant answered "Yes" in Question 8.* How would you describe the goal(s)? *Free response*

S1 tasks (RQ2)

*We introduce the concept of **declarative policies** and the **templates** used to write them. Policy templates and the work they come from are shown in Table 1. We write templates with **bold text** and underline fields in the templates. Fields which have not been filled are written __. For each task we specify whether the participant chooses a template (which may be filled or unfilled), or fill out the template using a drill-down interface.*

Participants are randomly assigned a survey flow to account for the learning effect of S1 Task 1: half are shown Task 1 first and the rest are shown Task 1 last.

[1] The choice was "somewhat important" in S1 and "important" in S2. The choices were otherwise the same between the surveys.

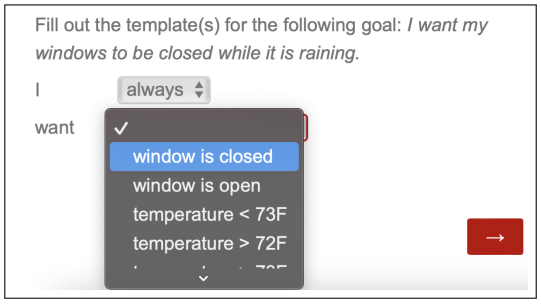

Figure 5: Screenshot of the interface for Task 2a, where the participants are asked to fill out the "**I** always/never **want** __" policy template for the "windows" scenario.

S1 Task 1: Template preference In this task, we want to know which goal formats are the most natural to use.

In the following questions, please select the option which best describes the goal, in your opinion. If none of the choices seem natural, please select the last option and briefly explain why. *Participants are shown 3 of the 5 questions in this section. Full question text may be found in our supplementary material [17].*

S1 Task 2: Template picking *Participants are shown both of the following questions and pick from one of the unfilled templates shown in Figure 1*

15. Pick one template for the following goal: I want my lights to blink to tell me that smoke has been detected. If the lights are blinking, I will assume there is a fire, so I don't want them to blink for any other reason.

16. Pick one template for the following goal: I have some automations to cause my lights to blink, but I am worried they might blink all night long and disturb my neighbors while I am out of town. I want to know that when they blink, they never blink for more than half an hour.

S1 Task 3: Template filling *Participants are shown each of the following questions and one of the templates (randomly*

selected), which they fill via a drill-down interface (shown in Figure 5). The participant also picks between the underlined choices shown in brackets.

17. Fill out the template(s) for the following goal: I want my windows to be closed while it is raining.
    (a) **I** [always/never] **want** __
    (b) __ **and** __ **should** [always/never] **occur together**
    (c) __ **should** [always/never] **be active**
    (d) __ **should** [always/never] **be active while** __
18. Fill out the template(s) for the following goal: I am worried my baby will overheat at night. I read that they should sleep in rooms cooler than 73F.
    (a) **Whenever** __ **make sure that** __
    (b) __ **and** __ **should** [always/never] **occur together**
    (c) __ **should** [always/never] **be active**
    (d) __ **should** [always/never] **be active while** __
19. My dryer is disrupting my Zoom meetings. I don't want it to run during business hours. My meetings are always finished by 5PM.
    (a) **Whenever** __ **make sure that** __
    (b) __ **should** [always/never] **be active while** __
    (c) __ **should** [only/never] **happen when** __
    (d) __ **should** [always/never] **happen within** __ [minutes/hours] **after** __

S2 tasks (RQ3 & RQ4)

Introduction to general properties, which we refer to as "patterns" in the survey.

TAP Pattern task 1 *Each participant answers the following questions for 2 properties, randomly selected from Table 8. The section begins with an explanation of **[Pattern]**, including an example. Half of the participants are asked to identify the example of **[Pattern]** first, the other half are asked the Likert questions first.*

10. Please select the example of automations with the "**[Pattern]**" pattern. (Exactly 1 will fit the pattern.)
    Participants choose from 4 simple programs, the correct response is shown in Table 8.
11. How difficult do you think it is to understand what the "**[Pattern]**" pattern is, as a concept?
    ○ Very easy ○ Easy ○ Neither easy nor difficult ○ Difficult ○ Very difficult
    (a) I find understanding **[Pattern]** to be...
    (b) Technical people would find understanding **[Pattern]** to be...
    (c) Most people would find understanding **[Pattern]** to be...
    (d) Finding **[Pattern]** in my home automations would be...
12. How often would people have the "**[Pattern]**" pattern in their home automations? How often would people want to have the "**[Pattern]**" pattern in their home automations?
    ○ Always ○ Sometimes ○ Rarely ○ Never
    (a) I think my home automations would have **[Pattern]**...
    (b) I would want **[Pattern]** in my home automations...
    (c) I would want to avoid **[Pattern]** in my home automations...
    (d) I think other people would want **[Pattern]** in their home automations...
13. Would people want help from a tool to find the "**[Pattern]**" pattern in their home automations?
    ○ Definitely ○ Probably ○ Probably Not ○ Definitely Not
    (a) I would need help finding **[Pattern]** in my home automations...
    (b) I would want help finding **[Pattern]** in my home automations...
    (c) I would use a tool if it looked for **[Pattern]**...
    (d) It would be annoying if the tool looked for **[Pattern]**...

TAP Pattern task 2 *This task is similar to the previous one except that the participant is asked to identify 2 examples of each **[Pattern]**. Each participant is shown 2 properties, randomly selected from Table 9. The section begins with an explanation of **[Pattern]**, including an example. Half of the participants are asked to identify the example of **[Pattern]** first, the other half are shown the Likert questions first.*

Bug fixing task

For this task, participants are randomly assigned to one of three groups which determines how much feedback they get to help them repair the bugs in this task: rule only, rule and state, or a full trace evets leading to the bug. Each participant is given 2 programs to fix. The scenario, program, and feedback given to each group is shown in Table 10.

Next, we are going to ask questions about what actions you would take if a tool reported possible problems with your home automations. Suppose you told a tool that your goal is **[Scenario]**. You have the following home automations installed: **[Buggy program]**.

14. What problem is happening here?
    ○ A home automation is missing ○ A home automation is

Property	Program
Different Triggers, Same Behavior [4]	“IF user arrives at home THEN lock door” “IF user leaves home THEN lock door”
Same Except No Condition [24]	"IF it begins raining THEN close window" "IF window is opened WHILE it is raining THEN close window"
Same Triggers, Different Behavior & Conditions [6, 7, 24, 27]	“IF the temperature is >74F WHILE the A/C is off THEN turn on A/C” “IF the temperature is >72F THEN turn on A/C”
Chains with Opposite Behaviors [7, 24]	“IF temperature <74F THEN turn off A/C” “IF temperature >78F THEN turn on A/C”
Chain [6, 7, 9, 19, 27]	"IF temperature >78F THEN turn on A/C" "IF temperature <73F THEN turn off A/C"
Different Triggers, Different Behaviors	"IF it begins raining THEN set light color to blue" "IF it becomes sunny THEN set light color to yellow"

Table 8: Example of each pattern for S2. We include citations for patterns which are inspired by prior work.

Anti-Pattern Name		Program
Loops [1, 5, 7, 16, 24]	(1)	"IF I post a new Facebook status THEN post a new tweet" "IF I post a new tweet THEN post a new Facebook status"
	(2)	"IF temperature >70F WHILE A/C is off THEN turn on A/C" "IF temperature <73F WHILE A/C is on THEN turn off A/C"
Opposite Behaviors [1, 4, 5, 7, 15, 18, 19, 24, 27]	(1)	"IF motion is detected THEN turn on security camera" "IF the time is 6AM THEN turn off security camera"
	(2)	"IF temperature >70 THEN turn on A/C" "IF temperature <73 THEN turn off A/C"
Same Behaviors [4, 5, 18, 24, 27]	(1)	"IF temperature >74F THEN turn on A/C" "IF humidity >80% THEN turn on A/C"
	(2)	"IF it begins raining THEN close window" "IF window is opened WHILE it is raining THEN close window"
Un-Paired Automations [1, 12, 19]	(1)	"IF it begins raining THEN close window" "IF window is opened WHILE it is raining THEN close window"
	(2)	"IF a presence is detected THEN turn on security camera" "IF new reminder added THEN send notification"
Extended Behavior [1, 12, 27]	(1)	“IF the time is 7AM THEN begin brewing coffee” “IF user arrives at home THEN begin brewing coffee”
	(2)	"IF email is received THEN turn on light sequence" "IF user leaves work THEN turn off light"
Privacy [3, 5, 11, 18, 19, 21]	(1)	“IF user leaves home THEN turn on porch light” “IF user arrives at home THEN unlock door”
	(2)	"IF the time is 7AM THEN post a new tweet" "IF I post a new status on Facebook THEN post a new tweet"
Trust [5, 11, 18, 19, 21]	(1)	"IF someone I follow tags me THEN re-tweet their post” "IF I post a new status on Facebook THEN post a new tweet"
	(2)	"IF an email is received THEN flash lights" "IF new reminder added THEN send notification"

Table 9: Example of each pattern for S2. Participants are asked to identify the example of **[Pattern]** twice for this task. We include citations for patterns which are inspired by prior work.

	Scenario text	Buggy program
S1	Whenever I am not at home, I want my door to be locked	**"IF the time is 7AM THEN unlock door"** "IF the user leaves home THEN lock door" "IF the temperature is above 75 THEN turn on A/C" "IF the temperature is below 68F THEN turn off the A/C"
S2	I want my lights to blink only when smoke has been detected	**"IF new email received THEN blink lights"** "IF smoke detected THEN blink lights" "IF the time is 7AM THEN unlock door" "IF the user leaves home THEN lock door"
S3	Temperature above 72 should never happen	**"IF window opened THEN turn off the A/C"** "IF the time is 7AM THEN unlock door" "IF the temperature is below 68F THEN turn off the A/C" "IF the temperature is 71F THEN turn on the A/C"

	State	Full trace
S1	the time is 7AM	Initially, the time is 6AM, the user is at home, the door is unlocked, the temperature is 70F, and the A/C is off. Next, the time is 6:30AM and the user leaves home. An automation is triggered and the door is locked. Finally, the time is 7AM. An automation is triggered and the door is unlocked.
S2	new email received	Initially, the time is 6AM, the user is at home, the door is unlocked, the lights are not blinking, and no smoke is detected. Next, the time is 6:30AM and an email is received. An automation is triggered and the lights begin blinking.
S3	window opened	Initially, the time is 10AM, the window is closed, the door is unlocked, the temperature is 70F, and the A/C is off. The temperature is rising. Next, the time is 10:30AM, the temperature is 71F, and the user opens the window. An automation is triggered and the A/C turns on. The temperature begins falling. An automation is triggered and the A/C is turned off. The temperature begins rising. The time is 11AM and the temperature is 72F. Finally, the time is 11:30AM and the temperature is 73F.

Table 10: Bug-fixing task for S2. For each scenario we show the full program, and the feedback given to the participant to help them make a fix. The rule shown in bold text is the one shown to participants as feedback.

	Scenario text	Partial program
S1	The window should never be open while it is raining	"IF the user leaves home THEN lock door" "IF it begins raining THEN close window" "IF temp > 75 WHILE it is not raining THEN open window" "IF temp > 75 WHILE it is raining THEN turn on A/C"
S2	The temperature should never be above 75F for more than 1 hour	"IF it begins raining THEN close window" "IF the user leaves home THEN lock door" "IF smoke detected THEN blink lights" "IF the time is 7AM THEN unlock door"

	State	Full trace
S1	user opens window and it is raining	Initially, the temperature is 70F, the window is open, and it is not raining. Next, it begins raining. An automation is triggered, closing the window. Finally, the user opens the window.
S2	the temperature increases above 75F	Initially, the time is 10AM, the temperature is 75F, the window is open, and it is not raining. The temperature is rising. Next, the time is 10:30AM and the temperature is 76F. Next, the time is 11AM and the temperature is 77F. Finally, the time is 11:30AM and the temperature is 78F.

Table 11: Program writing task for S2. For each scenario we show the partial program, and the feedback given to the participant to help them complete the program.

doing something I don't want ○Multiple home automations are interacting to do something I don't want ○I don't know ○Something else *Free response*

15. What action would you take?
○Add another home automation ○Modify a home automation ○Delete a home automation ○Do nothing/I don't know ○Something else *Free response*

16. *If "Add another home automation" was selected for Question 15* Which format would the new home automation take?
○If-then ○If-while-then

17. *Wording depends on whether "If-then" or "If-while-then" was selected for Question 16* Please enter the missing fields for the "if-then" home automation, below. *Drill-down format*

18. *Wording depends on whether "Modify a home automation" or "Delete a home automation" was selected for Question 15* Which home automation would you modify? *Participant selects one of the TAP rules in* ***[Buggy program]***

TAP program completion task *For this task, participants are assigned to a group which determines how much feedback they get to help them complete the partial TAP program. They are assigned to the same group as the previous task. Each participant is given the same 2 programs to complete. The scenario, program, and feedback given to each group is shown in Table 11.*

Suppose you told a tool that your goal is **[Scenario]**. You have the following home automations installed: **[Partial program]**.

If the participant is given a rule as feedback: If the tool told you there was a missing home automation, what home automation would you add?

If the participant is given a rule and state as feedback: If the tool told you there was a missing home automation when: **[State]** What home automation would you add?

If the participant is given a full trace as feedback: If the tool told you there was a missing home automation after the following sequence of events: **[Full trace]** What home automation would you add?

19. Would you like to add a home automation in the "if-then" or the "If-while-then" format?
○If-then ○If-while-then

20. *Wording depends on whether "If-then" or "if-while-then" was selected for Question 19* Please enter the missing fields for the "if-then" home automation, below. *Drill-down format*

Smart Devices Owned	S1	S2
Voice Assistant	81.6%	78.8%
Smart TV	78.7%	83.5%
Smart Lightbulb or Switch	36.2%	44.7%
Doorbell Camera	28.2%	35.9%
Security Camera	27.6%	39.2%
Smart Thermostat or A/C	26.4%	23.8%
Smart Vacuum or Mop	12.6%	17.9%
Smart Lock	9.8%	8.8%
Baby Monitor	9.2%	13.2%
Other	7.5%	8.8%
Smart Smoke or CO Detector	5.7%	5.1%
Smart Lawn Mower or Sprinkler	0.6%	2.9%

Table 12: Percent of participants that own smart home devices from various categories.

C Device Background and Experience

Table 12 shows the percent of participants that own each type of smart home device. Most participants have a smart TV or smart speaker. Other device types are less common.

D Statistical Results for TAP Goals (RQ1)

To determine which goals for TAP were most important to participants, we conducted 15 pairwise Wilcoxon signed-rank tests, corrected with Holm's sequential Bonferroni procedure, between each pair of goals. In each test, we paired each participants' responses to one goal to their response in the other goal. The tests indicated significant differences in the proportion of responses for every pair of goals except for (Home Security, Home Safety), (Comfort and Convenience, Understanding Failures), and (Comfort and Convenience, Privacy).

E Template Preferences (RQ2)

Table 13 shows the proportion of responses for each of the policies in the template preferences task as well as the templates involved in each policy and their attributes.

F Anti-pattern preferences (RQ3)

Table 14 shows the proportion of participants who report needing/wanting help identifying TAP anti-patterns, as well as how many would use a tool that looks for the anti-patterns or would be annoyed by a tool looking for the anti-patterns.

Scenario	Tool	Template(s)	Policy Structure	Sentiment	% Preferred
Temperature	AutoTap	State-State Conditional	2 Templates	Negative	42%
	SafeTAP	Always/Never	1 Template With And	Positive	38%
	SafeTAP	Always/Never	1 Template With Negation + Or	Negative	13%
	N/A	None of the Above	N/A	N/A	5%
	AutoTap	One-State Duration	2 Templates	Negative	2%
Smoke	AutoTAP	Event-State Conditional	1 Template	Positive	50%
	SafeTAP	Whenever	2 Templates	Positive	24%
	SafeTAP	Only When	1 Template	Positive	21%
	AutoTap	Multi-State Unconditional	2 Templates	Negative	4%
	N/A	None of the Above	N/A	N/A	1%
Security Camera	SafeTAP	Whenever	2 Templates	Positive	67%
	Both	Whenever, Event-State Conditional	3 Templates	Negative	16%
	AutoTap	State-State Conditional	1 Template With Or	Positive	11%
	AutoTap	Multi-State Unconditional	2 Templates	Negative	5%
	N/A	None of the Above	N/A	N/A	1%
Humidity	SafeTAP	Always/Never	1 Template With And	Positive	46%
	SafeTAP	Always/Never	2 Templates	Negative	29%
	AutoTap	One-State Unconditional	1 Template With And	Positive	14%
	AutoTap	One-State Unconditional	2 Templates	Negative	9%
	N/A	None of the Above	N/A	N/A	2%
Forecast	SafeTAP	Whenever	Consistent Conjunctions	Positive	42%
	SafeTAP	Only When, Whenever	Mixed Conjunctions	Positive	25%
	Both	Whenever, Event-Event Conditional	Mixed Conjunctions	Positive	17%
	Both	Only When, Event-Event Conditional	Consistent Conjunctions	Positive	11%
	N/A	None of the Above	N/A	N/A	5%

Table 13: Proportion of survey responses for most natural-sounding policies across five scenarios.

Pattern	**Percent of Participants Who...**			
	Need Help	Want Help	Would Use Tool	Would Be Annoyed by Tool
Different Triggers, Same Behavior	41.1%	56.7%	65.6%	23.3%
Same Behaviors	43.6%	56.4%	73.1%	24.4%
Different Triggers, Different Behaviors	47.3%	61.5%	71.4%	22.0%
Trust	52.5%	60.0%	68.8%	25.0%
Loops	48.7%	65.4%	75.6%	25.6%
Privacy	59.2%	71.1%	75.0%	15.8%
Chain	55.3%	74.5%	78.7%	18.1%
Same Except No Condition	61.8%	73.0%	76.4%	23.6%
Opposite Behaviors	54.4%	72.2%	68.4%	31.6%
Un-Paired Automations	68.4%	77.6%	76.3%	25.0%
Same Triggers, Different Behavior & Conditions	65.9%	75.8%	89.0%	26.4%
Extended Behavior	64.6%	73.4%	69.6%	25.3%
Chains with Opposite Behaviors	59.3%	63.7%	70.3%	23.1%

Table 14: Percent of participants who need/want help identifying TAP anti-patterns, whether they would use a tool to help identify such anti-patterns, and whether a tool would be annoying. Anti-patterns are sorted in order of easiest to understand to hardest to understand, based on participants responses in Figure 3.