SkillScanner: Detecting Policy-Violating Voice Applications Through Static Analysis at the Development Phase

Song Liao
Clemson University
liao5@g.clemson.edu

Long Cheng
Clemson University
lcheng2@clemson.edu

Haipeng Cai
Washington State University
haipeng.cai@wsu.edu

Linke Guo
Clemson University
linkeg@clemson.edu

Hongxin Hu
University at Buffalo
hongxinh@buffalo.edu

ABSTRACT

The Amazon Alexa marketplace is the largest Voice Personal Assistant (VPA) platform with over 100,000 voice applications (*i.e.*, skills) published to the skills store. In an effort to maintain the quality and trustworthiness of voice-apps, Amazon Alexa has implemented a set of policy requirements to be adhered to by third-party skill developers. However, recent works reveal the prevalence of policy-violating skills in the current skills store. To understand the causes of policy violations in skills, we first conduct a user study with 34 third-party skill developers focusing on whether they are aware of the various policy requirements defined by the Amazon Alexa platform. Our user study results show that there is a notable gap between VPA's policy requirements and skill developers' practices. As a result, it is inevitable that policy-violating skills will be published.

To prevent the inflow of new policy-breaking skills to the skills store from the source, it is critical to identify potential policy violations at the development phase. In this work, we design and develop SkillScanner, an efficient static code analysis tool to facilitate third-party developers to detect policy violations early in the skill development lifecycle. To evaluate the performance of SkillScanner, we conducted an empirical study on 2,451 open source skills collected from GitHub. SkillScanner effectively identified 1,328 different policy violations from 786 skills. Our results suggest that 32% of these policy violations are introduced through code duplication (*i.e.*, code copy and paste). In particular, we found that 42 skill code examples from potential Alexa's official accounts (*e.g.*, "alexa" and "alexa-samples" on GitHub) contain policy violations, which lead to 81 policy violations in other skills due to the copy-pasted code snippets from these Alexa's code examples.

CCS CONCEPTS

• **Software and its engineering** → **Automated static analysis**;
• **Security and privacy** → **Privacy protections**.

KEYWORDS

Amazon Alexa, Policy Violation Detection, Static Analysis

CCS '23, November 26–30, 2023, Copenhagen, Denmark
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0050-7/23/11.
https://doi.org/10.1145/3576915.3616650

ACM Reference Format:
Song Liao, Long Cheng, Haipeng Cai, Linke Guo, and Hongxin Hu. 2023. SkillScanner: Detecting Policy-Violating Voice Applications Through Static Analysis at the Development Phase. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3576915.3616650

1 INTRODUCTION

Amazon Alexa is one of the leading Voice Personal Assistant (VPA) platforms that allow third-party developers to build new voice applications (*i.e.*, called skills) and publish them to the skills store [10]. The openness of the Amazon Alexa platform has greatly attracted skill developers and inflated VPA's capabilities. Currently, more than 100,000 Alexa skills are available [5], with functions such as ordering pizza, listening to the news and weather, locking doors, or checking credit card balance. However, such an open VPA ecosystem inevitably provides unscrupulous or inexperienced developers an opportunity to publish buggy or dangerous skills in the store [32, 62]. Consequently, these poor-quality and problematic skills could cause user frustration, stir up negative effects on engagement, and even place end users in a vulnerable position.

To ensure the content safety and privacy of skills in Alexa skills store, Amazon has defined various policy requirements, including 7 privacy requirements [7], 14 main sections of content guidelines [6], and code inconsistency [1] to be adhered to by third-party skill developers. For example, skills should not have advertisements or promote alcohol. Although these policies are checked during the skill certification process (which rejects a skill if it violates any of the pre-defined policies), prior work demonstrated the ease of policy-violating skills being certified [32, 55]. Several recent works [39, 41, 42, 53, 59] developed tools to measure the policy compliance of skills on the Amazon Alexa platform through a dynamic analysis approach (*i.e.*, by exploring the outcomes of skills). For example, SkillExplorer [39] found 1,141 skills that collect different types of private information but without disclosing the data collection in their privacy policies.

In particular, researchers in SkillDetective [59] found 3,473 Alexa skills violating the same policy whereby skills are forbidden to "explicitly request that users leave a positive rating of the skill". It is likely that many third-party developers were not completely aware of such policy requirements, and inadvertently violated them in their skills. Past research has shown that software developers find it difficult to understand various policies and requirements when they develop software applications [50, 52], and are often unaware of all the related policies [48]. In this work, we are curious about whether developers are aware of the various policy requirements defined by the VPA platform. To this end, we first conduct an in-depth user study to understand third-party skill developers' perceptions and interpretations regarding VPA's policy requirements. Our user study results show that there exists a notable gap between VPA's policy requirements and skill developers' practices. To prevent the inflow of new policy-violating skills to the public, it is critical to identify potential policy violations in skills early in the skill development lifecycle.

In this work, we seek to develop static analysis techniques to facilitate the development of policy-compliant skills by third-party developers. However, the unique code structure of skills poses challenges for performing a static analysis of the skill code since we need to consider the interaction between the front-end code and back-end code during our analysis. The diverse nature of policy requirements defined by Amazon Alexa is another challenge for detecting policy violations in skill code. To address these challenges, we propose SkillScanner to automatically evaluate skills' conformity to various policy requirements before their deployment. Compared with existing dynamic analysis works [39, 53, 59], SkillScanner takes a static analysis approach and is expected to identify more code-specific violations that the dynamic analysis approach cannot find. In summary, we make the following contributions:

- To our best knowledge, SkillScanner is the first static analysis tool to facilitate the development of policy-compliant skills by third-party developers. SkillScanner is able to detect various policy violations and code defects, which can effectively help developers improve the quality of their code, for a sustainable VPA ecosystem. We shared the SkillScanner tool, related datasets and results with the community to facilitate future research[1].
- To understand the root causes of the prevalence of policy-violating skills in the skills store, we conducted a user study with 34 third-party skill developers. The results suggest that most skill developers in our study were not completely aware of the policy requirements defined by Amazon Alexa.
- We collected a benchmark dataset for Alexa skill code with 2,451 open-source skills from the GitHub. We conducted a comprehensive analysis of these skills using SkillScanner and evaluated the performance of SkillScanner. We identified 1,328 different types of policy violations among 786 skills. 694 of them are about privacy issues and 298 skills violate the content guidelines defined by Amazon Alexa. 32% of these policy violations are because of code duplication (*i.e.*, code copy&paste). Table 9 summarizes our detection results. Surprisingly, we found that 42 skill code examples from Alexa's official accounts contain policy violations, which led to 81 policy violations in other skills due to the code duplication. We discuss the responsible disclosure in Section 6.8.3.

2 BACKGROUND AND CHALLENGES

2.1 Skill Code Structure

A skill has a front-end interaction model (*i.e.*, front-end code[2]) and back-end code that processes requests and tells a VPA device what to respond. Amazon Alexa Cloud provides hosting for the front-end interface of a skill [16], but its back-end code can be hosted on the developer's server (*e.g.*, either hosted by AWS Lambda under the developer's account or other third-party servers). In addition, the skill code also contains a skill manifest file (*i.e.*, named "skill.json") [17], which stores the skill name, category, description, privacy policy, as well as permission information. Once a skill is certified and published in the Alexa's skills store, such information will be shown on the skill's web page.

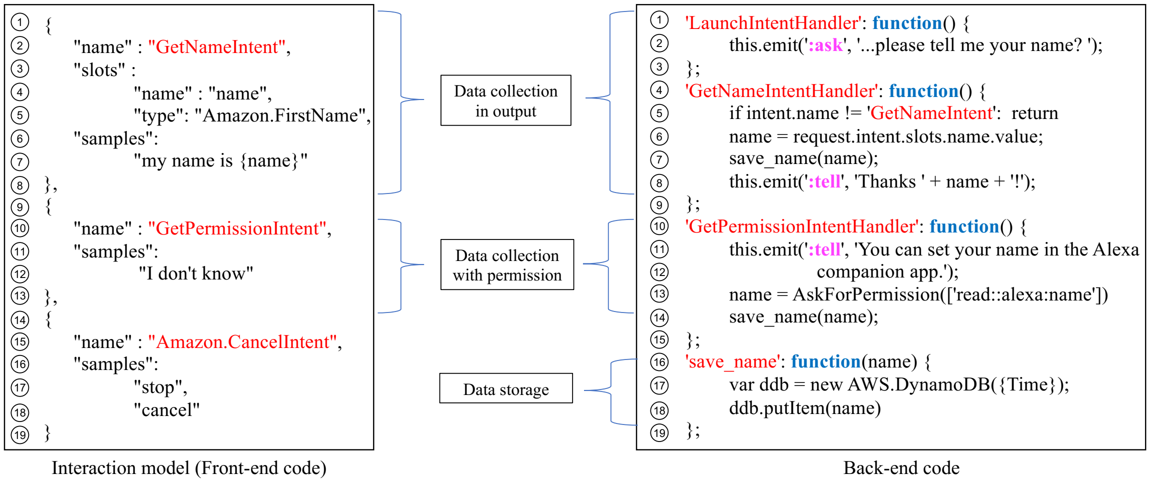
Front-end code. There are mainly three types of data defined in the front-end code: *intent*, *slot* and *sample utterances*. An intent represents an action that fulfills a user's spoken request. Intents normally have two values: the intent name and sample utterances. Sample utterances are a set of likely spoken phrases mapped to the intents and developers should include representative phrases so that the interaction model can better learn the sentence pattern. In addition, intents can optionally have arguments called slots. Slot is the variable that can capture a specific type of user's verbal reply, such as username or user address.

When a skill is created, several default Amazon Alexa built-in intents are created automatically for basic functionality, *e.g.*, `Amazon.StopIntent` for stopping a skill, and `AMAZON.CancelIntent` for canceling an instruction [20]. Developers can also create new intents by using an Alexa's pre-defined intent or creating a custom intent. In the latter case, developers define the intent name and then provide some sample utterances so that when users give a reply semantically close to these utterances, the intent will be matched and invoked to process the user's request. An intent can either be used for providing services (*e.g.*, telling a story) or collecting data from users (*e.g.*, asking for names from users). If developers want to use the intent to collect data from users, they need to define slots and specify the slot type. The slot type defines what data the slot collects, and developers can use Alexa's built-in slot types, such as "Amazon.FirstName" trained with thousands of popular first names and "Amazon.City" for local and world cities. For example, in Figure 1, `GetNameIntent` is a custom intent. It contains a slot "`name`" with the type "Amazon.FirstName", and the sample utterance is "My name is {name}". Then, when a user replies "My name is Jack", "Jack" will be extracted as a slot value and then passed to the back-end code.

Back-end code. A skill's back-end code includes a list of intent handler functions for processing different intents defined in the front-end code. For example, `GetNameIntentHandler` in Figure 1 is an intent handler corresponding to the intent `GetNameIntent` in the front-end code. Several built-in intent handlers are provided by Amazon Alexa corresponding to the built-in intents, such as `HelpIntentHandler` for providing helpful information. If no intent is triggered, the `FallbackIntentHandler` will be invoked and provide an output "Sorry, I don't know about that. Please try again" by

[1] The details of our tool implementation, evaluation results and representative skills are available at https://github.com/CUSecLab/SkillScanner.

[2] According to Alexa Developer Documentation [18], a skill package includes the skill's front-end interaction model file, back-end source code files, and skill manifest file. We refer to the front-end interaction model as front-end code in our analysis.



```
① {
②     "name" : "GetNameIntent",
③     "slots" :
④             "name" : "name",
⑤             "type": "Amazon.FirstName",
⑥     "samples":
⑦             "my name is {name}"
⑧ },
⑨ {
⑩     "name" : "GetPermissionIntent",
⑪     "samples":
⑫             "I don't know"
⑬ },
⑭ {
⑮     "name" : "Amazon.CancelIntent",
⑯     "samples":
⑰             "stop",
⑱             "cancel"
⑲ }
```

```
① 'LaunchIntentHandler': function() {
②       this.emit(':ask', '...please tell me your name? ');
③ };
④ 'GetNameIntentHandler': function() {
⑤       if intent.name != 'GetNameIntent':  return
⑥       name = request.intent.slots.name.value;
⑦       save_name(name);
⑧       this.emit(':tell', 'Thanks ' + name + '!');
⑨ };
⑩ 'GetPermissionIntentHandler': function() {
⑪       this.emit(':tell', 'You can set your name in the Alexa
⑫                 companion app.');
⑬       name = AskForPermission(['read::alexa:name'])
⑭       save_name(name);
⑮ };
⑯ 'save_name': function(name) {
⑰       var ddb = new AWS.DynamoDB({Time});
⑱       ddb.putItem(name)
⑲ };
```

Data collection in output
Data collection with permission
Data storage
Interaction model (Front-end code)
Back-end code

Figure 1: Example of the front-end and back-end code of an Alexa skill.

default. Typically, an intent handler processes the received data, performs skill functions and provides a reply to users. In Figure 1, the GetNameIntentHandler retrieves the received name value from the user's verbal input, stores it in a local database, and generates an output to thank the user for providing the name.

2.2 Challenges

The unique code structure of skills poses a challenge for data flow analysis. The back-end code of skill is tightly coupled with the front-end code. User input is first sent to the front-end interaction model, matched with intent, and then processed by the back-end code, as shown in Figure 1. In this example, there is a data collection request in line 2 of the back-end code. From the front-end code, we know that if users provide a name, it will be matched with the intent GetNameIntent (lines 2-8 in the front-end code) and then transferred to the function GetNameIntentHandler in the back-end code. Then, inside the function, the name data is stored in a local database and used for output (lines 16-18 and line 8 in the back-end code). The Amazon Alexa platform also provides APIs for skills to collect the information of device address, customer name, email address, phone number and location. In Figure 1, if the user doesn't provide a name (*e.g.*, saying "I don't know"), the GetNamePermissionIntent will be invoked and the skill will collect the user's name information using Alexa's built-in permission API AskForPermission in line 13 of the back-end code. Due to the unique code structure of skills, existing static analysis tools cannot be directly applied to analyze the skill code since they are unable to adequately model the interaction between the front-end and back-end code. In SkillScanner, we develop static analysis techniques to extract connections between the front-end code and back-end code.

No existing benchmarks of skill code for static analysis research. As discussed in Section 2.1, the back-end code of skills is hosted on the developer's server and developers don't need to submit the code for skill certification. For this reason, existing works only focused on the dynamic testing [39, 53, 59] and analyzing the metadata of skills, such as permission information [35, 36], to identify potential issues in skills. We address this challenge by collecting a skill code dataset from the GitHub. We have made our benchmark skill code dataset available to the community to facilitate future research.

3 UNDERSTAND DEVELOPERS' PERCEPTION AND PRACTICE REGARDING POLICY COMPLIANCE

We conduct a user study to understand the gap between VPA's policy requirements and skill developers' practices. The user study has been approved by our university's IRB office. We briefly describe the recruitment, survey questions, and results of our user study.

3.1 Recruitment and Survey Questions

There are skill developers leaving their emails in the skill descriptions for user feedback. We built a crawler to collect skill developers' contact emails from the US skills store and obtained 1,568 developer emails. We used the Qualtrics platform [13] to build survey questions and reach out to these developers to invite them to participate in our user study. We included a pre-screening phase to ensure that the participants do have skill development experience. Finally, we received 44 responses but 10 participants' responses had to be removed due to the meaningless answers. Our survey was conducted in August 2022, and we provided a $10 Amazon gift card to each valid participant. The average time for completing the survey was 16 minutes. Although most of the participants are from the United States, there were a few developers from other countries such as India and Italy. 10 participants developed more than 10 skills and the average number of skills developed by these participants is over 5, which indicates that they have enough coding experience in skill development. 8 participants also have experience of developing Google actions.

Table 1 lists several selected user study questions due to space. The complete survey questions and responses are available in our GitHub repository (https://github.com/CUSecLab/SkillScanner), which is composed of three sections. First, we asked developers whether they are aware of Amazon's policy requirements. Next, we asked developers whether they could identify any policy violation in skill outputs from 8 example skills (in which there are 5 policy violations in these skills). Finally, we asked developers whether they routinely perform consistency checking during their skill development, such as code, content, privacy policy and whether they think that a static analysis tool can be helpful for them in developing policy-compliant skills.

Question	Response	Developers
Q1: Are you aware that there are some platform required policies?	Completely aware	65%
	Somewhat aware	32%
	Neither aware nor unaware	0%
	Somewhat unaware	3%
	Completely unaware	0%
Q2: Do you read the Alexa's policy requirements before developing skills?	Always	24%
	Most of the time	26%
	About half the time	32%
	Sometimes	9%
	Never	9%
Q3: Do you think the Amazon Alexa platform should provide a policy compliance training for developers before they develop skills?	Definitely Yes	29%
	Probably Yes	47%
	It doesn't matter to me	6%
	Probably not	18%
	Definitely not	0%
Q4: Do you think the Amazon Alexa platform should check the skill code to determine whether it violates any policies?	Definitely Yes	24%
	Probably Yes	47%
	It doesn't matter to me	18%
	Probably not	6%
	Definitely not	6%
Q5: Do you check whether your privacy policy and code are consistent?	Always	26%
	Most of the time	24%
	About half the time	35%
	Sometimes	3%
	Never	12%
Q6: Do you fully inspect your code and remove any inconsistent code before submission?	Always	35%
	Most of the time	32%
	About half the time	9%
	Sometimes	18%
	Never	6%

Table 1: Selected user study questions and responses.

3.2 Survey Results

Although 33 skill developers (out of 34 participants) claimed that they are "completely aware" or "somewhat aware" that there are some platform required policies (Q1 in Table 1), most participants couldn't correctly recognize the 5 policy violations in our example skills (results are shown in Table 2). When asked "Do you read Alexa's policy requirements before developing skills?" (Q2), only 8 developers selected "always read" and 3 even selected "never". 26 participants agreed that "VPA platform should provide a policy compliance training for developers before they develop skills" (Q3) and 24 participants thought "VPA platform should check the skill code about policy violation" (Q4). Although 17 developers selected they would "always" or "most of the time" check whether privacy policy and code are consistent (Q5), and 23 developers mentioned they would "always" or "most of the time" check their code consistency before submission (Q6), SkillScanner identified hundreds of skills with inconsistency issues in our skill code dataset (details in Section 6). At last, most participants thought that a static analysis tool for checking skill code could be "extremely" or "very" useful for policy-compliant skill development.

Table 2 lists the policy violations in our example skills as well as users' response results. For each skill, we asked developers, "Do you think this skill violates any Amazon policy requirements? If so, which policy does it violate?". 21 (62%) participants reported that the second skill (S2) contains a policy violation (*i.e.*, collecting kids personal data) and gave the right reason. Only 13 (38%) developers were aware that "asking for a positive rating" is a policy violation (S3). This partially explains why there are considerable published skills in the skills store violating this policy [59]. For the other three policy-violating skills that "call emergency responder", "predict gender" and "have a disallowed invocation name" (invocation name has two words and one word is a definite article "the"), only 7, 1 and 5 developers correctly reported the violations, respectively. In addition, for the 3 skills without a violation, 21 developers wrongly thought they contained some policy violations. Our user study results show that there is a notable gap between VPA's policy requirements and skill developers' practices and perceptions. Such observations motivate us to develop a static analysis tool to facilitate third-party developers in developing policy-compliant skills as well as improving the quality of their code.

Skill index	Violation type	Skill output that contains a policy violation	# of users that reported a violation
S2	Collect kids data	Welcome to cake walk for kids! Can you tell me your birthday?	21 (62% ✓)
S3	Ask for a positive rating	If you like this skill, please give us a 5 star rating.	13 (38% ✓)
S6	Call emergency responder	We can also call hospital or emergency responders.	7 (21% ✓)
S7	Predict gender	Would you like Alexa to predict the gender of your baby?	1 (3% ✓)
S8	Disallowed invocation name	the birthday	5 (15% ✓)
S1, S4, S5	No violation	N/A	21 in total (21% ✗)

Table 2: Policy violations in example skills and the responses from 34 participants (✓ means a correct answer and ✗ means an incorrect answer).

4 SKILLSCANNER OVERVIEW

Threat Model. The proposed SkillScanner is mainly designed for benign third-party developers to identify potential policy violations in skills at the development phase. We assume that inexperienced developers may unknowingly develop and publish policy-breaking skills to the public, since our user study results in Section 3 demonstrate that they are not completely aware of the related policies. These problematic skills could pose privacy, safety, and security threats to VPA users. SkillScanner is designed to run locally (which also protects developers' proprietary code) to identify potential policy violations and inconsistencies in skill code. Thus, we assume that the source code of skills is provided by developers as inputs of the tool. Since SkillScanner aims to help benign developers improve the policy compliance of their skills, for adversarial application scenarios such as a developer intends to violate the policy, it is out of this work's scope. SkillScanner can be potentially used by the Amazon Alexa platform for policy violation detection if developers provide access to their skill code to Amazon Alexa during the skill certification process.

System Overview. With the goal of ensuring privacy and policy compliance in the Amazon Alexa ecosystem, SkillScanner mainly focuses on detecting violations of privacy requirements [7] and skill

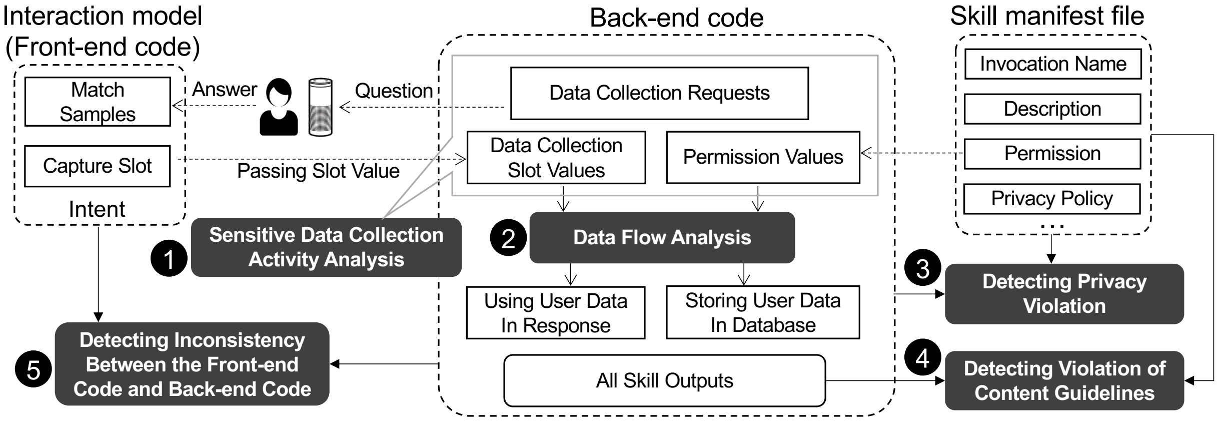

Interaction model (Front-end code)
Match Samples
Capture Slot
Intent
Answer
Question
Passing Slot Value
Back-end code
Data Collection Requests
Data Collection Slot Values
Permission Values
Skill manifest file
Invocation Name
Description
Permission
Privacy Policy
...
1 Sensitive Data Collection Activity Analysis
2 Data Flow Analysis
3 Detecting Privacy Violation
Using User Data In Response
Storing User Data In Database
All Skill Outputs
4 Detecting Violation of Content Guidelines
5 Detecting Inconsistency Between the Front-end Code and Back-end Code

Figure 2: System Overview of SkillScanner.

content guidelines [6] (which all skills must adhere to according to the Amazon Alexa documentation), as well as inconsistencies in skill code. Figure 2 shows the design overview of SkillScanner. To detect data collection related issues, we first extract data collection activities by analyzing both the back-end code and front-end code (❶). A skill can collect user data through the conversational interface or using Alexa's built-in permission APIs. For collecting user data through the conversational channel, a skill normally asks questions such as "How old are you?" in the back-end code. We extract all potential skill outputs and then conduct an NLP analysis of these outputs to check whether they collect any user data. Then we find corresponding data collection slots in the front-end code by checking whether the slot name and slot type contain any data collection keywords. Developers can also directly collect user data by requesting for specific permissions to access user data in the back-end code. We identify such data collection activities by checking if any permission API is used in the code. Next, SkillScanner tracks how the collected data is used in skill code by conducting a taint analysis (❷). The source of taint analysis can be a data collection slot value or the returned value from a permission API in the back-end code. A skill may use the collected user data in response (*e.g.*, to establish a rapport with users), store the data in a local database, or never use the data in the back-end code.

After the taint analysis, we check whether the data collection and data usage in code are consistent with what developers claimed in the skill manifest file (❸). For example, developers are required to provide a privacy policy document to outline data collection and usage, and there might be inconsistencies between the privacy policy and the actual data collection activities in the code. In addition, developers may request for more or less permissions than they actually need. Amazon also defines a diverse range of content guidelines, ranging from skill output content safety to user review manipulation (*e.g.*, explicitly requesting 5-star ratings). To ensure compliance with content guidelines, SkillScanner analyzes all skill outputs to check whether there exists any violation of the content guideline (❹). At last, a poorly-written interaction model (*i.e.*, the front-end code) may trigger incorrect intent and return users with content they don't expect. Such poor user experience can lead to user frustration and a decrease in user engagement. To facilitate developers to improve the quality of their code, SkillScanner also detects inconsistencies in the back-end and front-end code (❺). With the help of SkillScanner, developers can improve their code quality based on the policy violation report and provide a better user experience of skills.

5 SYSTEM DESIGN AND IMPLEMENTATION

5.1 Sensitive Data Collection Activity Analysis

To learn what data a skill is able to collect, we need to analyze both the back-end and front-end code. The front-end code is stored in a JSON file and the back-end code is written in JavaScript or Python language. We first define the sensitive/personal data types that are considered in SkillScanner, including 1) 24 types of PII (personally identifiable information) from a NIST (National Institute of Standards and Technology) report [46] and Amazon [14], 2) Amazon's built-in slot types [19] such as "Amazon.Person" for getting the user name and "Amazon.US_City" for US cities; 3) specific types of data that can be collected through the permission APIs [2]; and 4) common health information. The keywords about data collection are listed in Table 3.

5.1.1 Identifying conversational sensitive data collection activities. Our analysis starts from identifying data collection requests (*e.g.*, "What is your name" or "Please tell me your name") from a skill's possible outputs. We search for all strings in the back-end code (excluding comments). Given an output extracted from a string, we apply an NLP-based method and use the Spacy library [21] to check if any data collection keyword is used as a noun because some keywords like "address" can be used as a verb. Developers may also provide their own data, so we check whether any PII data is used as a noun following the word "your". If a sentence contains "your + sensitive data collection noun", we consider it as a data collection request. In addition, we check the output with a list of common sentences of personal data collection [59], such as "how old are you" or "what can I call you" to improve accuracy. To track whether user provided data can be properly processed and passed to the front-end code, we need to analyze slots and sample utterances of each intent in the front-end code. As introduced in Section 2, when users hear a question and make a reply, there should be an intent and slot for processing the reply. If a slot includes any data collection keyword listed in Table 3 in its name or slot type, we define such slot as a

Personally Identifiable Information (PII) [14, 46]	Address, Name, Email, Birthday, Age, Gender, Account, Location, Contact, Phonebook, Profession, Income, Zipcode, Postal code, Phone number, Passport number, Driver license number, Bank account number, Debit card number, Credit card number, Credit card verification code, Taxpayer identification number, Social Security number (SSN), Vehicle identification number (VIN)
Amazon's built-in slot types	FirstName, Person, US_FirstName, PhoneNumber, PostalAddress, Region, RelativePosition, City, US_City, AdministrativeArea, StreetAddress, StreetName, US_State, Professional, ProfessionalType
Data types supported by permission APIs	Profile: Name, Given_name, Email, Mobile_number Devices: Address, CountryAndPostalCode, Geolocation
Health Information	Height, Weight, Blood group, Blood pressure, Blood glucose, Blood Oxygen, Heart rate, Body temperature, Sleep data, Fat percentage, Mass index, Waist circumference, Menstruation, Period

Table 3: Keywords related to personal data collection.

sensitive data collection slot, *e.g.*, a slot named as "username" with the slot type "Amazon.FirstName". Similar to identifying sensitive data collection requests, if a sample utterance includes "my + {data collection slot}", we consider it as a data collection utterance.

We capture conversational data collection activities of a skill by checking the following three conditions: 1) whether there exists a data collection request asking for user data; 2) whether there is any sample utterance that can be matched to potential user replies to this data collection request; and 3) whether a slot of the sample utterance is used to capture sensitive user data. The back-end code could not correctly get user data through the conversational interface if missing any one of these conditions. Once we identify conversational data collection activities in the skill code, the corresponding slot values are used as taint sources in our data flow analysis in Section 5.2.1.

5.1.2 Identifying data collection permission requests. Another way for a skill to collect user data is to request permissions for customer information, which is from the user's Alexa account. Amazon states that "Alexa skills might require personal information from the customer in order to provide relevant information in skill responses or to complete transactions". Table 3 lists the data types that are considered for permission-based data collection activity analysis in SkillScanner. In this method, developers first request for permissions in the Amazon Alexa developer console, and permission request information is stored in the skill manifest file. Before accessing data, a skill needs to get a user's grant to access permission data when a user first enables the skill. Once granted, the skill can retrieve the requested permission data using different methods in the back-end code: either directly using certain pre-defined endpoints or calling permission APIs. Since they exhibit distinct patterns, we can easily identify data collection permission requests in the back-end code. For example, in line 18 of Listing 1, the user email data is obtained from a permission request, and thus we take this value as the taint source of the data flow analysis.

5.2 Data Flow Analysis

If a skill contains any data collection activity, we track how the collected user data is used by conducting a taint analysis in the back-end code. Our objective is to check whether user data is properly used or saved into databases, which is useful for detecting privacy related violations in Section 5.3. For example, in Listing 1, we know the slot value `userName` and permission value `profileEmail` (in lines 8 and 18) are data collected from users. By tracking their data flows, we learn that these two values are used for generating customized responses (in lines 12 and 20) and storing to the database (in line 10).

```
1  const LaunchRequestHandler = {
2    handle(handlerInput){
3      const speakOutput = 'Hello! What is your name?'; // Ask
        for user name
4      return handlerInput.responseBuilder.speak(speakOutput).
        reprompt(speakOutput).getResponse();
5  },};
6  const CaptureUserNameHandler = {
7    handle(handlerInput){
8      const userName = handlerInput.requestEnvelope.request.
        intent.slots.name.value;          // Get name from slot
9      const sessionAttributes = handlerInput.attributesManager.
        getSessionAttributes();
10     sessionAttributes.name = userName; // Store name in
        database
11     const speakOutput = 'Thanks ${userName}';
12     return handlerInput.responseBuilder.speak(speakOutput).
        reprompt(speakOutput).getResponse();       // Use name
        for response
13 },};
14 const EmailIntentHandler = {
15   handle(handlerInput){
16     const { serviceClientFactory, responseBuilder } =
        handlerInput;
17     const upsServiceClient = serviceClientFactory.
        getUpsServiceClient();
18     const profileEmail = await upsServiceClient.
        getProfileEmail();  // Get email from permission
19     const speechResponse='Your email is ${profileEmail}';
20     return handlerInput.responseBuilder.speak(speechResponse)
        .reprompt(speechResponse).getResponse();  // Use email
        for response
21 },};
```

Listing 1: Real-world skill code with sensitive data collection.

5.2.1 Taint sources. A skill can obtain user data by retrieving slot values or permission values, which are the sources of our taint analysis. We describe how SkillScanner locates taint sources in the back-end code.

Slot values as taint sources. There are two ways in a skill's back-end code to retrieve slot values. 1) The first method is to send requests and get slot values. The back-end code can use a request handler to handle intent and get slot values with "handlerInput.requestEnvelope.request.intent.slots". 2) The second method is using the Alexa Skills Kit (ASK) SDK, which provides several functions for retrieving slot values. Developers can directly call "Alexa.getSlotValue" and use slot name as parameter to get a slot value. Accordingly, after identifying conversational data collection activities in a skill (in Section 5.1.1), we use the "intent.slots" and "Alexa.getSlotValue" as sources in our taint analysis.

Permission values as taint sources. A skill obtains permission values either through pre-defined endpoints or permission APIs in the back-end code. 1) A skill can directly access pre-defined endpoints to get customers' contact information and setting information such as address or postal code. Table 4 lists the pre-defined endpoints in Alexa, including two types of user data: device address (denoted as "/v1/") and customer profile (denoted as "/v2/"). For the customer profile, Alexa provides information about the name (full name and give name), email and mobile phone number. SkillScanner searches for all string values in code with endpoints listed in

Requested Information	Endpoints
Device Address/ Device Country and Postal Code	/v1/devices/{deviceId}/settings/address (/countryAndPostalCode)
Full Name/ Given Name/ Email/ Phone Number	/v2/accounts/~current/settings/Profile.name (/givenName/email/mobileNumber)
Full Name/ Given Name/ Phone Number	/v2/persons/~current/profile/name (/givenName/mobileNumber)

Table 4: Pre-defined endpoints to obtain permission data

Table 4 and treats the returned values of these endpoints as the taint sources. 2) The second approach of obtaining permission values is to use Alexa Service APIs. Amazon provides “DeviceAddressServiceClient” for getting device addresses and “UpsServiceClient” for customer profiles. For example, a skill can get the device address through “DeviceAddressServiceClient.getFullAddress()” or obtain user email with “UpsServiceClient.getProfileEmail()” (as shown in line 18 of Listing 1). We also take the returned values of these permission requests as the taint sources.

5.2.2 *Taint sinks.*

We mainly track two data usage cases in a skill, 1) using the collected user data in the response or 2) storing the data in a local database. Typically, a skill uses “handlerInput.reponseBuilder” to generate a verbal response by calling “.ask(output)”, “.speak(output)”, or “.reprompt(output)”. To track if the collected data is used in generating customized skill responses, we search for these functions in the back-end code and use them as taint sinks. Skills can also save the user data to a database so that they can use them for other purposes. If a skill needs to store data in a database, it usually first builds the database connection. Developers need to call specific APIs to create or retrieve data from a DynamoDB database provided by Amazon, or save data to the Alexa session attribute. These database APIs are considered as taint sinks in SkillScanner.

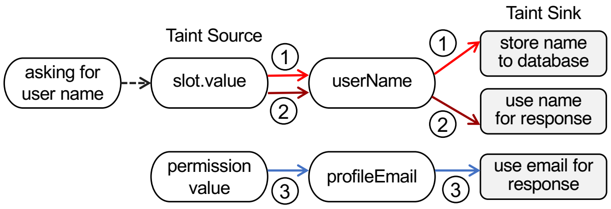


Figure 3: Data flows corresponding to Listing 1.

5.2.3 *Tracking data flows.*

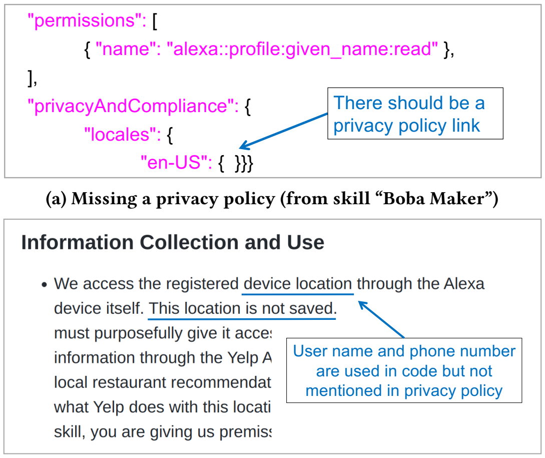
Since a skill can be written in different programming languages, we use the CodeQL tool [8] that supports multiple programming languages to track data flows of different variables in skill code. We demonstrate the performance of CodeQL in Section 6.3 and it performs well in our results. Figure 3 shows the data flow analysis results corresponding to the code in Listing 1. SkillScanner identifies three data flows, the first one is from the sensitive data collection slot value (`userName` in line 8 of Listing 1) to the database usage (line 10 of Listing 1) and the second data flow goes to a speaker output (line 12 of Listing 1). The third data flow is from the permission request of user email address (`profileEmail` line 18 of Listing 1) to a speaker output (line 20 of Listing 1).

5.3 Detecting Privacy Violation

Among the various policy violation problems reported by recent research [36, 39, 41, 43, 53, 59], many issues are related to the privacy policy. In particular, Jide *et al.* [36] performed a longitudinal measurement of privacy policies of skills across three years. They show that many developers were not engaging in good data practices, which is still an unresolved issue. In the skill manifest file, we can obtain several types of skill information such as skill name, category, description, privacy policy, permission, etc. SkillScanner detects problematic privacy policies and inconsistencies between the back-end code and the disclosed information provided by developers according to the skill manifest file.

5.3.1 *Problematic privacy policy.*

According to Alexa’s privacy requirements [7], it requires that skills with data collection activities must provide a privacy policy URL that links to a legally adequate privacy policy webpage. The privacy policy URL will be displayed to end users on a skill’s web page in the skills store. Given a skill source code package, SkillScanner first checks if the skill provides a valid privacy policy link (*i.e.*, not a broken link or leading to an unrelated webpage). Based on the results of the sensitive data collection activity analysis in Section 5.1 and taint analysis in Section 5.2, we check whether a skill’s data collection and usage are properly disclosed in its privacy policy. If a privacy policy is provided but the data practices are not mentioned in it, such privacy policy is incomplete since it doesn’t disclose the sufficient information. Figure 4 shows two skills that miss a privacy policy or provide an incomplete privacy policy. For each sentence in a privacy policy, we check whether it mentions any sensitive data collection or data storage activity. For the data collection activity, we check whether a data collection verb, such as “collect”, “ask” or “access” and the collected data appear in the same sentence. For the data storage activity, we check if the “store” verbs, such as “keep”, “retain” or “store”, and collected data are in the same sentence [28, 60].



```
"permissions": [
      { "name": "alexa::profile:given_name:read" },
],
"privacyAndCompliance": {
      "locales": {
            "en-US": {  }}}
```

There should be a privacy policy link

(a) Missing a privacy policy (from skill “Boba Maker”)

Information Collection and Use

- We access the registered device location through the Alexa device itself. This location is not saved. must purposefully give it acce: information through the Yelp A local restaurant recommendat what Yelp does with this locati skill, you are giving us premis:

User name and phone number are used in code but not mentioned in privacy policy

(b) Incomplete privacy policy (from skill “feed me now”)

Figure 4: Examples of problematic privacy policies.

5.3.2 *Over-privileged data collection permission requests.*

Over-privileged data collection permission requests refer to a skill requesting more permissions than it requires [54], and thus the skill violates the principle of least privilege. With the taint analysis, SkillScanner checks if a skill requests for permission data but never uses it in the back-end code. SkillScanner also detects if a skill tries to use a permission to retrieve user data but doesn’t ask for that permission first.

5.3.3 *Inconsistent data collection disclosure to Alexa platform.* Amazon provides a selection for developers before submitting a skill for certification that asks "Does this Alexa skill collect users' personal information?" and developers can select "yes" or "no" (they can also choose to not answer this question). Different from a privacy policy, which is provided for end users, such a question is for developers to disclose data collection behavior to the Alexa platform and certification team. Researchers in [32] observed that if a developer selects no to this question but his/her skill collects data through the conversational interface, it is more likely the skill could bypass the certification process. The developer's answer information can be found in the skill manifest file. Based on the results of our data collection activity analysis, SkillScanner checks if any inconsistent data collection disclosure between the back-end code and developers' answers.

5.4 Detecting Violation of Content Guidelines

The Amazon Alexa platform has defined a list of content guidelines [6], describing the inappropriate content that skill should not deliver to users. These involve disturbing content, false information, profanity, etc. In this section, we focus on ensuring the content safety and compliance of invocation names.

5.4.1 *Content safety violations.* SkillScanner mainly checks three types of content violations: toxic content , user review manipulation and content safety of external websites involved in a skill's back-end code. SkillScanner detects whether there exists potential toxic content in skill outputs by using Google's Perspective tool [11], which is a machine learning based tool for detecting toxic content. SkillScanner extracts all possible skill outputs from a skill's back-end code. If any skill output's *Toxicity* and *Profanity* scores are higher than the threshold (0.9), SkillScanner generates a warning of possible toxic content for developers.

To avoid user review manipulation, Amazon Alexa enforces a policy prohibiting "Explicitly requests that users leave a positive rating of the skill". SkillScanner checks whether a skill explicitly asks for the user to provide a positive rating in the skill outputs as well as the skill description. We detect this violation using an NLP-based method by checking whether keywords such as "5 star" or "five star" are used followed by verbs "give" or "leave".

A skill's back-end code may contain external websites from which the skill can call HTTP requests to get data (*e.g.*, pre-recorded audio streams and images) from an external data source. Modern VPA devices (*e.g.*, Amazon Echo Show) can display different media files including text, images and even movies. We consider three different cases that websites are involved in skill code: 1) retrieving audio data such as "mp3" files; 2) retrieving image data such as "jpg" files; or 3) retrieving textual content from external websites. For audio and image files, we first download them and then use speech-to-text [22] and image-to-text tools [12] to extract the textual data from these files. The output data is treated as normal skill output and checked whether it includes any policy violation such as data collection or toxic content. We also detect if an outside website is flagged by ViruslTotal [23] as a suspicious or malicious website. For the skills retrieving data from external websites, we check whether the webpage contains any inappropriate content according to Amazon Alexa's content guidelines.

5.4.2 *Non-compliance of invocation names.* Under the Amazon Alexa's content guidelines, there is a special category for skill invocation names and several requirements are listed, such as "one-word invocation names are not allowed" and "the invocation name must not contain any of the Alexa skill launch phrases". More details about invocation name requirements and disallowed words as well as representative violation cases we found using SkillScanner are listed in Table 5. Since a skill's invocation name information is stored in the front-end code, SkillScanner extracts a skill's invocation name and checks whether it violates the policy requirements.

Requirements on Invocation Name	Example invocation names with violation
One-word invocation names are not allowed, unless the invocation name is unique to your brand/intellectual property.	beeper, jokes
Two-word invocation names are not allowed if one of the words is a definite article ("the"), indefinite article or preposition.	the template, the radiator
The invocation name must not contain any of the Alexa skill launch phrases (such as "play", "launch", "open"), connecting words (such as "to", "about", "and") or wake words (such as "Alexa", "skill", "app").	play radio, to jeff, video app
The invocation name must contain only lower-case alphabetic characters, spaces between words, and possessive apostrophes.	AlTranslate, Ryan's note
The invocation name should be distinctive to ensure users can enable your skill.	hello world (used by 85 skills)

Table 5: Invocation name requirements [6].

5.4.3 *Category-specific violations.* Amazon defines specific policies for selected categories, such as "Kids" and "Health & Fitness" categories. For skills in the Kids category, they can't include unsuitable content, direct users to external websites, or collect any personal information [3], regardless of whether a privacy policy is provided. For data collection in Kids skills, we used the same method as described in Section 5.1 to detect data collection requests and the method in Section 5.4.1 to check skill outputs about external websites and toxic content. For the Health category, Alexa requires that the skill must "include a disclaimer in the skill description stating that the skill is not a substitute for professional medical advice" [4]. We checked whether the words "medical advice", "educational purpose" or "information purpose", which are keywords taken from an example disclaimer provided by Alexa, show up in a skill's description.

5.5 Detecting Inconsistency Between the Front-end Code and Back-end Code

As mentioned in Section 2.1, intents (as well as sample utterances and slots) in a skill's front-end code and the back-end code are tightly coupled. Typically, developers define a data collection request in the back-end code (*e.g.*, what's your name), create an intent based on possible user replies in the front-end code, and then create the corresponding intent handler for processing user replies in the back-end code. Inconsistency among them could trigger incorrect intents and impact user experience. For example, if a skill asks for user data through the conversation but doesn't have an intent for processing the user reply, the skill will trigger a built-in intent "FallbackIntent" which replies "Sorry, I don't know that.

Please try again". Amazon requires developers "reviewing the intent schema, the set of sample utterances, and the list of values for any custom slot types developers have defined to ensure that they are correct, complete, and adhere to voice design best practices" [1]. So, SkillScanner also detects potential bugs and improves the code quality early in the skill development lifecycle because of its influence on skill function and user experience.

SkillScanner detects three cases of code inconsistency. 1) A skill's back-end code has a request asking for user data, but there is no slot in the front-end code for processing the user reply. 2) A skill defines a slot to obtain users' possible data but there is no request in the back-end code asking for user data. 3) A data collection slot or an intent doesn't have any sample utterances. If so, they could not match with any user reply, and thus the slot or intent will never be triggered. The second case is actually a code vulnerability [32, 54] due to the fact that the Amazon Alexa platform does not require a re-certification when a change is made in the back-end code of a skill. If a skill has a data collection slot but doesn't have a data collection request asking for user data, after the skill has been certified and published in the skills store, developers can arbitrarily change the back-end code and ask for any type of user data. Benign developers may collect user data for non-malicious purposes but unintentionally violate policy after the code update (*e.g.*, collecting user names and using them to establish a rapport with users). SkillScanner reminds developers of such potential policy violations. Since Amazon Alexa can not check these consistencies at the skill certification phase, SkillScanner facilitates developers to detect potential code defects that are difficult to find for improvements.

6 EVALUATION

In this section, we evaluate SkillScanner by answering the following three research questions:

RQ1: *How effective is SkillScanner in capturing sensitive data collection and usage behaviors in skills? (§ 6.2 and § 6.3)*

RQ2: *How effective is SkillScanner in identifying policy violations in open source skills? (§ 6.4, § 6.5, and § 6.6)*

RQ3: *SkillScanner found many similar violations in different skills. What's the main reason for these similar policy violations? (§ 6.7)*

6.1 Skill Source Code Collection & Setup

Before discussing the evaluation results, we explain how we collected open source skill code from GitHub and the setup that we used to perform the evaluation.

In contrast to traditional apps on smartphone platforms (*e.g.*, Android or iOS) where apps run on host smartphones, a skill's back-end code runs on the developer's server, and is not available even for the Amazon Alexa platform. We can not obtain skill code from the Alexa's skills store. Instead, we collected open-source skill code from GitHub, which is one of the largest open-source project platforms. The unique file structure (*e.g.*, every skill comes with a front-end code file "en-US.json") of skill packages allows us to accurately locate skill code in GitHub. Specifically, we searched a combination of keywords "en-US.json" (front-end code filename), "interactionModel", "languageModel" and "intents" for the front-end code file using GitHub search API. Finally, we were able to find 2,451 skill repositories after removing duplicate skills and we shared the dataset to facilitate future research (https://github.com/CUSecLab/SkillScanner). Table 6 shows the statistics of skill code in our dataset.

Front-end code	Total # of custom intents	7,880
	Total # of slots	5,702
	Total # of sample utterances	67,884
Back-end code	Total # of functions	26,216
	Total # of skills	2,451

Table 6: Statistics of skill code in our dataset.

Since the skills in our dataset were written either in Python or Node.js, our data flow analysis module has two versions. We ran SkillScanner on a server with 3.0 GHz 6-core CPU and 16GB RAM. Finally, SkillScanner analyzed 2,451 skills in total and obtained 797,501 skill outputs, 1,656 HTML webpages, 1,439 mp3 files and 932 images in our experiments. The output of SkillScanner is a report that notifies developers about potential policy violations or bugs and the corresponding code location. Then, developers can fix these issues, otherwise the skill may fail in the certification process.

By comparing skills' names, developers and descriptions of skills in our dataset and skills in the Alexa skills store, we found that 93 skills in our dataset have been published in the skills store. We did notice that there exists "toy" projects in our dataset, which have a single contributor, few commits, and non-informative readme files. As previous works demonstrate, most skills are developed by third-party developers and these skills are easy to be published [32], and also the quality of these published skills is not good [39, 59]. In our dataset, 15% of skills that are published have a repository with 1 contributor, less than 10 commits and lack a readme file. Since these low-quality skills are also possible to be published to the skills store, we didn't remove them from our dataset.

6.2 Sensitive Data Collection Activity Analysis

After getting all possible outputs from each skill's back-end code, we observed that 321 skills contain sensitive data collection requests asking for user data through the conversational interface. 197 of them have sensitive data collection slots in front-end code to process the collected data. For the other 124 skills that don't have a data collection slot, we will discuss this issue in Section 6.6.1. Over half of the skills ask for the name information followed by the birthday, email and address information. Skills also ask for seven other types of data such as age, gender, location, phone number or zip code. The most common data collection requests are "You can introduce yourself by telling me your name" and "What is your name?". For example, the skill "E-Nurse" speaks out "Kindly tell me your name" when it is invoked. In the front-end code, it defines an intent called "NameIntent" and one slot "name" with slot type "Amazon.FirstName". Two sample utterances are "my name is {name}" and "I am {name}". In addition, we found that 133 skills request for permission data. For example, a skill named "Pizza Search" requests for 7 permissions including five sensitive data collection permissions for address, location, email, mobile number and name information. Overall, SkillScanner achieves high accuracy for the sensitive data collection activity analysis. After manually checking all the results, only 15 sentences are false positive cases out of 767 sensitive data collection requests, with an accuracy of 98%. SkillScanner achieves 99.7% accuracy in identifying data collection slots and 100% for permission requests. Most false positives occur in sentences that include "your + sensitive data collection noun" but do not ask for data, such as "Your location is around...". The recall rates for each part are 84%, 93%, and 91%, respectively. The false negatives are because of the diverse ways to ask for user data (*e.g.*, "How should I call you?" and "Please specify an address.") or inability to locate and read the skill file.

6.3 Data Flow Analysis

```
//back-end code:
const ScheduleTripIntentHandler = {
    ...
    const email = await upsServiceClient.getProfileEmail();
                          // Get user email
    const givenName = await upsServiceClient.getProfile
     GivenName();          // Get user name
    ...
    const speechText = 'Enjoy your trip, ${givenName}!';
     // Only name is used, thus here is an over-privilege
     issue in the skill.
    return handlerInput.responseBuilder.speak(speechText).
     reprompt(speechText).getResponse();
};

//manifest file:
"permissions": [
  {"name": "alexa::profile:given_name:read"},
  {"name": "alexa::profile:email:read"
],
"privacyAndCompliance": {
  "locales": {
    "en-US": {}      // Here should be a privacy policy link
     but it is missing
  }},
```

Listing 2: An example skill that requests the email and user name but doesn't fully use the requested data and lacks a privacy policy.

For the 197 skills containing slots for sensitive data collection and 133 skills asking for at least one permission, we conducted a taint analysis of these sensitive data sources. After manually checking the results, SkillScanner achieves an accuracy of 95% for slot data flow analysis and 89% for permission data flow analysis. The failure of data flow analysis often results from overlooked sinks that SkillScanner doesn't consider, such as using a user's name to make an appointment or using their location to find the nearest restaurant. For the 187 skills (after removing false positives from 197 skills) that contain slots for sensitive data collection, we found that 8 skills don't use the data, 121 skills use the collected data in their outputs/responses and 91 skills save data to databases (one skill may contain multiple data usages). For the 118 skills asking for permission data, 42 skills don't use at least one permission data, which are over-privilege skills. 57 skills store data in databases and 30 skills use data for outputs or responses. Listing 2 shows an example that doesn't fully use the data it asks for. The skill requests the email and user name information but only the name is used in the skill response, and thus it is flagged as an over-privileged skill. SkillScanner can effectively detect such issues and warn developers to fix them during the development phase.

6.4 Privacy Violation Detection

When skills collect data from users, they should also provide complete and informative privacy policies for disclosing such data collection activities. Amazon Alexa also asks developers "Does this Alexa skill collect users' personal information?" during the skill submission phrase. However, we observed that many skills failed in disclosing their data practices in their privacy policy documents.

	# of skills	# of skills missing a privacy policy	# of skills having an incomplete privacy policy
Data collection request	321	240 (75%)	19 (6%)
Ask for permission	133	94 (71%)	23 (17%)
Store slot value in database	89	81 (91%)	4 (4%)
Store permission value in database	57	45 (79%)	6 (11%)

Table 7: Number of skills with privacy policy issues.

6.4.1 *Problematic privacy policy.*

A privacy policy should clearly describe how user data is collected, used and shared. However, after checking the privacy policies of all data collection skills, we found that only 19% of the skills provide a complete privacy policy. Table 7 shows the breakdown of privacy policy issues in different types of data collection. For the 321 skills with data collection requests, most of them have inconsistency issues between their data collection request and privacy policies. 75% of skills don't provide a privacy policy and 6% of skills provide an incomplete privacy policy that doesn't claim their actual data collection behavior. For the 133 skills asking for data with permissions, 71% of them lack a privacy policy and 17% of them have an incomplete privacy policy.

When it comes to how data is stored in databases, few skills mention that in privacy policies. For the 89 skills that store data collection slot value, 91% of them lack a privacy policy and only 4 skills (4%) mention the data would be stored. For the 57 skills that ask for permissions and store data in a database, only 21% provide a privacy policy and 10% all are useful. Other 6 skills provide broken or unrelated websites while one skill even says that it "will not retain data", which is deceptive.

Since the privacy policy link is extracted from the skill manifest file, a skill lacks a privacy policy if we couldn't find it. So for skills missing a privacy policy, SkillScanner achieves 100% accuracy. For skills providing an incomplete privacy policy, we manually checked their privacy policy content. There are 3 false positive cases (already removed in Table 7) for data collection skills and SkillScanner achieves an accuracy of 93%. The recall for detecting problematic privacy policy is 86% and it's partially because of the false negatives in detecting data collection requests (details in Section 6.2) or failure to read the manifest file to obtain the privacy policy link.

6.4.2 *Over-privileged data requests.*

The requested permission information can be extracted from the skill manifest file. To detect over-privileged permission data requests, we checked whether these permissions are used in the back-end code or not. There are 42 skills that request for permissions but don't use them. There are also 18 skills that get permission data without requesting for permissions.

6.4.3 *Inconsistent data collection disclosure to Alexa platform.*

Amazon asks developers whether their skills collect user data before submission and developers can select yes or no by themselves. The answers will be stored in the skill manifest file if a developer selects it. There are 126 skills with data collection in our dataset making a selection, in which 99 skills claim not using personal info while only 27 developers correctly select they use personal info. However, for the 27 skills, 4 skills don't provide a privacy policy and 16 provide an incomplete one.

6.5 Content Guideline Violation Detection

6.5.1 Content safety violations. After checking toxic content with the Perspective tool in skill outputs, we found one skill named "Wired life hacks" that outputs "Anything else you would like to know, mother f*ckers?" and "you f*cked up! you f*cked up! you f*cked up!". Such outputs can be harmful to users. Next we checked whether any skill asks for a positive rating. One skill named "Word Cyclopedia" says "If you like this skill, please give this skill five star and write your valuable feedback" in the description.

Skills may get functional content from a website or provide users with an audio/image as output. After extracting and processing media files separately, we got 1,656 websites, 1,439 mp3 files and 932 images from skill code. For websites, we checked whether they are malicious websites and whether the content of a website has toxic content. We didn't find any such case in our dataset. For the extracted audio and image files, we didn't find any violation either. Detecting content safety from the dynamic and changing external websites (where skills obtain data) can be challenging. However, upon revisiting and comparing the content of all websites used in skills in a six-month period, we found that the majority of these websites (85%) didn't change their content.

6.5.2 Non-compliance of invocation names. We found 281 skills with different types of invocation name violations. Among these skills, 53% of skills have an invocation name with only one word, such as "greeter", "eva" or "jokes"; 4% of them have two-word invocation names with an article or preposition, like "the car" or "the helper"; 35% of skills used an invocation name that contains Alexa-related words, such as "play radio", and the other 8% skills don't use invocation name in lower case like "AlTranslate" or "Ryan's note". Example invocation names with violations are provided in Table 5. SkillScanner successfully detected invocation name violations with an accuracy of 96% and recall of 99%. SkillScanner incorrectly reported skills with invocation names that are one-word brand names as violations. The false negatives are mainly due to the failure to read the skill file with errors.

Amazon Alexa also requires that "the invocation name should be distinctive to ensure users can enable your skill". Such violations of the invocation name can negatively influence user experience or be used for squatting attack [40]. For example, one skill's invocation name is "weather app". When users try to invoke the "weather" skill with "Alexa, open the weather app", this skill will be invoked instead of the "weather" skill. For users who want to invoke a skill with commonly used invocation name, he/she first needs to find and enable that skill, so that it can be correctly invoked from multiple candidate skills. However, we found such a violation is very common in both our dataset and Amazon Alexa skills store. So, we decided not to include it in our results. We found that 202 invocation names used by 990 skills (40%) in our dataset, and 16,020 published skills have this violation in Alexa's skills store.

6.5.3 Category-specific violations. For policy violations related to different categories, we found one skill asks for user data in the Kids category. 13 skills in the Health category lack a disclaimer in their description. For example, one skill named "UCSD Health" in the "Health" category provides a description "Ask UCSD Health to save your spot at the nearest clinic" and it doesn't contain a disclaimer which is required by Alexa.

6.6 Code Inconsistency Detection

6.6.1 A skill with a data collection request but without a slot for processing user reply. When a skill asks for user data, it should define an intent and slot for capturing and processing such data, such as "NameIntent" with slot "name". However, we found that 124 skills ask for user data but don't have corresponding intent and slot. For example, the skill "awsFact" tells user "You can introduce yourself by telling me your name", but it only contains four Amazon built-in intents. For such a skill without proper intent and slot, if users provide data following the instruction, the skill will invoke a wrong intent or trigger built-in "FallbackIntent" with the reply "Sorry, I don't know that. Please try again". Such an incorrect reply will no doubt influence user experience. We found that over 50 skills use the same sentence ("You can introduce yourself by telling me your name") in the "HelpIntent" but don't have an intent or slot for processing the data.

6.6.2 A skill with a data collection slot but without data collection requests to trigger the slot. We found 147 skills that define a slot for data collection, but they don't have any data collection requests. For example, a skill named "theStudyBar" has an intent named "myNameIs". The slot defined in this intent is "firstName" and the sample utterances are "{firstName}" and "my name is {firstName}". When we checked the skill back-end code, there didn't exist a request asking for the user name. This makes the data collection activity hidden when the skill is submitted during the certification phase. However, developers can update the code after the skill is published, then the skill can collect user data again.

6.6.3 Data collection slot and intent without sample utterances. We found 24 skills with 95 sensitive data collection slots and 40 skills with 87 intents (not for data collection) don't have any corresponding sample utterances. This can be a bug or error because the interaction model doesn't know how to match a user's reply with the intent and slot. When a user provides a reply, no intent or a wrong intent will be invoked and the slot value will never be captured. Such cases are similar to missing an intent because although the intent is defined, it will not be triggered.

For code inconsistency detection, we also manually checked all our results. For skills with data collection requests but without a slot, the accuracy is 91%. For skills with data collection slots but without data collection requests, the accuracy is 84% because some data collection outputs cannot be correctly recognized due to the complexity of NLP. For detecting data collection slots and intents without sample utterances, SkillScanner achieves 100% accuracy. Since the inconsistency detection is reliant on the results from sensitive data collection activity detection, the reasons for false negatives are the same with Section 6.2 and the recall rates for the three parts are 82%, 92%, 93%, respectively.

6.7 Impact of Code Duplication

6.7.1 Impact of code snippet Copy&Paste. Although we have removed the duplicate skills before the analysis, we still found many different skills with the same policy violations. For example, 67 skills output "You can introduce yourself by telling me your name" in their HelpIntent, which is designed to provide helpful information for users. However, they don't provide a privacy policy while collecting user data. Since the default sentence in HelpIntent is "You can say hello to me! How can I help?", it is possible that these skills didn't use Alexa's default template but copied the code snippet from other places for their functions.

To find such cases in skill violations, we used the copydetect [9] tool to detect the similarity between the source code files where skill violations appeared. Then, we grouped similar skills with the same violations and checked whether there exists any template skill, especially skill tutorials or templates, which were potentially copied by other skills in the same group. As a result, we found there exist 453 violations (38%) because of copying code from other skills. Such cases were more serious in two types of violations: "sensitive data collection in output but missing a privacy policy" (53% of the skills with such issue are due to the code snippet copy&paste) and "sensitive data collection in output but without a slot for processing it" (67% of the skills that violate this policy are due to the code snippet copy&paste). As for the source of copied skills, we found the most commonly copied skills were from the GitHub accounts "dabblelab" and "alexa-samples", which lead to violations in another 87 and 65 skills, respectively. Note that "alexa-samples" seems to be an Alexa official account and it contains 109 skill tutorials. More details about Alexa's official accounts will be discussed in the next section.

6.7.2 Code snippet Copy&Paste in potential Alexa's official skills. After tracing the sources of copied skills, we found that the GitHub accounts "alexa-samples" , "alexa", "alexa-dev-hub", "alexa-labs" and "aws-samples" (which are potential Amazon Alexa's accounts on GitHub), published skills that were often copied by other skills and they lead to violations in tens of skills. Although these Alexa-provided skills aim to provide tutorials for developers to learn how to develop skills, a few issues in these skills could lead to violations in other skills and provide bad examples for third-party developers. To understand whether these official skills' code has violations, we checked all their skills and found these accounts don't provide a good example for others. The account "alexa-samples" published 125 skills, in which 20 skills are supposed to provide a privacy policy but only 1 skill provides the privacy policy. This account influenced 62 other skills. What's more, instead of leaving a blank for the privacy policy link, as shown in Listing 2, some official skills don't even have the "privacyAndCompliance" attribute in the manifest file, which makes developers not notice there should be a privacy policy link. Another issue is that the tutorial skills are in different standards regarding policy compliance. For a template named "zero to hero", which includes 10 different versions of skill templates, only the tenth skill provides a privacy policy while the others do not, although all ten skills ask for the same user data. In total, we found 42 official skills with violations and they influenced 71 other skills with 81 policy violations.

6.8 Performance and Responsible Disclosure

6.8.1 Performance. Table 8 summarizes the performance of each component in SkillScanner and Table 9 summarizes our detection results. For each component, we manually checked all the detected results to get the accuracy of SkillScanner. In addition, we also randomly selected 100 skills from our dataset and manually labeled them to evaluate the overall performance of SkillScanner. The results show that SkillScanner achieves overall 90% precision and 87% recall.

Analysis	SkillScanner Component	Accuracy	Recall
Data Collection Activity	Sensitive data collection request detection	98%	84%
	Sensitive data collection slot detection	99.7%	93%
	Sensitive data collection permission detection	100%	91%
Data Flow Analysis	Slot data flow analysis	95%	-
	Permission data flow analysis	89%	-
Privacy Violation	Missing a privacy policy	100%	86%
	Having an incomplete privacy policy	93%	86%
Content	Invocation name violation	96%	99%
Code Inconsistency	Data collection request but missing a slot	91%	82%
	Data collection slot but missing a request	84%	92%
	Data collection slot but missing an utterance	100%	93%
	Intent but missing an utterance	100%	93%
Overall performance of SkillScanner		90%	87%

Table 8: Performance of different components in SkillScanner.

6.8.2 Latency. For each skill, SkillScanner needs only 75s on average for scanning all source files, performing the policy violation detection, and reporting analysis results. Figure 5 shows the Cumulative Distribution Fraction (CDF) of SkillScanner's latency performance. Over 85% of skills need less than two minutes for a comprehensive detection. Compared to SkillDetective [59], which uses a dynamic testing method that needs to wait for skill responses and takes around 10 minutes to test one skill, SkillScanner has a significantly lower time cost. The majority of time consumed by SkillScanner is on toxic content detection (sending outputs to Perspective) and downloading audio, images, and websites.

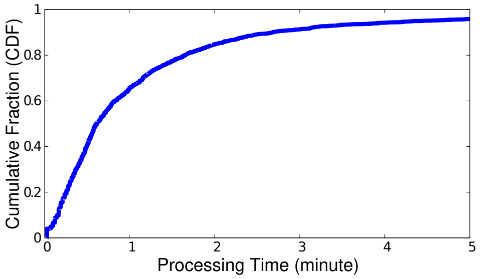


Figure 5: Latency performance of SkillScanner.

6.8.3 Responsible disclosure. We performed a responsible disclosure process with several steps. 1) We reported the identified issues in Alexa's sample code to the Amazon Alexa team. 2) We notified 83 developers who provided email addresses on their GitHub pages about the potential policy violations in their code. 3) We created new issues on the GitHub repositories to notify the developers who had multiple problematic skills but didn't provide an email address. It is worth mentioning that we received feedback from 6 developers (among the 83 developers we reached out to). They expressed appreciation for our work and acknowledged that such a static analysis tool is helpful in facilitating their development of policy-compliant skills. They also mentioned the reasons for the policy violations, such as being unaware of policies, using a template that does not have a privacy policy or developing skills for learning purposes. Three respondents mentioned that they were impressed by our SkillScanner tool. Two respondents planned to fix the issues and one had already fixed the issue. Two respondents explained that the skills were just developed for learning purposes.

7 DISCUSSION

7.1 Potential Impact of Problematic Skills

Problematic skills could have a negative impact on users, VPA platforms, and developers. 1) *End-users are concerned about policy violations in skills.* We collected a user review dataset from the skills store and we did find negative reviews talking about the policy violations in skills that impacted their experience. 2,383 users complained about the data collection and 107 users mentioned that skills repeatedly asked for a positive rating in reviews. 2) *VPA platforms are concerned about policy violations in skills.* The Amazon Alexa platform enforces a certification process to ensure that each skill meets the required content and privacy requirements before it becomes publicly available. According to recent works [35, 36, 59], after researchers reported the policy-violating skills in the skills store to the Amazon Alexa team, VPA platforms would remove these skills from the store. Out of 93 skills in our dataset published in the Alexa skills store, SkillScanner detected 14 problematic skills that either missed a privacy policy or had an invalid invocation name. Interestingly, the policy violations in these skills were fixed in their published versions. It is possible they failed to pass Amazon's certification process with policy violations and then the developers had to resolve the issues for publishing them. 3) *Developers are concerned about policy violations in skills.* Since policy-violating skills may be rejected at the certification phase and developers need to fix issues in skills and wait several days to get feedback from the certification team, policy violations in skills inevitably slow down the skill development and publishing.

7.2 Limitations

Despite the limited number of skill packages from GitHub, SkillScanner is able to identify 1,328 violations in 786 policy-noncompliance skills. 4 policy-violating skills have been published on the skills store without fixing them (2 skills have an incomplete privacy policy and 2 skills in the Health category lack a disclaimer). However, SkillScanner has several limitations. First, for the data flow analysis, more types of taint sources and sinks can be considered. For example, a skill may collect users' email addresses and send emails to users in the back-end code. Account linking [15] can be another way of user data leakage and we will consider that in our future research. Second, the current design of SkillScanner does not cover all Alexa's privacy requirements and content guidelines. In addition, the policy violation detection performance can be further improved by using advanced machine learning techniques. However, it is non-trivial to collect high-quality datasets for model training. Third, SkillScanner only focuses on the Amazon Alexa platform, which is the most popular VPA platform. We plan to extend SkillScanner to analyze Google Actions as our future work. Forth, most of the skills in our dataset were not published in the skills store and they need to pass a certification process before being published. As a result, the skills in our dataset may have more issues than the skills in the store. However, we don't claim the identified issues in our results are representative of actual skills in production. Rather, the main purpose of our evaluation in Section 6 is to demonstrate the efficiency of SkillScanner in finding policy violations given skill code. Since SkillScanner is designed for facilitating third-party developers in developing policy-compliant skills, it is important to evaluate the developer acceptance rate of SkillScanner. We also plan to conduct user studies to get feedback from skill developers about the usability of SkillScanner.

8 RELATED WORK

There has been an increasing number of studies on VPA security and privacy [33, 37, 57, 58]. The majority of research efforts have been undertaken to identify various acoustic-based attacks (*e.g.*, out-of-band signal attacks and adversarial example attacks) against the Automatic Speech Recognition (ASR) modules in VPA systems [25, 29, 30, 58] and the corresponding defenses in mitigating these attacks [26, 31, 47, 61]. As hundreds of thousands of skills have been published in VPA platforms, the security of skills has attracted attention from the research community. Since attacks that exploit skills' vulnerabilities can be launched remotely, they could potentially be more powerful than acoustic-based attacks [38, 56, 63]. In this section, we briefly discuss recent research on identifying problematic (*e.g.*, privacy-invasive) skills in Amazon Alexa and Google Assistant platforms.

Kumar *et al.* [40] presented the skill squatting attack and they found 381 pairs of skills were likely to be squatted. Lentzsch *et al.* [41] discovered 262 skills with permissions provide an incomplete privacy policy. In [36], the authors reported 675 skills that request permission data have privacy issues about privacy policies. SkillVet [35] presented a machine learning based method for checking data practices and privacy issues. SkillExplorer [39] tested 28,904 Amazon skills and identified 1,141 skills requesting users to provide personal information without disclosing in their privacy policies. VerHealth [53] analyzed 813 health-related skills on the Amazon Alexa platform. Vitas [42] interacted with 41,581 skills and found that 51% of skills suffered from problems such as unexpected exit/start and privacy violation. SkillDetective [59] identified 6,079 policy-violating skills in the current skills stores by evaluating skills' conformity to more than 50 different policy requirements. Many research efforts have been undertaken to study user concerns (human factors) about the security and privacy of VPA devices [24, 27, 34, 64]. Malkin *et al.* [45] conducted a user study about how people react to the runtime permission. Sharma *et al.* [51] focused on Google Assistant and showed that most participants have superficial knowledge about the data collected by the platform. Liu *et al.* [44] studied 214 participants about whether they consider privacy permissions while installing apps. Sabir *et al.* [49] did a user study to analyze users' awareness of third-party skills. However, none of them conducted user study to understand skill developers' perceptions and practices regarding policy compliance in skill development.

Distinction from existing works. SkillScanner distincts itself from existing policy violation detection works (including SkillExplorer [39], VerHealth [53], Vitas [42] and SkillDetective [59]) in three ways. First, different from recent skill testing tools that utilize dynamic analysis and can only detect violations exercised by run-time inputs, SkillScanner is the first-of-its-kind static analysis tool and it detects more possible violations/inconsistencies in the code. Second, SkillScanner identifies policy violations of skills developed by inexperienced benign developers in the development phase while existing works detect policy violations in the post-deployment phase. Third, SkillScanner detects more types of violations existing in the skill code (such as data storage, over-privileged data, and code inconsistency) that existing works can't find. For example, compared to SkillDetective, SkillScanner detects 12 new types of violations. Even for the 8 types of violations that SkillDetective can find, such as skill output and content safety, SkillScanner can find more potential outputs that might not appear in dynamic testing. As a result, 934/1328 (70%) of violations are new and can't be found by SkillDetective.

9 CONCLUSION

In this work, we first conducted a user study to understand the gap between VPA's policy requirements and skill developers' practices. Informed by our user study results, we designed and implemented the first static analysis tool named SkillScanner, which helps developers automatically identify potential policy violations in skills at the development phase. To evaluate the performance of SkillScanner, we collected 2,451 open source skills from GitHub, and conducted a comprehensive analysis of these skills using SkillScanner. SkillScanner effectively identified 1,328 violations among 786 skills. 694 of them are about privacy non-compliance and 298 of them violate content guidelines imposed by the Amazon Alexa platform. We found that 32% of policy violations are introduced through code copy and paste. The policy violations in Alexa's official accounts led to 81 policy violations in other skills. As our future work, we plan to conduct user studies to evaluate the usability (*e.g.*, acceptance and user-friendliness) of SkillScanner by skill developers.

ACKNOWLEDGMENT

The work of L. Cheng is supported by National Science Foundation (NSF) under the Grant No. 2239605, 2228616 and 2114920. The work of H. Hu is supported by NSF under the Grant No. 2228617, 2120369, 2129164, and 2114982. The work of L. Guo is supported by NSF under grant IIS-1949640, CNS-2008049, and CCF-2312616.

REFERENCES

[1] Alexa Certification Tests for VUI and UX. https://developer.amazon.com/fr-FR/docs/alexa/custom-skills/voice-interface-and-user-experience-testing-for-a-custom-skill.html.

[2] Alexa Permissions. https://developer.amazon.com/en-US/docs/alexa/custom-skills/configure-permissions-for-customer-information-in-your-skill.html .

[3] Alexa Policy for Child-Directed skill. https://developer.amazon.com/fr-FR/docs/alexa/custom-skills/policy-testing-for-an-alexa-skill.html#cert-child-directed.

[4] Alexa Policy for Health skill. https://developer.amazon.com/fr-FR/docs/alexa/custom-skills/policy-testing-for-an-alexa-skill.html#3-health.

[5] Alexa Skill Counts Surpass 80K in US. https://voicebot.ai/2021/01/14/alexa-skill-counts-surpass-80k-in-us-spain-adds-the-most-skills-new-skill-introduction-rate-continues-to-fall-across-countries.

[6] Alexa Skills Policy Testing. https://developer.amazon.com/fr/docs/custom-skills/policy-testing-for-an-alexa-skill.html.

[7] Alexa Skills Privacy Requirements. https://developer.amazon.com/fr/docs/custom-skills/security-testing-for-an-alexa-skill.html#25-privacy-requirements.

[8] CodeQL. https://codeql.github.com/.

[9] copydetect. https://github.com/blingenf/copydetect .

[10] First Alexa Third-Party Skills Now Available for Amazon Echo. https://developer.amazon.com/es/blogs/alexa/post/TxC2VHKFEIZ9SG/first-alexa-third-party-skills-now-available-for-amazon-echo .

[11] Perspective. https://www.perspectiveapi.com/home.

[12] pytesseract 0.3.7. https://pypi.org/project/pytesseract/.

[13] Qualitrics. https://www.qualtrics.com/ .

[14] Sensitive data types: Personal information. https://docs.aws.amazon.com/macie/latest/user/managed-data-identifiers.html#managed-data-identifiers-pii.

[15] Skill Account Linking Schemas. https://developer.amazon.com/en-US/docs/alexa/smapi/account-linking-schemas.html .

[16] Skill Interaction Model Schemas. https://developer.amazon.com/en-US/docs/alexa/smapi/interaction-model-schema.html .

[17] Skill Manifest Schema. https://developer.amazon.com/en-US/docs/alexa/smapi/skill-manifest.html.

[18] Skill Package Format. https://developer.amazon.com/en-US/docs/alexa/hosted-skills/alexa-hosted-skills-git-import.html.

[19] Skill Slot Type Reference. https://developer.amazon.com/en-US/docs/alexa/custom-skills/slot-type-reference.html .

[20] Skill Standard Built-in Intents. https://developer.amazon.com/en-US/docs/alexa/custom-skills/standard-built-in-intents.html .

[21] spaCy. https://spacy.io.

[22] SpeechRecognition 3.8.1. https://pypi.org/project/SpeechRecognition/.

[23] VirusTotal. https://www.virustotal.com/gui/home/search.

[24] Noura Abdi, Kopo M Ramokapane, and Jose M Such. More than smart speakers: Security and privacy perceptions of smart home personal assistants. In *SOUPS@ USENIX Security Symposium*, 2019.

[25] Hadi Abdullah, Kevin Warren, Vincent Bindschaedler, Nicolas Papernot, and Patrick Traynor. Sok: The faults in our asrs: An overview of attacks against automatic speech recognition and speaker identification systems. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 730–747, 2021.

[26] Shimaa Ahmed, Ilia Shumailov, Nicolas Papernot, and Kassem Fawaz. Towards more robust keyword spotting for voice assistants. In *31st USENIX Security Symposium (USENIX Security 22)*, 2022.

[27] Tawfiq Ammari, Jofish Kaye, Janice Y. Tsai, and Frank Bentley. Music, search, and iot: How people (really) use voice assistants. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 26(3):1–28, 2019.

[28] Travis D Breaux, Hanan Hibshi, and Ashwini Rao. Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements. *Requirements Engineering*, 19(3):281–307, 2014.

[29] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *USENIX Security Symposium (USENIX Security)*, pages 513–530, 2016.

[30] Guangke Chen, Sen Chen, Lingling Fan, Xiaoning Du, Zhe Zhao, Fu Song, and Yang Liu. Who is real bob? adversarial attacks on speaker recognition systems. In *IEEE Symposium on Security and Privacy (SP)*, 2021.

[31] Yanjiao Chen, Yijie Bai, Richard Mitev, Kaibo Wang, Ahmad-Reza Sadeghi, and Wenyuan Xu. Fakewake: Understanding and mitigating fake wake-up words of voice assistants. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1861–1883, 2021.

[32] Long Cheng, Christin Wilson, Song Liao, Jeffrey Young, Daniel Dong, and Hongxin Hu. Dangerous skills got certified: Measuring the trustworthiness of skill certification in voice personal assistant platforms. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2020.

[33] Peng Cheng and Utz Roedig. Personal voice assistant security and privacy—a survey. *Proceedings of the IEEE*, 110(4):476–507, 2022.

[34] H. Chung, M. Iorga, J. Voas, and S. Lee. "alexa, can i trust you?". *IEEE Computer*, 50(9):100–104, 2017.

[35] Jide Edu, Xavi Ferrer Aran, Jose Such, and Guillermo Suarez-Tangil. Skillvet: Automated traceability analysis of amazon alexa skills. *IEEE Transactions on Dependable and Secure Computing*, 2021.

[36] Jide Edu, Xavier Ferrer-Aran, Jose Such, and Guillermo Suarez-Tangil. Measuring alexa skill privacy practices across three years. In *Proceedings of the ACM Web Conference (WWW)*, page 670–680, 2022.

[37] Jide S. Edu, Jose M. Such, and Guillermo Suarez-Tangil. Smart home personal assistants: A security and privacy review. *ACM Computing Surveys*, 53(6), 2020.

[38] Sergio Esposito, Daniele Sgandurra, and Giampaolo Bella. Alexa versus alexa: Controlling smart speakers by self-issuing voice commands. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 1064–1078, 2022.

[39] Zhixiu Guo, Zijin Lin, Pan Li, and Kai Chen. Skillexplorer: Understanding the behavior of skills in large scale. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 2649–2666, 2020.

[40] Deepak Kumar, Riccardo Paccagnella, Paul Murley, Eric Hennenfent, Joshua Mason, Adam Bates, and Michael Bailey. Skill Squatting Attacks on Amazon

	Problem	Detailed policy	Data source	# of skills with policy violation
Privacy Violations	Data collection/storage but missing a privacy policy	Provide a legally adequate privacy notice that will be displayed to end users on your skill's detail page.	Output	240
			Permission	94
			Database	126
			Disclosure to Alexa	4
	Data collection/storage but having an incomplete privacy policy	Ensure that your collection and use of that information complies with your privacy notice and all applicable laws.	Output	19
			Permission	23
			Database	10
			Disclosure to Alexa	19
	Over-privileged data requests	Collect and use the data only if it is required to support and improve the features and services your skill provides.	Permission	42
	Not-asked permission usage	Bug	Permission	18
	Incorrect data collection disclosure to Alexa	Wrong answer to "Does this Alexa skill collect users' personal information?"	Disclosure to Alexa	99
Violations of Content Guidelines	Content safety	Contains excessive profanity.	Output	1
	Asking for positive rating	Can not explicitly requests that users leave a positive rating of the skill.	Description	1
	Invocation name requirements	Does not adhere to Amazon Invocation Name Requirements. Please consult these requirements to ensure your skill is compliant with our invocation name policies.	Invocation name	281
	Kid category policy	It doesn't collect any personal information from end users.	Output	1
	Health category policy	A skill that provides health-related information, news, facts or tips must include a disclaimer in the skill description stating that the skill is not a substitute for professional medical advice.	Description	13
Code Inconsistency	Data collection request but missing a slot	Bug	Output	124
	Data collection slot but missing a request	Vulnerability	Intent	147
	Data collection slot but missing an utterance	Bug	Slot	24
	Intent but missing an utterance	Bug	Intent	40

Table 9: Policy violations and code inconsistency in skill code and detailed policies they violate. We have removed the false positives by our manual verification.

Alexa. In *27th USENIX Security Symposium (USENIX Security)*, pages 33–47, 2018.

[41] Christopher Lentzsch, Sheel Jayesh Shah, Benjamin Andow, Martin Degeling, Anupam Das, and William Enck. Hey Alexa, is this skill safe?: Taking a closer look at the Alexa skill ecosystem. In *Proceedings of the 28th ISOC Annual Network and Distributed Systems Symposium (NDSS)*, 2021.

[42] Suwan Li, Lei Bu, Guangdong Bai, Zhixiu Guo, Kai Chen, and Hanlin Wei. Vitas: Guided model-based vui testing of vpa apps. In *37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–12, 2022.

[43] Song Liao, Christin Wilson, Long Cheng, Hongxin Hu, and Huixing Deng. Measuring the effectiveness of privacy policies for voice assistant applications. In *Annual Computer Security Applications Conference (ACSAC)*, page 856–869, 2020.

[44] Gary Liu and Nathan Malkin. Effects of privacy permissions on user choices in voice assistant app stores. *Proceedings on Privacy Enhancing Technologies*, 4:421–439, 2022.

[45] Nathan Malkin, David Wagner, and Serge Egelman. Runtime permissions for privacy in proactive intelligent assistants. In *Eighteenth Symposium on Usable Privacy and Security (SOUPS 2022)*, pages 633–651, 2022.

[46] Erika McCallister. *Guide to protecting the confidentiality of personally identifiable information*, volume 800. Diane Publishing, 2010.

[47] Yan Meng, Jiachun Li, Matthew Pillari, Arjun Deopujari, Liam Brennan, Hafsah Shamsie, Haojin Zhu, and Yuan Tian. Your microphone array retains your identity: A robust voice liveness detection system for smart speaker. In *USENIX Security*, 2022.

[48] Lisa Parker, Tanya Karliychuk, Donna Gillies, Barbara Mintzes, Melissa Raven, and Quinn Grundy. A health app developer's guide to law and policy: a multi-sector policy analysis. *BMC Medical Informatics and Decision Making*, 2017.

[49] Aafaq Sabir, Evan Lafontaine, and Anupam Das. Hey alexa, who am i talking to?: Analyzing users' perception and awareness regarding third-party alexa skills. In *CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2022.

[50] Awanthika Senarath and Nalin A. G. Arachchilage. Why developers cannot embed privacy into software systems? an empirical investigation. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, page 211–216, 2018.

[51] Vandit Sharma and Mainack Mondal. Understanding and improving usability of data dashboards for simplified privacy control of voice assistant data. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3379–3395, 2022.

[52] Swapneel Sheth, Gail Kaiser, and Walid Maalej. Us and them: A study of privacy requirements across north america, asia, and europe. In *Proceedings of the 36th International Conference on Software Engineering (ICSE)*, page 859–870, 2014.

[53] Faysal Hossain Shezan, Hang Hu, Gang Wang, and Yuan Tian. Verhealth: Vetting medical voice applications through policy enforcement. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2020.

[54] Dan Su, Jiqiang Liu, Sencun Zhu, Xiaoyang Wang, and Wei Wang. " are you home alone?"" yes" disclosing security and privacy vulnerabilities in alexa skills. *arXiv preprint arXiv:2010.10788*, 2020.

[55] Dawei Wang, Kai Chen, and Wei Wang. Demystifying the vetting process of voice-controlled skills on markets. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(3), 2021.

[56] Yuanda Wang, Hanqing Guo, and Qiben Yan. Ghosttalk: Interactive attack on smartphone voice system through power line. *arXiv preprint arXiv:2202.02585*, 2022.

[57] Chen Yan, Xiaoyu Ji, Kai Wang, Qinhong Jiang, Zizhi Jin, and Wenyuan Xu. A survey on voice assistant security: Attacks and countermeasures. *ACM Computing Surveys*, 2022.

[58] Qiben Yan, Kehai Liu, Qin Zhou, Hanqing Guo, and Ning Zhang. Surfingattack: Interactive hidden attack on voice assistants using ultrasonic guided wave. In *Network and Distributed Systems Security (NDSS) Symposium*, 2020.

[59] Jeffrey Young, Song Liao, Long Cheng, Hongxin Hu, and Huixing Deng. SkillDetective: Automated policy-violation detection of voice assistant applications in the wild. In *USENIX Security Symposium (USENIX Security)*, 2022.

[60] Le Yu, Xiapu Luo, Xule Liu, and Tao Zhang. Can we trust the privacy policies of android apps? In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 538–549. IEEE, 2016.

[61] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack: Inaudible voice commands. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 103–117, 2017.

[62] Nan Zhang, Xianghang Mi, Xuan Feng, XiaoFeng Wang, Yuan Tian, and Feng Qian. Dangerous skills: Understanding and mitigating security risks of voice-controlled third-party functions on virtual personal assistant systems. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1381–1396. IEEE, 2019.

[63] Yangyong Zhang, Lei Xu, Abner Mendoza, Guangliang Yang, Phakpoom Chinprutthiwong, and Guofei Gu. Life after speech recognition: Fuzzing semantic misinterpretation for voice assistant applications. In *Network and Distributed System Security Symposium (NDSS)*, 2019.

[64] Serena Zheng, Noah Apthorpe, Marshini Chetty, and Nick Feamster. User perceptions of smart home iot privacy. *Proc. ACM Hum.-Comput. Interact.*, 2018.