Demystifying the Vetting Process of Voice-controlled Skills on Markets

DAWEI WANG, SKLOIS, Institute of Information Engineering, Chinese Academy of Sciences and School of Cyber Security, University of Chinese Academy of Sciences, China
KAI CHEN*, SKLOIS, Institute of Information Engineering, Chinese Academy of Sciences, School of Cyber Security, University of Chinese Academy of Sciences, and Beijing Academy of Artificial Intelligence, China
WEI WANG*, Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, China

Smart speakers, such as Google Home and Amazon Echo, have become popular. They execute user voice commands via their built-in functionalities together with various third-party voice-controlled applications, called skills. Malicious skills have brought significant threats to users in terms of security and privacy. As a countermeasure, only skills passing the strict vetting process can be released onto markets. However, malicious skills have been reported to exist on markets, indicating that the vetting process can be bypassed. This paper aims to demystify the vetting process of skills on main markets to discover weaknesses and protect markets better. To probe the vetting process, we carefully design numerous skills, perform the Turing test, a test for machine intelligence, to determine whether humans or machines perform vetting, and leverage natural language processing techniques to analyze their behaviors. Based on our comprehensive experiments, we gain a good understanding of the vetting process (e.g., machine or human testers and skill exploration strategies) and discover some weaknesses. In this paper, we design three types of attacks to verify our results and prove an attacker can embed sensitive behaviors in skills and bypass the strict vetting process. Accordingly, we also propose countermeasures to these attacks and weaknesses.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing**; Ubiquitous and mobile computing design and evaluation methods.

Additional Key Words and Phrases: Alexa, Google-Home, Skill, vetting process

ACM Reference Format:
Dawei Wang, Kai Chen, and Wei Wang. 2021. Demystifying the Vetting Process of Voice-controlled Skills on Markets. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 3, Article 130 (September 2021), 28 pages. https://doi.org/10.1145/3478101

1 INTRODUCTION

Smart speakers such as Google Home and Amazon Echo have become extremely popular. According to a recent report [25], 163 million smart speakers will be installed worldwide in 2021. Smart speakers receive voice commands

*Corresponding authors.

Authors' addresses: Dawei Wang, SKLOIS, Institute of Information Engineering, Chinese Academy of Sciences and School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China, 100044, wangdawei@iie.ac.cn; Kai Chen, SKLOIS, Institute of Information Engineering, Chinese Academy of Sciences, School of Cyber Security, University of Chinese Academy of Sciences, and Beijing Academy of Artificial Intelligence, Beijing, China, 100044, chenkai@iie.ac.cn; Wei Wang, Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, 3 Shangyuancun, Beijing, China, 100044, wangwei1@bjtu.edu.cn.

2474-9567/2021/9-ART130 $15.00
https://doi.org/10.1145/3478101

from users and perform corresponding tasks, such as playing music, querying information (e.g., weather and stocks), online shopping (e.g., ordering pizza), and controlling smart devices in homes. In addition to the limited built-in functionalities, providers (e.g., Google and Amazon) support third-party developers to create extra capabilities (i.e., *skills* by Amazon and *actions* by Google[1]). Like installing apps on smartphones to acquire more functions, users can enable skills on smart speakers to do more things, including publishing records on their social networks, controlling their Internet of Things devices, and unlocking their cars. By the end of 2020, over 80,000 Alexa skills [49] and 18,826 Google actions [48] have been released.

While most skills support users with pleasant functionalities, some potentially dangerous skills (PDS) are harmful. For example, PDS may collect users' private information, similar to malicious applications on smartphones. Previously, studies demonstrated that it is feasible to create a malicious skill with the same or similar pronunciation of the names with an existing skill [50, 91]. For example, an attacker can create a malicious skill, called "Cat Fats," which sounds similar to the skill "Cat Facts." In this manner, the users' requests could be diverted to the malicious skill, even unnoticeable by users, together with their private information. In addition to using confusing names, PDS can eavesdrop on users' conversations by carefully designing a question with unpronounceable characters. Even if such a skill is stopped through voice commands, it can still keep eavesdropping on users' conversation by reading unpronounceable questions [15].

To protect users from malicious skills, providers build vetting systems to analyze skills and remove those violating the market requirements. For example, Amazon requires each skill to pass through a certification process, including validation, functional tests, and review[11]. Ideally, markets should verify both names and skill behaviors to vet skills before making them available to users. However, recent reports indicate that PDS still exists in the market [15], and the vetting system may not be trustable [24]. Since the vetting strategies on skill markets are unclear, the guarantee of the skill qualities on markets might not be explicit. However, probing the vetting mechanisms is highly challenging because of the various challenges presented below.

Challenges. First, from the developers' viewpoint, the vetting process works as a black box. The developers only know the rules requested by markets to prohibit sensitive skill behaviors but do not know how the markets guarantee rule enforcement. To probe the vetting process, what we can do is submit various skills to summarize the vetting strategy. The feedback we gain from the markets comes with minimal information (e.g., pass the vetting process or not). How to use limited information to outline the vetting strategies is extremely hard. Second, to our knowledge, no open materials disclose the internal mechanisms of the vetting process. Although recent research demonstrates that the vetting system is not trustworthy [24, 43], little has been done to study the internal reasons. Meanwhile, the markets neither show any hint of internal procedures of the vetting system. Note that vetting a skill is considerably different from vetting a smartphone application. A smartphone application's behaviors can be extensively verified by running the code submitted to the markets. However, the skills are in the form of services without a code available to the markets. Thus, previous studies that analyzed an application cannot directly be transferred into the skill vetting field. Third, the market strategies may evolve, and there is no ground truth to confirm the results. Therefore, it may be necessary for probing to perform a long-time analysis based on the results and attacks (i.e., evading the vetting process).

In this paper, we carefully designed a large variety of skills to address the above-mentioned challenges. To predict whether the testers behind the vetting process were humans or machines, we designed a set of skills with certain Turing test questions since the Turing test was a common way to measure whether one was speaking to a human or a chatbot [28]. We further leveraged additional information (e.g., userID and timestamp) to understand strategies from the communication between the markets and skills' online services in the vetting process. The evaluations were performed for eight months on two main skill markets (i.e., Amazon and Google), some of which were repeated several times in different periods to verify the results.

[1]In the rest of this paper, we use skills to refer to both skills and actions.

Finally, we obtained a number of interesting results after an 8-month long-time evaluation with over 200 skills submitted. For example, both humans and machines participate in the vetting process for Amazon and Google because humans are essential for understanding natural language questions from skills and generating suitable answers to continue the conversation. Because of the limit of time and cost, it may be necessary to replace certain humans with machine testers to handle certain questions. However, this may cause a misunderstanding of the questions from the skills and let the vetting process miss some sensitive behaviors.

We also find that not all voice commands can be evaluated because of the time limit, which may cause certain sensitive behaviors to be missed. We also generalize the traversing strategies from our probing, and attackers can easily develop a PDS to evade detection by understanding the strategies. Furthermore, we identify the machine testers' limitation of parsing the questions in natural languages from skills, which may lead to the vulnerability of an attacker being able to design a carefully prepared sentence to evade the detection of sensitive behaviors.

Attacks. Based on the above-mentioned results, we designed three attacks to embed sensitive behaviors in a skill and let them evade the vetting process using machine testers. These three attacks leveraged the NLP's weaknesses, including the weakness of sentence parsing, question understanding, and context understanding. So they cannot be defended by merely changing the strategies of machine testers. We tested these attacks end-to-end in the Amazon and Google markets and the state-of-the-art system SkillExplorer [41] and achieved a 100% success rate.

Ethical Issues. All user studies reported in this study (Sections 3 to 5) were carefully evaluated and approved by our IRB. Once we receive the vetting results, our script automatically removes the skills from the market so that no ordinary user would be involved in our experiments. Thus, our skills would not hurt real users.

We reported our attacks and results to Amazon and Google in March 2020, and Amazon has acknowledged some results (Section 4.2). We are communicating via email to help them understand such new security risks.

Contributions. The contributions of this study are summarized as follows:

• We investigate the vetting process of skills on the main markets (i.e., Amazon and Google) by carefully designed skills. To our knowledge, this is the first study that tries to understand how markets vet uploaded skills. Benefiting from the 8-month continuous probing, we have a number of interesting results (e.g., vetting strategies), and Amazon has acknowledged some results.

• We propose three attacks to evade the automated vetting process by machine testers and successfully publish the skills in Google/Amazon markets. They can confuse machine testers' parsing process without affecting the users' feelings. Then we design a linguistic-knowledge-based defense to mitigate the attacks.

2 BACKGROUND

2.1 Skill and Its Development

Skills and VPA Systems. Virtual personal assistant (VPA) systems usually built in smart speakers (e.g., Amazon Echo Dot and Google Home) are voice-controlled systems performing tasks and services according to the users' voice commands. In recent years, VPA systems, such as Amazon Alexa and Google Assistant, have gradually come into a large number of families. In addition to the VPA's built-in functionalities, third-party developers can create their voice applications (or *skills*), which extend the original VPA systems' capabilities.

Figure 1 shows an overview of how a skill works. Before interacting with a skill, a user needs to utter wake words (e.g., *"Alexa."*) to wake up the smart speaker. After the speaker is awakened, this user says his/her voice command, *"ask OurGroceries to add milk to shopping list."* (Step 1 in the figure). The voice command is then sent to the VPA server (Step 2) through the smart speaker (i.e., Amazon Echo) and parsed into the name of skills (i.e., "OurGroceries") with the command "add milk to shopping list." The command is further sent to the server provided by the skill developer (Step 3). Once the developer's server receives the requests, the skill performs corresponding actions (i.e., add milk to the shopping list of the user), generates responses (i.e., "Adding milk to

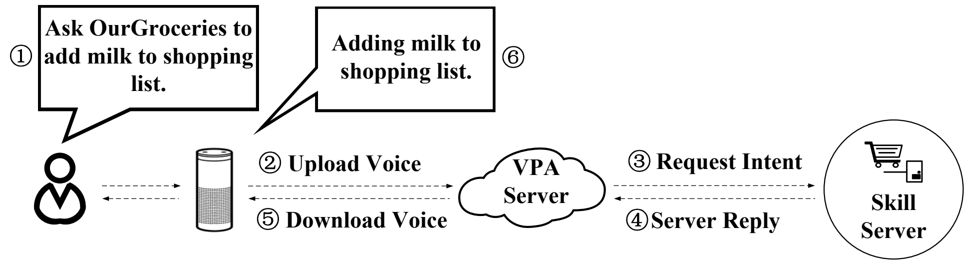


Fig. 1. Overview of user-skill interaction in VPA systems.

shopping list."), and replies to the VPA server (Step 4). Finally, the VPA server sends the responses to the user's smart speaker for replying (steps 5 and 6).

Note that, unlike smartphone applications that need to be installed explicitly, skills can be installed implicitly according to users' voice commands [56]. After receiving a voice command, the VPA server will extract the skill name and search for the corresponding skill in the markets. If such a skill exists, the VPA server will transparently trigger it for the user.

Developing a Skill. As previously mentioned, third-party developers are encouraged to publish skills and provide their services to users to extend smart speakers' capabilities. A skill typically includes two parts: back and front ends. The skill's back-end code is placed on a remote server provided by the developer, which receives and responds to the VPA server's requests. The skill's front end is an interaction model containing intents, sample utterances, and slots. An intent accomplishes users' spoken requests. The sample utterances are the phrases that users would use when making requests to smart speakers. For example, "add milk to shopping list" is the sample utterance. Note that after a skill is invoked, it will maintain an open dialogue with the user. The skill will send a specific request to the back end only when the user's voice command matches a sample utterance. Otherwise, the skill will report an error (e.g., "Sorry, I don't know about that. Please try again."). A slot is a variable in the utterance and can be mapped to an argument in the intent. In the previous example, *milk* is the slot that can be replaced by other items. A developer must register all of them in the VPA server to help the server parse the spoken requests from users and invoke corresponding skills with the parsed intents and slots.

In the communication between the front and back ends, requests and responses are embedded in JavaScript Object Notation objects. Together embedded in the objects are the skillID, deviceID, and userID used to distinguish different skills, devices, and users, respectively [9, 38]. Note that the userIDs of the same user are different in different skills. This was designed to prevent the same developer from tracking users by developing different skills.

2.2 Security of Skills

Threats from Skills. Existing work reported that certain potentially dangerous skills (PDS) are threatening user privacy. For example, a malicious skill can use an invocation name with the same or similar pronunciation of a legitimate skill (e.g., Boil an Egg and Boyle an egg), diverting users' requests to itself [50]. A PDS can also hijack a legitimate skill by concatenating its invocation name (e.g., OurGroceries) with other words (e.g., please). When a user says, "open OurGroceries, please." the PDS is invoked because of the VPA's longest string matching algorithm to identify skills [91]. In addition to using confusing names, some PDS become harmful by explicitly requesting the users' private information (e.g., personal identifiable information, phone number, health information, and credit card number) or by providing inappropriate content (e.g., violence [5, 40]). Some PDS have been reported to eavesdrop on users' conversations by designing a question with unpronounceable characters [15]. For example, a PDS can design a question only containing the reserved Unicode character *U+D801*. When the speaker reads the question, it will not make any sound because such a character is not a valid character [77]. It is difficult for the user to figure out whether the skill is running or not.

Vetting Skills. Providers vet skills before putting them online to protect users from PDS. For example, they can check the content to ensure no one violates market restrictions [11]. Different markets may have different vetting strategies, which are usually unclear to the public. Markets cannot vet skills by their code; they can only check a skill by capturing its questions and designing suitable answers to continue the conversation. For automatic analysis, natural language process (NLP) techniques should be utilized to understand the questions from skills and generate answers. However, because of the NLP's weakness and the long time required to finish the conversation, the vetting could be incomplete, which is another reason why the PDS exists on markets. In this study, we are motivated to understand the strategies of vetting skills and clarify the vulnerabilities for better protecting users.

3 HUMAN OR MACHINES?

Before exploring the vetting strategies, we should know whether the vetting is performed by human or machine testers or a combination of both.

3.1 Approach

The aim of vetting is to enumerate and verify all outputs of the uploaded skill, and a (human or machine) tester should correctly respond to outputs to continue conversations. As mentioned in Section 2.1, the vetting process contains an open dialogue. Therefore, we could design a skill's outputs and distinguish human and machine testers by observing their feedback. The whole process is similar to the Turing test to determine whether the remote tester can think like a human being [28]. Based on insights, we designed Turing test questions (e.g., riddles, see below for details) for verifying remote testers. Specifically, we embedded these Turing test questions in a custom intent (named *QuestionIntent*). Unlike the built-in intents [10, 36], which are predefined by VPA systems, the custom intents can be customized by the developers [6, 39]. When a tester invokes the *QuestionIntent*, our skill will provide a Turing test question but not tell him/her that the question is to test if he/she is a human. We introduce the details of such skills in Section 3.3. Note that the testers have to answer to continue the conversation and cover more paths. Therefore, we can determine if he/she is a human tester based on the evaluation of the answers.

We specifically chose two types of questions for the Turing test: riddles and pronoun disambiguation problems (PDP). For the former, we reported questions from a website called *Get Riddles* [64], which are suitable for humans aging eight years old and above. For the latter, we randomly selected questions from the *Winograd Schema Challenge*, an alternative to the Turing test [27]. Table 1 shows the details of the questions used and the suitable human ages. For example, Question 7 is, "The city councilmen refused the demonstrators a permit because they feared violence. Who feared violence?" Question 8 changes the verb *feared* in the question to *advocated*. The answers to the two similar questions are quite different, which requires human intelligence to answer.

We cannot merely rely on answers to distinguish the two types of testers because even human beings cannot guarantee to answer all questions correctly. Based on our observations, the machines' answers are usually wrong and irrelevant to the question. However, humans' answers are relevant to the question in most cases, although they are sometimes wrong answers. Therefore, we used the relevance of the answer to the question to distinguish human testers from machine testers. Note that a lackadaisical human may behave like a machine; hence, their answers may also be irrelevant. We classified them as machines because such humans' vetting results are the same as machines'.

Before introducing our relevance calculation algorithm, we first introduce our pre-processing method. Note that it is not difficult for a machine to form an answer by duplicating the question, and such an answer is always

Table 1. Questions used in our user study. Pct.: percentage of relevant answers; ACC, FPR, and Recall: accuracy, false positive rate and recall, respectively. DUR.: duration of the audio of each question.

Type	ID	Question	# Answers	Pct.	ACC.	FPR.	Recall	DUR.
Riddles	1	I have wings, I am able to fly, I'm not a bird yet I soar high in the sky. What am I?	162	87.65%	93.21%	5.00%	92.96%	6s
	2	You'll find me in a soup, in a burger, in a pizza, I am green when raw and red when ripened and ready to become a condiment. What am I?	139	90.65%	84.89%	0%	83.33%	8s
	3	I have no doors but I have keys, I have no rooms but I do have a space, you can enter but you can never leave. What am I?	124	89.52%	83.87%	7.69%	82.88%	7s
	4	I call the trees my home, yet I never go inside, and if I ever fall off the tree I will be surely be dead. What am I?	118	87.29%	94.92%	20.00%	97.09%	7s
	5	People say I put doctors out of business, sometimes I am sour, sometimes I am sweet, I can be eaten and can also be drunk. What am I?	111	90.99%	92.79%	10.00%	93.07%	8s
PDP	6	The city councilmen refused the demonstrators a permit because they feared violence. Who feared violence?	109	94.50%	95.41%	0%	95.15%	6s
	7	The city councilmen refused the demonstrators a permit because they advocated violence. Who advocated violence?	109	91.74%	94.50%	0%	94.00%	6s
	8	The trophy doesn't fit into the brown suitcase because it's too small. What is too small?	108	95.37%	99.07%	0%	99.03%	5s
	9	The trophy doesn't fit into the brown suitcase because it's too large. What is too large?	108	95.37%	100.00%	0%	100.00%	5s
	10	Joan made sure to thank Susan for all the help she had given. Who had given help?	109	93.58%	99.08%	0%	99.02%	4s
	11	Joan made sure to thank Susan for all the help she had received. Who had received help?	108	97.22%	98.15%	0%	98.10%	5s
	12	Paul tried to call George on the phone, but he wasn't successful. Who was not successful?	108	94.44%	98.15%	0%	98.04%	5s
	13	Paul tried to call George on the phone, but he wasn't available. Who was not available?	108	94.44%	100.00%	0%	100.00%	5s
	14	The lawyer asked the witness a question, but he was reluctant to answer it. Who was reluctant to answer the question?	108	93.52%	99.07%	0%	99.01%	6s
	15	The lawyer asked the witness a question, but he was reluctant to repeat it . Who was reluctant to repeat the question?	108	90.74%	98.15%	0%	97.96%	6s
	16	The delivery truck zoomed by the school bus because it was going so fast. What was going so fast?	108	93.52%	98.15%	14.29%	99.01%	6s
	17	The delivery truck zoomed by the school bus because it was going so slow. What was going so slow?	108	94.44%	99.07%	0%	99.02%	6s
Average					95.79%	3.35%	95.74%	

relevant to the question. We conducted a sentence-level similarity analysis between the answer and the question to reduce the impact under these circumstances for answers with many words (e.g., four words). We particularly used the *BLEU* algorithm [60] for the similarity calculation. When the score was higher than a threshold (e.g., 0.05), we believe that the answer came from the duplication and considered the answerer as machines. Moreover, we removed all the stop words in the questions and answers during pre-processing because stop words (commonly used words [76]) in the questions have nothing to do with the questions' information, such as the word "the."

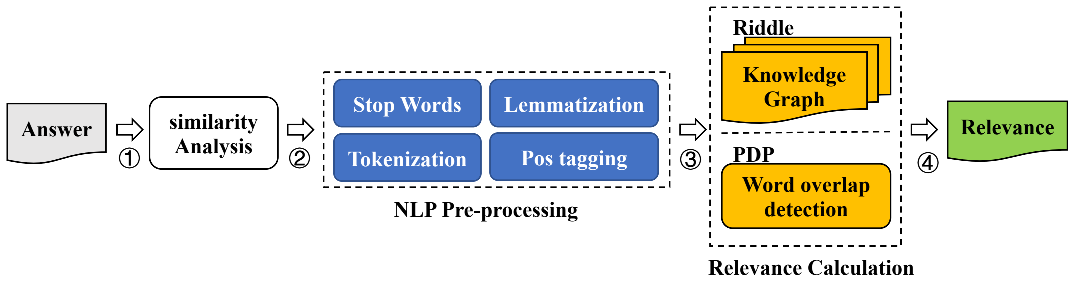


Fig. 2. The process of calculating the relevance between questions and answers.

Algorithm 1: Calculating relevance for PDP.

```
Result: Relevance
1  subject_words = getSubject(question);
2  object_words = getObject(question);
3  answer_words = preprocessing(answer);
4  in_subject = in_object = False;
5  relevance = 0;
6  for word in answer_words do
7      if word in subject_words or object_words then
8          relevance += 1;
9          if word in subject_words then in_subject = True ;
10         else in_object = True ;
11 end
12 if in_subject and in_object then relevance = 0 ;
13 if len(answer_words) == 0 then relevance = 0 ;
14 else relevance /= len(answer_words) ;
```

Algorithm 2: Calculating relevance for riddles.

```
Result: Relevance
1  question_words = processing(question);
2  answer_words = preprocessing(answer);
3  relevance = 0;
4  for word in answer_words do
5      related_items = conceptNet(word);
6      for item in related_items do
7          if item in question_words then
8              relevance += getWeight(item);
9          end
10     end
11 end
12 if len(answer_words) == 0 then relevance = 0 ;
13 else relevance /= len(answer_words) ;
```

We developed a system to automatically calculate the relevance between the answers and the questions (Figure 2) and determine whether the answer is relevant. After the NLP pre-processing, we designed two algorithms for riddle-based questions and PDP questions to calculate the relevance. Algorithm 1 shows the algorithm for the PDP question. For PDP questions, we can determine whether an answer is relevant by judging whether the answer contains the words in the subject (Line 9) or the direct object (Line 10) of the question because their answers always appear in the subject and direct object. Note that the previous similarity analysis filters out answers with a similar length to the question and answers with many overlapping words and will not filter out the regular answers. For such questions, we set the relevance as the number of overlapping words with the subject and the direct object. Note that since an answer cannot contain both the words in the subject and the direct object, we regard such an answer as irrelevant (Line 12). However, the riddle's answer is not formed by directly picking the words from sentences but inferred through existing knowledge. Algorithm 2 shows the algorithm for riddle-based questions. To discover the unseen link between answers and questions, we used a knowledge graph to derive the knowledge related to the words appearing in the answer by reasoning. We particularly used the knowledge graph in [68], which contained much common-sense information necessary for solving our questions.

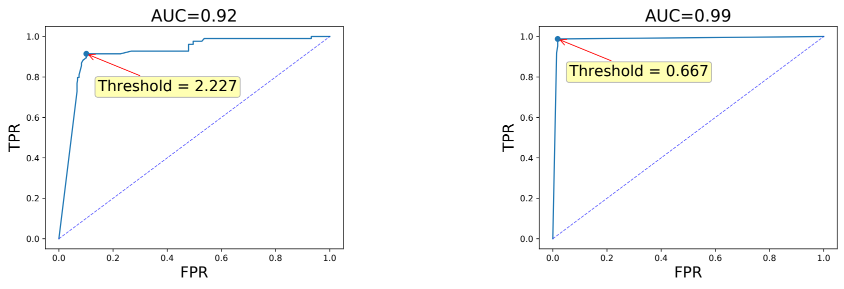
For each keyword, we used the knowledge graph to search for its related item and compare it with the question (Line 7). We regarded the sum of the weights of all matching items as relevance (Line 8). Based on the calculation method, an answer containing many words may be falsely determined as very relevant to the question. Thus, it is necessary to decrease the relevance of the answers with too many words. We divided the previously calculated relevance by the total number of words in the pre-processed answer as the weighted relevance to balance the relationship between the answer length and relevance (Line 14 in Algorithm 1 and Line 13 in Algorithm 2). Finally, we considered the answers with a relevance higher than the threshold as relevant answers.

3.2 Evaluation

We surveyed on Amazon Mechanical Turk (MTurk) [3], an online marketplace for crowdsourcing intelligence, to verify whether our method can distinguish between relevant and irrelevant answers. Each participant was required to answer the 17 questions listed in Table 1. We placed certain trap questions in the questionnaire to assess whether participants randomly enter answers without caution. We recruited 182 human users from MTurk to answer questions, of which 74 did not complete the questionnaire, and eight answered without caution. In the end, we obtained 100 valid questionnaires and paid \$0.5 for the users (average completion time: 13 min). Their ages ranged from 18 to 70. Accordingly, 13% of the participants were 18–25 years old; 37% were 26–35 years old; 27% were 36–45 years old; 12% were 46–55 years old, and 11% were over 55 years old. 64% were male, and 36% were female.

To fairly judge whether each answer is relevant to the question, we hired five college students to assess whether these answers were relevant to the question and adopted majority rules when opinions differed. Based on their judgment, 149 answers were irrelevant. Column 4 of Table 1 shows the number of answers collected for each question. Column 5 lists the percentage of relevant answers. The vast majority of answers were relevant to the question (92.54% on average), proving that humans' answers are often relevant to the question.

To make a reasonable classification based on relevance obtained by our algorithms, we needed to find the optimal threshold for the two types of questions. We first used our proposed method to calculate all answers' relevance and selected 1000 values in the relevance range of these two types of questions as the thresholds to draw the Receiver Operating Characteristic (ROC) curves, as shown in Figure 3. According to [61], the optimal threshold point is the point of the ROC curve closest to (0,1). We found the optimal thresholds for riddles and PDP (2.227 and 0.667).



(a) ROC curve of riddles. (b) ROC curve of PDP questions.

Fig. 3. ROC Curve of the 1000 thresholds.

With the optimal threshold we found, we calculated the accuracy, false-positive rate, and recall as shown in columns 6–8 of Table 1. The accuracy ranged from 83.87% to 100.00%. The average accuracy for all our questions was 95.79%, indicating that our classification method has high accuracy. The average false positive rate and the recall of our classification were 3.35% and 95.74%, respectively, implying that our method had a lower risk of identifying relevant answers as irrelevant and, thus, may avoid revealing our operations to human testers. Note that 3.35% was the average false positive for each question. Our skills often contained several questions; therefore, the actual false positive rate will be much smaller. For example, the average false positive rate for two of our 17 questions was 0.0904%, and the value decreased to 0.0018% when the number of questions was three.

We selected eight best AI chatbots from [16] and let them answer our questions to verify whether our method is effective for the machine's answers. The eight chatbots are Mitsuku [58], Cleverbot [19], Replika [51], Rose [78], Jabberwacky [18], A.L.I.C.E. [59], Elbot [67], and Simsimi [45]. Table 2 shows the percentage of questions correctly answered by each chatbot. The specific answer of each chatbot is shown in Appendix A.1. Note that although Rose seems to be good at dealing with PDP questions, the average percentage of relevant answers for each chatbot is only 15.0% and 11.5% for riddles and PDP questions, and the median is 10.0% and 4.2%, indicating that most chatbots cannot give a relevant answer. Therefore, we considered those answerers who give at least one relevant answer as humans.

Table 2. Percentage of questions answered with a relevant answer by each chatbot.

Bot	Mitsuku	CleverBot	Replika	Rose	Jabberwacky	A.L.I.C.E	Elbot	Simsimi	Average
Riddle	60%	20%	0	0	0	0	20%	20%	15%
PDP	8.3%	8.3%	0	66.7%	8.3%	0	0	0	11.5%

3.3 Probing

Note that if the questions are too hard, even human testers' answers may not be correct because their goal is to continue the conversation rather than correctly answering the questions. Human testers may not be patient enough to think about the correct answers. Hence, we designed a skill with several Turing test questions organized as follows: a tester can only get the next question after correctly answering a question. The testers are trying to enumerate as many intents as possible; therefore, they should try their best to answer the questions. Note that some questions may be too hard to answer because of the limited knowledge of human beings correctly. Hence, we designed another skill with the Turing test questions. The difference between this skill and the previous one is that it randomly selects a question for the testers. In this way, even if the answer to one question is wrong, the testers still have a chance to answer the other questions.

We developed four custom skills[2] for evaluation, and two of them contained riddles. We call these two the *Riddle Researcher* and the *Riddle Master*. The difference between them is the way they pick up questions, as previously mentioned. The other two skills ask questions from the *Winograd Schema Challenge*, named *Test your comprehension* and *Comprehension test*, and still use different methods to choose questions. For simplicity, we refer to the four skills as *Skill-R1*, *Skill-R2*, *Skill-W1*, and *Skill-W2*. We then uploaded the four skills to the two markets: Amazon and Google. We later recorded each answer's relevance and the time intervals for every two answers, which could estimate the time needed to generate answers. Moreover, we recorded the identifiers (*userID*, Section 2.1) to distinguish different testers in the same skill. We submitted each skill for five times and collected the results.

[2]All the skills created in this paper are of the custom type.

3.4 Results

In the five attempts[3], 268 and 56 userIDs invoked our skills during the vetting process, with 72 and 266 answers in Amazon and Google, respectively. Table 3 shows the results of the statistical analysis. 1.9% of the userIDs in Amazon and 55.8% in Google gave at least one relevant answer. According to the above-mentioned threshold, we speculated that these userIDs are from humans.

Table 3. Results of the statistical analysis in the five attempts of probing.

	UserIDs				Answers		
	#relevant answers>0	#relevant answers=0	#answers=0	#total	#irrelevant	#relevant	#total
Amazon	5	32	231	268	66	6	72
Google	24	19	13	56	170	96	266
Sum	29	51	244	324	236	102	338

Interestingly, our results indicated that most userIDs (84.3%) in the two markets who gave irrelevant answers spent a very short time answering a question (i.e., less than 2 seconds). We measured the minimum time required for a human to answer a question by converting the questions into audio through the Amazon text-to-speech (TTS) service and recording the corresponding duration (Column 9, Table 1). The duration represents the minimum timing requested by human testers to finish answering a question. These durations were much longer than 2 seconds, indicating that these userIDs were more likely from machines or lackadaisical humans (also classified as machines, see Section 3.1). We verified our classification approach based on this metric. Many humans would say something when they are unable to answer, such as "I don't know," "no idea," and "pass," which are irrelevant to the question. We treat these commonly used sentences as relevant answers. Compared with the classification results based on our classifier, five userIDs (6.25%) were inconsistent with our classification results. The accuracy was 93.83%, and the false-positive rate and recall were 4.44% and 91.67%, respectively, indicating that our classifier accurately distinguished between humans and machines.

Note that we distinguished humans from machine testers based on the relevance of the answers instead of the response interval, which we used only for verifying our proposed method. The verification results proved that our classifier only needs relevance to make a precision classification. In the future, Amazon and Google can manipulate the response interval. However, our classification results will not be affected.

3.5 When Do Human and Machine Testers Vet Skills?

We tried to understand herein when human and machine testers vet skills. To achieve this goal, we developed 42 skills and submitted one of them to Amazon and Google markets every two hours within 14 days[4]. Each skill was submitted four times. As expected, human testers only work on weekdays, which is different from machine testers (vetting skills every day). Specifically, human testers in Amazon primarily work from 10 pm to 10 am on weekdays (Eastern Daylight Time), while human testers in Google work day and night on weekdays. Furthermore, we found that both markets employ machine testers to vet a skill before human testing. On average, based on our observation, a machine tester starts to vet skills 222 seconds after the submission. The time for human testers is 11.35 hours. Details of vetting time are shown in Appendix A.2.

4 EXPLORING THE STRATEGIES OF MACHINE TESTERS

Analyzing human testers' strategies is tricky because of the uncertainty of human behavior. In this study, we primarily investigated machine testers' strategies. Our previous results showed that machine testers always vet a

[3]We submitted our skills to Amazon market from February 6 to February 15, 2020, and to Google market from April 21 to April 27, 2020.
[4]We submitted these skills from 12 am on April 20 to 12 am on May 4, Eastern Daylight Time.

skill immediately after submission and much more quickly than human testers. We withdrew the submitted skills immediately after receiving the vetting results of machine testers to ensure that no human testers are involved in the experiments [5].

4.1 How Do Machine Testers Initiate the Conversation with a Skill?

We tried herein to understand how machine testers start conversations with skills. The search space of voice commands is infinite; thus, testers must identify a suitable voice command to start the conversation. As Amazon and Google employees, testers may be able to see all intents registered by developers and directly call a specific intent with a sample utterance. In this case, testers can access any intent that developers try to hide. However, directly invoking these intents may cause abnormal results because certain intents have contextual information, which may cause the vetting results to be inconsistent with the actual results. Therefore, testers may imitate regular users' behavior to invoke a skill's intent (i.e., through interaction with the skill). For example, testers can imitate users to learn the basic functionalities through the voice command "help." In the vetting process, if the testers cannot select the suitable intents to start with, the vetting could be incomplete, indicating that they cannot verify all the functionalities in the skill.

Approach. We built a skill with ten intents to determine how machine testers perform vetting. Appendix A.3 shows the interaction model of the skill. The intents *Launch*, *Help*, *Cancel*, and *Stop* are built-in intents that developers can select to implement without providing any sample utterances (*Help* is not a built-in intent in Google). The *HiddenA*, *HiddenB*, *HiddenC*, *VisibleA*, *VisibleB*, and *VisibleC* intents are custom intents whose names and lists of utterances should be provided by developers. Note that the sample utterance of the hidden intents does not show in the responses of the *Launch*, and *Help* intents but can still be triggered by users. We then submitted this skill to Amazon and Google markets and performed five times of evaluation[6]. Note that our results may change with the strategy updating in Amazon/Google, but our method still works and can be used to obtain new results.

Results in Amazon. Interestingly, for the five times of vetting, Amazon always inspected the skills using eight different userIDs. Here, we regard a unique userID as a unique machine tester. The eight testers performed the vetting process in sequence. We number the eight userId in the order of their appearance, and Table 4 shows the vetting tasks of different testers.

Table 4. Behavior of eight machine testers in the Amazon market.

ID	The called intents in sequence	ID	The called intents in sequence	ID	The called intents in sequence
1	Launch, Help	4	Launch, Stop	7	Launch, Stop
2	Launch, Custom Intents	5	Launch, Cancel	8	Launch, Launch
3	Launch, Stop, Launch, Stop, Launch, Stop	6	Launch		

The first machine tester launches the skill (i.e., using the intent *Launch*) and calls the intent *Help*. It mimics the manner of operation of a human user who starts the conversation with the intent *Help*, a popular approach to start an unfamiliar skill. The second tester calls the custom intents (e.g., *VisibleA*, *VisibleB*, and *VisibleC*). The third tester continues to launch and stop the skill thrice. Maybe it was designed to test whether the skill can behave normally when launched and exited frequently. The fourth, fifth, sixth, and seventh testers were similar to the third tester. One difference was the cancel operation, which was for canceling a task without exiting and was different from the exit operation. Here, one interesting finding is that the machine testers cooperate. For

[5]All the skills in this section were submitted to different categories (e.g., "Games&Trivia", "Lifestyle", and "Kids"). We also tested skills requesting permissions, and the results were the same.

[6]We submitted these skills to Amazon market in November 2020, and submitted to Google market in April 2020.

example, the sixth tester starts the skill, and the seventh tester starts it again, but with a different userID, which mimics two different users calling the same skill.

Results in Google. In Google, only one userID participates in each vetting process, which launches the skill and calls the custom intents just like the second tester in Amazon. One difference is that Google's tester only selects one intent from *Launch* to check. After five times of probing, we found that only the intent *VisibleC* was invoked.

In very rare situations, the hidden intents may be checked (Appendix A.5). Note that although machine testers in both Amazon and Google have the capability to enumerate all the custom intents, they do not usually do this. Instead, they only check the intents invoked from "Launch" or "Help."

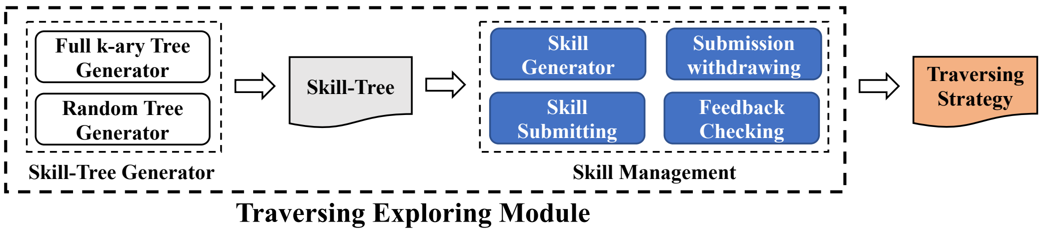


Fig. 4. Architecture of the traversing exploring module.

4.2 How Do Machine Testers Traverse Intents in a Skill?

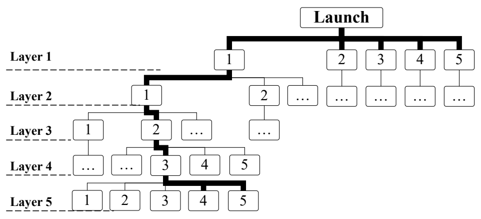
Machine testers only invoke intents from skill responses; thus, it is feasible to influence machine testers' vetting process by controlling the skill responses. A skill may contain many intents; thus, we try to understand the vetting strategies to traverse the intents. Can machine testers check all intents? If so, the risk left to users could be minimum. If not, how many intents can they verify, and which intents do they select? We developed an automatic exploring module to discover the intents that machine testers cannot invoke and understand the vetting strategies (Figure 4).

Before introducing the automatic exploration module, we first describe herein the hierarchical structure of a skill, which is a tree-like structure. We refer to the skill structure as a *skill-tree*. Each node in the skill-tree represents an intent, and a command in the parent node can trigger the child node.

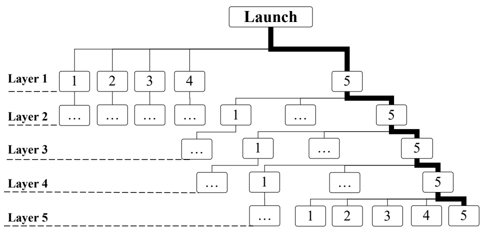
Skill-Tree Generator. We designed two skill trees with different widths and depths to explore traversing strategies. The first skill-tree had k layers, and each intent had k children (i.e., full k-ary tree). We can systematically explore the strategies by increasing the value of k from 1. Moreover, considering some exceptional circumstances, we designed a skill with a random number of layers and intents (i.e., random tree). For evaluation, we generated ten full k-ary trees with k from 1 to 10 and ten random trees with layers < 20 and intents < 10. We did not use too many layers or intents because a complicated skill may not be user-friendly.

Skill Management. We implemented a module to manage the lifecycle of skills, including skill generating, skill submitting, submission withdrawing, and feedback checking. The generation of skills is implemented based on the Jovo framework [31], a cross-platform skill development framework. For each intent in the skill-tree, we put a simple sentence that can always be correctly parsed by machine testers, which is in the form of "If you want to do n, you can say n," where n is a unique number for each intent. In this way, if the tester returns the corresponding number, we will know which intent is invoked. Then the management module will submit the skill and immediately withdraw it after receiving the vetting results, which was implemented by Alexa Skills Kit Command Line Interface [4] (for Amazon) and mocking requests (for Google). In this way, we can summarize the traversing strategy by analyzing the feedback of the 20 skill-trees.

Results in Amazon. After testing the 20 skills[7], we reported that the traversing strategies were constant. We tested these skills in different periods and reported that the results were the same. If a skill had only one layer, the machine testers verified all the intents. Suppose a skill had n layers ($1 < n \leq 7$), the machine testers first verified all the intents in the first layer and then verified the $(k-1)$th intent in every kth layer ($1 < k \leq n$). Finally, they verified the $(n-1)$th intent and its right-sibling intents in the nth layer. For example, the traversing path for a full five-ary skill-tree is marked with bold lines in Figure 5(a). The result when $n > 7$ was the same as when $n = 7$. In particular, if there were less than $(k-1)$ intents in the kth layer ($1 < k \leq n$), the machine testers checked the $(k-2)$th intent with its right-sibling intents in the $(k-1)$th layer (As shown in Appendix A.4).
Results in Google. The strategy of Google is straightforward. From the results, we reported that the machine testers always only verified the last intent in each layer (Figure 5(b)). For a skill with n layers ($n \leq 20$), the machine testers only checked each layer's last intent. If the skill had more than 20 layers, the machine testers would end the session after traversing the last intent in the 20th layer. Once exiting a skill (actively or passively), the machine testers would finish their vetting job, although many unchecked intents still existed, which was different from Amazon.



(a) Traversal results in Amazon.



(b) Traversal results in Google.

Fig. 5. Traversal results for a full five-ary skill-tree in Amazon and Google. The bold lines indicate the traversal paths. We omitted some nodes from the figure due to the space issue.

Confirmation. We reported this finding to Amazon Security Team. They said, "In addition to the one that you outline in your document, we have other methodologies to ensure that Skills behave correctly." As mentioned before, although Amazon may change its strategy, we can still use our method to detect its new strategy.

4.3 How Do Machine Testers Parse a Sentence?

In this section, we try to understand the weakness of the machine tester to parse questions. The questions are usually in two forms: interrogative sentence and command sentence. Based on our observations, given an interrogative sentence, the machine tester from Amazon will randomly pick a word from the sentence as the answer. Google's machine tester will search the question directly and return the results as answers. So both the two testers seldom parse the interrogative sentence. For the other form, the testers will parse the questions. So we aim to find the weakness of parsing the second form.
Approach. We try to achieve this goal by developing a mutation fuzzing tool to generate voice commands that may trigger vulnerabilities automatically. To ensure that real users can still understand the mutated voice commands, we need to keep their grammar correct and their semantics unchanged, similar to the paraphrasing process. We collected paraphrasing rules from websites, such as in [57, 71], and summarized some rules to paraphrase our questions (Table 5). Figure 6 shows the architecture of the mutation fuzzing module. First, we

[7]We submitted these skills to Amazon market in October 2020 and submitted to Google market in April 2020.

selected command sentences in the developers' documentation as a seed and pre-processed the sentences' text. We then used a paraphrasing model to generate their mutated sentences. Combined with the traversal strategy obtained by the traversing exploring module in Section 4.2, our skill management module can put the mutated questions on the proper nodes in skill-trees, which will be checked by machine testers during the vetting process, thereby generating efficient test skills.

Table 5. Rules for paraphrasing.

Rule	Description	Example	Rule	Description	Example
R1	Using synonyms for all words.	Before: want After: need	R3	Simplifying the sentence.	Before: you can say help After: say help
R2	Using synonyms for all words.	Before: want After: need	R4	Adding meaningless punctuation	Before: say help After: say .help

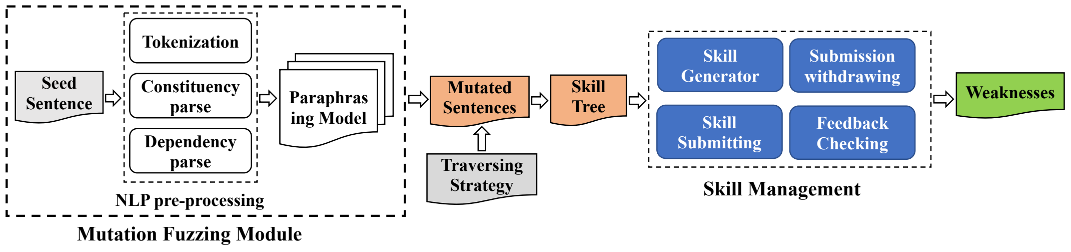


Fig. 6. Architecture of the mutation fuzzing module.

Evaluation. For Amazon, we chose two sentences as the seed, namely, "If you want to get A, you can say A" and "If you want to get A, you can ask a question like A." (for simplicity, we refer to the two sentences as Command-S and Command-A, respectively) For Google, we surrounded the commands in the two sentences with quotes. These sentences are commonly used in skills and can be parsed well by machine testers in the corresponding market. We then put the sentences in the module, obtained 108 sentences for each market, and embedded them into our skills. Subsequently, we submitted the skills to the two markets[8]. Note that for other markets using machine testers, the approach can be applied similarly. In this process, the machine testers successfully invoked all questions.

Results in Amazon. A total of 33 sentences were correctly answered. Among the 75 sentences that were not correctly answered, 26 obtained wrong answers, and 49 sentences acquired nothing. We manually analyzed these command sentences and tried to determine why machine testers cannot parse the questions correctly. Table 6 presents the results.

The first reason is changing the sentence structure. In our evaluation, we found that machine testers cannot correctly parse both questions when the subordinate clause is after the main clause. We take the question "You can say A, if you want to get A" as an example. The machine tester will extract two commands from the questions (i.e., "A" and "if you want") (Table 6). Although this may not affect the testing coverage (the voice command "A" can successfully trigger the correct intent in our skill), the second voice command "if you want" does not exist. We guess this is because the machine testers view all the contents after the word "say" as commands and split them by commas.

[8]We submitted these skills to Amazon market in October 2020 and submitted to Google market in April 2020.

Table 6. Reason for the command sentences not correctly parsed in Amazon. The position indicates whether the clause is before or after the main clause.

Type	Reason		Sentence	Result
Command-S	Position		You can say A, if you want to get A.	A
				if you want
	Synonyms	Pron.	If you want to get A, I/we/she can say A. [1]	[No reply]
		Modal	If you want to get A, you may/might/must/should say A. [1]	[No reply]
		Verb	If you want to get A, you can tell/answer A.[1]	[Same as the question]
			If you want to get A, you can speak/reply/utter A. [1]	[No reply]
	Modifier		If you want to get A, you say A.	[No reply]
			If you want to get A, say A.	[No reply]
	Extra punctuation		If you want to get A, you can say .A/...A. [1]	.
			If you want to get A, you can say !A/?A.[1]	[No reply]
			If you want to get A, you can say :A/;A/(A)/[A]/{A}/-A/'A'/"A"/- A.[1]	:A/;A/(A)/[A]/{A} /-A/'A'/"A"/- A.[1]
Command-A	Position		You can ask a question like A, if you want to get A.	A
				if you want
	Synonyms	Conj.	Providing/Supposing/When/Assuming that you want to get A, you can ask a question like A. [1]	[No reply]
		Prep.	If you want to get A, you can ask a question for instance /for example/such as A. [1]	a question for instance /for example/such as A. [1]
		Pron.	If they want to get A, you can ask a question like A.	[No reply]
			If you want to get A, he/she/I/they/we can ask a question like A. [1]	[No reply]
		Modal	If you want to get A, you could/may/might/must/should ask a question like A. [1]	[No reply]
		Verb	If you want to have A, you can question/inquire/quiz/query a question like A.	[No reply]
	Modifier		If you want to get A, you ask a question like A.	[No reply]
			If you want to get A, ask a question like A.	[No reply]
	Extra punctuation		If you want to get A, you can ask a question like .A/...A. [1]	.
			If you want to get A, you can ask a question like !A/?A.[1]	[No reply]
			If you want to get A, you can ask a question like :A/;A/(A)/[A]/{A}/-A/'A'/"A"/- A.[1]	:A/;A/(A)/[A]/{A} /-A /'A'/"A"/- A.[1]

[1] Considering the space issue, we put similar sentences together by a slash to indicate "or".

Second, we find that synonyms affect natural language processing. For example, if we change the word "can" in the question to "may," "must," "might," or "should," machine testers will not reply, which means they cannot extract the voice command. Furthermore, if we change the verb "say" to other words like "utter," "speak," or "reply," no answer will be given (Table 6). Amazon's algorithms to process natural language seem to have some vulnerabilities in parsing words in a sentence.

Finally, simplifying or adding meaningless punctuation can sometimes affect the parsing results. For example, machine testers in Amazon will answer nothing if we delete "can" or "you can," Moreover, if a period exists before the command "A," machine testers only extract the period mark as the command and feedback to our skills. As for the other punctuations (e.g., exclamation and question marks), the results are still incorrect (no answer or answers together with the punctuation).

Results in Google. All 108 sentences were correctly answered in Google. Machine testers in Google always extract anything in the quotations as a command, including punctuation. Although no answer changed in the experiment, we can still easily change the machines' answers by modifying the range of quotation marks. For example, we can change the sentence "You can say 'A"' to "You can 'say A"' to let their answers be "say A" rather than "A."

In summary, based on the analysis mentioned above, we find that the techniques for parsing a question by machine testers have some vulnerabilities. Consequently, machine testers may extract incorrect answers as commands or even not answer. Attackers can carefully craft a sentence exploiting the vulnerabilities and embed another sensitive content in its child node. Machine testers cannot reach such embedded content so that the content escapes being checked.

5 EXPLOIT VETTING MECHANISM

Although our primary purpose is to probe the vetting process's vulnerabilities, we still attempted herein to propose three types of attacks to evaluate these vulnerabilities' hazards. The main idea of our attack is to leverage the inner weakness of machine testers, that is, the difference between human and machine in understanding sentences. So we can distinguish machine testers from humans and make sure that only humans can access our malicious content (e.g., asking for the credit card numbers). As a result, the malicious skills can bypass the vetting process of machine testers with our attacks. Below are the details of the three attacks that cannot be defended by merely changing the vetting strategies.

5.1 Approach

Question Mutation Attack (QMA). This attack directly mutates a sentence's structure or replaces certain words with their synonyms while keeping the sentence with the same meanings. The four ways of modifying a sentence in Section 4 were typically adopted. As previously mentioned, these modifications will cause the machine testers to fail in parsing the sentences correctly, further breaking the conversation. Attackers can hide malicious intents after the sentences to evade vetting. We call such sentences entry sentences. Note that not all sentences mutated in this way can pass the vetting process based on our previous evaluation, which may expose the malicious skill. For example, the sentence "If you want to get A, you could say A" can still be correctly parsed by Amazon's machine tester. To increase the attack success rate, attackers should first test whether machine testers can correctly parse the generated question. Fortunately, Amazon provides a pre-certification test, called the *Functional test* [7], for developers to test their skills. It performs the same as the machine testers in parsing questions and does not report to Amazon, even if the skill seems to be malicious. Thus, the attacker can check their malicious skills using the pre-certification test before really submitting the skills.

Understanding Deviation Attack (UDA). We observe that in most cases, the machine testers respond to a sentence directly using the words extracted from the sentence, which may not be the same as the users' responses. Thus, attackers can design an entry sentence, letting the machine testers respond differently. We take the sentence, "You can say whatever you want" as an example. The machine testers will reply, "whatever you want," while human users will reply differently. Hence, we can create two intents after the sentence. One responds to "whatever you want," while the other intent responds to other answers. We can conceal the malicious part of the skill in the following intent. In this way, the machine testers will explore further intents after the first intent and miss the malicious one.

Confusing Pronoun Attack (CPA). It is hard for machine testers to correctly pick out the referent of a pronoun in a sentence, which may be due to natural language processing limitations. Attackers can leverage this weakness to embed malicious intents. For example, in the sentence "A man may forget his credit card number, but you can say it to avoid," the pronoun "it" refers to "credit card number" Nevertheless, it is hard for machine testers to identify. They usually directly reply with the word "it." The attackers can use two intents to attack: putting the referent in one intent and using the pronoun to refer to it in a later intent. For example, the skill can first say, "If you remember your credit card number, you can say yes," and wait for the reply. If the reply is "yes," the skill

Table 7. Survey results. ASR stands for the attack success rate, representing that we successfully obtained information from users. Pct. is the proportion of people whose answers are similar to those of the machine testers.

Attack	ID	Sentence	Reply of machines	ASR	Pct.
QMA	1	If you want to get new movies, you can say .search new movies	.	86%	0%
	2	You can say weather if you want to check weather	weather if you want	70%	0%
	3	If you want to get a joke, you may say tell me a joke	[No reply]	82%	0%
	4	If you want to get today's news, say today's news	[No reply]	82%	0%
	5	If you want to get started, you can say !start	[No reply]	89%	0%
	6	When you want to get something interesting, you can ask a question like what is funny	[No reply]	78%	0%
	7	If you want to get my information, we can ask a question like who are you	[No reply]	85%	0%
	8	If you want to get a pizza, you can ask a question like .order a pizza	.	77%	0%
	9	If you want to get recommended tourist cities, ask a question like where should I go	[No reply]	81%	0%
	10	When you want to get your location, you can ask a question like where am I	[No reply]	86%	0%
UDA	11	If you want to start a conversation, you can say whatever you want	whatever you want	78%	22%
	12	If you want to check the food calories, you can say the name of the food	the name of the food	64%	7%
	13	If you want to get new product recommendations, you can say a brand name	a brand name	63%	8%
	14	If you want to check the weather, you can say a city	a city	63%	8%
	15	If you want to generate a playlist, you can say a singer	a singer	64%	9%
	16	If you want to get a better experience, you can say your name	your name	63%	13%
	17	If you want to get the following content, you can say your age	your age	67%	17%
	18	If you want to get some fashion advice, you can say your gender	your gender	70%	13%
	19	If you want to get some interesting facts about animals, you can say your favorite animal	your favorite animal	70%	14%
	20	If you want to get daily horoscope, you can say your zodiac sign	your zodiac sign	68%	9%
CPA	21	If you want to share something, you can say it	it	99%	1%
	22	If you have a favorite food, you can say it	it	77%	1%
	23	If you have any clothing brand to share, you can say it	it	77%	1%
	24	If you have a city to visit, you can say it	it	78%	2%
	25	If you have a favorite singer, you can say it	it	76%	2%
	26	1.If you have a nickname, you can say yes. 2.To record, you can say it.	1.yes. 2.it	57%	2%
	27	1.If you have an age over eighteen, you can say yes. 2.To record, you can say it.	1.yes. 2.it	66%	2%
	28	1.If you do not mind revealing your gender, you can say yes. 2.To record, you can say it.	1.yes. 2.it	71%	2%
	29	1.If you have a favorite pet, you can say yes. 2.To record, you can say it.	1.yes. 2.it	72%	2%
	30	1.If you remember your zodiac sign, you can say yes. 2.To record, you can say it.	1.yes. 2.it	69%	2%

continues to say, "You can say that." Human users can now understand the referent of the pronoun "that" is the "credit card." However, machine testers cannot understand this question.

Note that timeout and wake words are important security mechanisms in the VPA systems. However, they do not affect our attacks. In most cases, the speaker will set a timeout to not wait too long for the user's response [8, 37]. This timeout only exists when the skill waits for the user to reply. Our attacks occur when the skill speaks to the user. Therefore, the timeout does not affect the attacks. As for wake words, different wake words have different levels of security. For example, The risks of falsely waking the speakers are different [66]. Since our attacks occur after the user successfully awakes the speaker and invokes the skill, the wake words do not affect our attacks.

5.2 Evaluations.

Bypassing Machine Testers and Performing End-to-end Attacks. We designed 30 entry sentences (ten for each attack, Table 7) and developed six skills (two for each attack) to demonstrate whether our attacks can distinguish between humans and machines. Note that these sentences do not contain malicious content. We then submitted these skills to the two markets and immediately withdrew our submissions after receiving the vetting results to ensure that no human testers are involved. All answers were consistent with the machine testers' parsing results in Table 6, verifying our previous findings and indicating that our attack can bypass machine testers by predicting the differences between machines and humans.

Then we further performed end-to-end attacks to demonstrate whether our attacks can bypass the whole vetting process. To prevent our attacks from being discovered by human testers, we designed a personality test skill with seven questions from [47]. Each question has five options, which means that the skill has 5^7 paths. We then put the entry sentence in a specific path. In this way, a human tester's chance to find the entry sentence is very low. Our malicious content (e.g., collecting social security numbers) is hidden in the following intent, which can only be accessed by users passing the entry sentence. Here, we put the entry sentence in the path that the machine tester can assess (according to Section 4.2), which again verifies that our attacks can bypass machine testers. The details of the conversation are shown in Appendix A.6. Later, we uploaded the skill to Amazon and Google three times, and all of them got published. At last, we removed the skills after invoked them to verify that they work normally. In this process, no users' privacy was collected.

Does the Entry Sentences Confuse Users? Our entry sentences should not confuse users, and thus we conducted a human-subject study. We recruited 100 human users from MTurk to answer the question. All of them are from the United States, and their previous task acceptance rate is higher than 95%. Their ages range from 18 to 70. In particular, 66% were 18–35 years old, and 34% were 36–70. 61% of all the users were male. Each participant needed to answer the 30 questions produced by Amazon TTS (Table 7). The average time to complete the questionnaire was 15.4 minutes. The results showed that the vast majority of sentences could help distinguish human users from machine testers. The average ratio of obtaining user information from each sentence was 74.3% (81.6% for QMA, 67% for UDA, and 74.2% for CPA). The success rate of obtaining information from users seems to be mostly related to the sentence designed (from 57% to 99%). In other words, a well-designed entry sentence may cause little confusion to users.

Effectiveness against the SOTA Tool. In addition to evaluating our attacks on the VPA platforms, we also evaluated our attacks on the state-of-the-art tool, SkillExplorer [41], designed to explore skill behaviors. If this tool cannot generate the same voice commands as humans for our entry sentences, it cannot discover our hidden malicious behaviors. We submitted our skills to SkillExplorer and obtained feedback. Twenty of the thirty entry sentences in Table 7 disturb the parsing process of SkillExplorer successfully, and ten (all are QMA) failed. Although no sentences of QMA in Table 7 work, we can achieve a 100% success rate by changing the sentence through R1 (using synonyms) and R4 (adding meaningless punctuation) in Table 5.

6 COUNTERMEASURE

To defeat the proposed attacks, we proposed countermeasures based on linguistic knowledge. Then we evaluated the effectiveness.

6.1 Command Extraction: Defend against QMA

For tools based on natural language parsing, the main challenge to defend against QMA is to extract commands when the sentence structure is disrupted by meaningless punctuation. We notice that although this attack changes the sentence structure, it does not affect the human users' understanding. The reason is that NLP techniques rely on punctuation to parse a sentence, while human users rely on the time intervals between words and sentences, which explains that a well-designed QMA sentence will not affect their answers.

To defend against this attack, we ignore the punctuation in the attack sentence and reorganize it. In detail, we experimentally get each punctuation's pause time and divide the punctuation into three categories: long-pause punctuation, short-pause punctuation, and no-pause punctuation. For example, a period has a long pause while a comma has a short pause (see Appendix A.7 for more details). Then we remove all non-pause punctuation in the sentence since users cannot perceive their existence. Next, we extract the content between every two pauses and filtered the irrelevant items according to the rules in Appendix A.8. Finally, we use SkillExplorer to remove the irrelevant words in each content to obtain the commands.

6.2 Determiner Detection: Defend against UDA

The difficulty in defeating UDA is to judge whether the answer is the noun itself (called *common nouns*) or its referent (called *proper nouns*). Common nouns represent a type of entity (such as a city), while proper nouns represent a single entity (such as Seattle) [75]. After analyzing many relevant sentences, we found that since the referent of proper nouns is specific, the answer is more likely to be proper nouns when the scope of common nouns is limited (i.e., determiner [74]). For example, the answer to the sentence "You can say name" is more likely to be the common noun "name," but the answer to "you can say a name" is more likely to be a proper noun like "Tom." So we first check whether the command is a noun with a determiner. If not, we will answer the command itself; otherwise, we will answer the corresponding proper noun. One may think of answering both the two answers. However, attackers can distinguish between machines and humans according to the answer history. So it is essential to answer the correct one.

6.3 Co-reference Resolution: Defend against CPA

When the command is only a pronoun (i.e., CPA), we try to use the reference resolution technique to find its referent. Specifically, we use the Neuralcoref library [44], the state-of-the-art co-reference resolution based on neural nets and spaCy. Besides, to restore the context, we spliced each sentence with its parent. The recovered referent is the command we need.

6.4 Evaluation

To evaluate the effectiveness of our defenses, we use 108 sentences in Section 4.3 and 30 sentences in Table 7 as the test set (including 118 for QMA, 10 for UDA, and 10 for CPA). The results are as follows: our defense correctly parsed all 138 attack sentences and no sentence was wrongly parsed, which indicates that our defense method has a good performance against the attacks.

We also checked whether our defense methods would cause normal sentences to be parsed incorrectly. We randomly selected 100 sentences in skills from all categories, which contained 117 valid commands. Among the 117 commands, we correctly generated 105 commands. Then we compare the results with SkillExplorer, which correctly generated 98 commands[9], which indicates that the defense method does not impact the sentence parsing. We also recorded the average time spent in the defense. It was around 14.95 milliseconds on an ordinary laptop (3.6GHz CPU, i7-7700), which does not impose significant overhead on the vetting.

7 RELATED WORK

Attacks on Skills. Prior studies suggested that attackers could hijack the voice commands of a legitimate skill based on misinterpretation issues. Kumar et al. [50] and Zhang et al. [91] demonstrated that a malicious skill could use an invocation name with a similar pronunciation of a legitimate skill to divert the users' requests to

[9]We regard the noun itself as the wrong answer to the UDA sentences.

itself. Zhang et al. [92] also found that a malicious skill could mimic speaker systems' sounds by pretending to have exited. They presented a linguistic-model-guided fuzzing tool to automatically detect malicious skills with obfuscated invocation names to resolve such issues. Meanwhile, Guo et al. [41] developed an automated interactive system, called *SkillExplorer*, to explore the skill content. More recently, Cheng et al. [24] found that the current VPA platforms' skill certification process is not trustworthy and does not strictly enforce policy requirements. Hu et al. [43] demonstrated that the vetting process is insufficient to identify the authentication issues in the endpoints. However, previous research did not explain why malicious skills can pass the vetting process, which helps protect the markets better.

Attacks on Smart Speakers. Some related work has shown that smart speakers can be remotely attacked through carefully designed audio. Vaidya et al. [72] proposed an approach for generating audio perceived as noise by humans but audio commands by smart speakers. Carlini et al. [17] designed a general attack procedure that works with any modern voice recognition system. They showed that hidden voice commands that humans cannot understand could be constructed with significant knowledge of the speech recognition system. Zhang et al. [89] used inaudible audio to inject the command into smart speakers without human awareness. Meanwhile, Blue et al. [14] designed a detection mechanism to differentiate humans from electronic speakers and avoid injection attacks from different electronic speakers. Yuan et al. [87] found that voice commands can be stealthily embedded into songs and designed the first practical adversarial attacks against ASR systems. Abdullah et al. [1] generated attack audio based on the feature vectors from signal processing algorithms, which works against a wide range of speech detection and speaker recognition systems. Chen et al. [23] proposed an approach for automatically embedding a set of commands into a song. Human users would perceive it as a typical song without feeling any difference. Abdullah et al. [2] presented the first threat model framework for reasoning more broadly about existing works in the adversarial space against VPSes. Yan et al. [86] leveraged ultrasonic guided waves to implement a new hidden attack that can enable multiple dialogues over a longer distance. Sugawara et al. [69] more recently proposed a novel signal injection attack on microphones by aiming an amplitude-modulated light at the microphone's aperture. Such attacks exploit smart speakers' weakness; that is, they cannot distinguish the voice owner and target smart speakers. Our attack is targeted at smart speakers' vetting processes and helps strengthen protections against potentially dangerous skills.

Detection of Malicious Apps. Many studies have focused on malicious app detection. The most popular approach is static [30, 33, 34, 54, 65, 70, 81, 94] and dynamic analysis [29, 62, 63, 80, 85]. Some studies used hybrid approaches [13, 35, 46, 55, 79, 82, 93, 96], combining static and dynamic analysis, due to the limitations of the static (e.g., false positives) and dynamic (e.g., false negatives) analysis. Moreover, some authors recognized that machine learning techniques could assist in the vetting process [12, 84, 90, 97]. Some studies focused on the program similarity analysis for repackaged application detection, including code similarity [20, 26, 42, 73, 83, 95], UI similarity [21, 88], and library similarity analyse [22, 52, 53]. These techniques mostly rely on access to the code or runtime environments. However, unlike mobile applications usually submitted to the markets with their code, skills are services without code available to the markets. Hence, existing studies on program analysis cannot directly be transferred into the field of skill vetting.

8 CONCLUSION

This work investigated the vetting process in popular skill markets, namely Amazon and Google, through carefully designed skills. We obtained intriguing findings from the probing results (e.g., machine or human testers and strategies of exploring skills), especially the vetting process's weakness. We proposed herein three attacks that help malicious skills successfully bypass the vetting process based on our findings. To mitigate such attacks, we also designed several linguistic-knowledge-based defense methods.

Interestingly, we found that the strategies of vetting change after we reported to Amazon/Google. For example, during our probing in October 2020, we found that Amazon's machine testers no longer answered wh-questions, and the vetting time had become much longer (about one week). Maybe they want to detect our Turing test questions. In Google, they try not to use machine testers. Note that, although the strategies change, the idea of our exploration method still works. Our attacks are also valid because these attacks exploit the weaknesses of NLP techniques, which cannot be easily fixed by merely changing the strategies.

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers and Dr. Yuqi Cui for their insightful comments. IIE authors are supported in part by the National Key R&D Program of China (2020AAA0105200), NSFC U1836211, Beijing Natural Science Foundation (No.JQ18011), National Top-notch Youth Talents Program of China, Youth Innovation Promotion Association CAS, Beijing Academy of Artificial Intelligence (BAAI) and a research grant from Huawei.

REFERENCES

[1] Hadi Abdullah, Washington Garcia, Christian Peeters, Patrick Traynor, Kevin RB Butler, and Joseph Wilson. 2019. Practical hidden voice attacks against speech and speaker recognition systems. In *NDSS*.

[2] Hadi Abdullah, Kevin Warren, Vincent Bindschaedler, Nicolas Papernot, and Patrick Traynor. 2020. SoK: The Faults in our ASRs: An Overview of Attacks against Automatic Speech Recognition and Speaker Identification Systems. arXiv:2007.06622 [cs.CR]

[3] Amazon. 2018. Amazon Mechanical Turk. https://www.mturk.com/.

[4] Amazon. 2021. Alexa Skills Kit Command Line Interface (ASK CLI) Overview. https://developer.amazon.com/en-US/docs/alexa/smapi/ask-cli-intro.html.

[5] Amazon. 2021. Certification Requirements in Alexa. https://developer.amazon.com/en-US/docs/alexa/custom-skills/certification-requirements-for-custom-skills.html.

[6] Amazon. 2021. Create Intents, Utterances, and Slots. https://developer.amazon.com/en-US/docs/alexa/custom-skills/create-intents-utterances-and-slots.html.

[7] Amazon. 2021. Functional Testing for a Custom Skill. https://developer.amazon.com/en-US/docs/alexa/custom-skills/functional-testing-for-a-custom-skill.html.

[8] Amazon. 2021. Manage the Skill Session and Session Attributes. https://developer.amazon.com/en-US/docs/alexa/custom-skills/manage-skill-session-and-session-attributes.html.

[9] Amazon. 2021. Request and Response JSON Reference. https://developer.amazon.com/en-US/docs/alexa/custom-skills/request-and-response-json-reference.html.

[10] Amazon. 2021. Standard Built-in Intents. https://developer.amazon.com/en-US/docs/alexa/custom-skills/standard-built-in-intents.html.

[11] Amazon. 2021. Test and Submit Your Skill for Certification. https://developer.amazon.com/en-US/docs/alexa/devconsole/test-and-submit-your-skill.html.

[12] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. 2014. Drebin: Effective and explainable detection of android malware in your pocket.. In *Ndss*, Vol. 14. 23–26.

[13] Ravi Bhoraskar, Seungyeop Han, Jinseong Jeon, Tanzirul Azim, Shuo Chen, Jaeyeon Jung, Suman Nath, Rui Wang, and David Wetherall. 2014. Brahmastra: Driving apps to test the security of third-party components. In *23rd USENIX Security Symposium (USENIX Security 14)*. 1021–1036.

[14] Logan Blue, Luis Vargas, and Patrick Traynor. 2018. Hello, Is It Me You're Looking For? Differentiating Between Human and Electronic Speakers for Voice Interface Security. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. 123–133.

[15] Fabian Bräunlein and Luise Frerichs. 2019. Smart Spies: Alexa and Google Home expose users to vishing and eavesdropping. https://srlabs.de/bites/smart-spies/.

[16] BusinessWorldIT. 2019. 12 Best Artificial Intelligence Chatbots For Android. https://www.businessworldit.com/ai/best-ai-chatbots-for-android/.

[17] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden voice commands. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 513–530.

[18] Rollo Carpenter. 2011. jabberwacky - live chat bot. http://www.jabberwacky.com/.

[19] Rollo Carpenter. 2021. Cleverbot.com - a clever bot. https://www.cleverbot.com/.

[20] Kai Chen, Peng Liu, and Yingjun Zhang. 2014. Achieving accuracy and scalability simultaneously in detecting application clones on android markets. In *Proceedings of the 36th International Conference on Software Engineering*. 175–186.

[21] Kai Chen, Peng Wang, Yeonjoon Lee, XiaoFeng Wang, Nan Zhang, Heqing Huang, Wei Zou, and Peng Liu. 2015. Finding unknown malice in 10 seconds: Mass vetting for new threats at the google-play scale. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*. 659–674.

[22] Kai Chen, Xueqiang Wang, Yi Chen, Peng Wang, Yeonjoon Lee, XiaoFeng Wang, Bin Ma, Aohui Wang, Yingjun Zhang, and Wei Zou. 2016. Following devil's footprints: Cross-platform analysis of potentially harmful libraries on android and ios. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 357–376.

[23] Yuxuan Chen, Xuejing Yuan, Jiangshan Zhang, Yue Zhao, Shengzhi Zhang, Kai Chen, and XiaoFeng Wang. 2020. Devil's Whisper: A General Approach for Physical Adversarial Attacks against Commercial Black-box Speech Recognition Devices. In *29th USENIX Security Symposium (USENIX Security 20)*.

[24] Long Cheng, Christin Wilson, Song Liao, Jeffrey Young, Daniel Dong, and Hongxin Hu. 2020. Dangerous Skills Got Certified: Measuring the Trustworthiness of Skill Certification in Voice Personal Assistant Platforms. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, USA) *(CCS '20)*. Association for Computing Machinery, New York, NY, USA, 1699–1716. https://doi.org/10.1145/3372297.3423339

[25] Andrew Cohen. 2020. https://www.wearerockwater.com/blog/smart-speaker-boom-at-home-audio. https://www.wearerockwater.com/blog/smart-speaker-boom-at-home-audio.

[26] Jonathan Crussell, Clint Gibler, and Hao Chen. 2013. Scalable semantics-based detection of similar android applications. In *Proc. of ESORICS*, Vol. 13. Citeseer.

[27] George Dvorsky. 2015. 8 Possible Alternatives To The Turing Test. https://io9.gizmodo.com/8-possible-alternatives-to-the-turing-test-1697983985.

[28] dylanavalverde. 2018. A Brief History of Chatbots. https://pcc.cs.byu.edu/2018/03/26/a-brief-history-of-chatbots/.

[29] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. 2014. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)* 32, 2 (2014), 1–29.

[30] William Enck, Machigar Ongtang, and Patrick McDaniel. 2009. On lightweight mobile phone application certification. In *Proceedings of the 16th ACM conference on Computer and communications security*. 235–245.

[31] Linked Experiences. 2021. Jovo Framework. https://www.jovo.tech/framework.

[32] Explosion. 2021. Industrial-Strength Natural Language Processing. https://spacy.io.

[33] Yu Feng, Saswat Anand, Isil Dillig, and Alex Aiken. 2014. Apposcopy: Semantics-based detection of android malware through static analysis. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 576–587.

[34] Yu Feng, Osbert Bastani, Ruben Martins, Isil Dillig, and Saswat Anand. 2017. Automated synthesis of semantic malware signatures using maximum satisfiability. *NDSS* (2017).

[35] Xi Ge, Kunal Taneja, Tao Xie, and Nikolai Tillmann. 2011. DyTa: dynamic symbolic execution guided with static verification results. In *Proceedings of the 33rd International Conference on Software Engineering*. 992–994.

[36] Google. 2020. App Actions built-in intents. https://developers.google.com/assistant/app/reference/built-in-intents.

[37] Google. 2020. Conversation webhook format. https://developers.google.com/assistant/conversational/df-asdk/reference/conversation-webhook-json.

[38] Google. 2020. Dialogflow webhook format. https://developers.google.com/assistant/conversational/df-asdk/reference/dialogflow-webhook-json.

[39] Google. 2021. Custom intents. https://developers.google.com/assistant/app/custom-intents.

[40] Google. 2021. Policies for Actions on Google. https://developers.google.com/assistant/console/policies/general-policies.

[41] Zhixiu Guo, Zijin Lin, Pan Li, and Kai Chen. 2020. SkillExplorer: Understanding the Behavior of Skills in Large Scale. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 2649–2666. https://www.usenix.org/conference/usenixsecurity20/presentation/guo

[42] Steve Hanna, Ling Huang, Edward Wu, Saung Li, Charles Chen, and Dawn Song. 2012. Juxtapp: A scalable system for detecting code reuse among android applications. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 62–81.

[43] H. Hu, L. Yang, S. Lin, and G. Wang. 2020. A Case Study of the Security Vetting Process of Smart-home Assistant Applications. In *2020 IEEE Security and Privacy Workshops (SPW)*. 76–81. https://doi.org/10.1109/SPW50608.2020.00029

[44] Huggingface. 2021. NeuralCoref 4.0: Coreference Resolution in spaCy with Neural Networks. https://github.com/huggingface/neuralcoref.

[45] SimSimi Inc. 2019. SimSimi. www.simsimi.com.

[46] Yajin Zhou Xuxian Jiang and Zhou Xuxian. 2013. Detecting passive content leaks and pollution in android applications. In *Proceedings of the 20th Network and Distributed System Security Symposium (NDSS)*.

[47] Carl Gustav Jung. 2014. *Psychological types*. Routledge.

[48] BRET KINSELLA. 2019. Google Assistant Actions Total 4,253 in January 2019. https://voicebot.ai/2020/01/19/google-assistant-actions-grew-quickly-in-several-languages-in-2019-match-alexa-growth-in-english/.

[49] BRET KINSELLA. 2020. Alexa Skill Counts Surpass 80K in US, Spain Adds the Most Skills, New Skill Rate Falls Globally. 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/.

[50] Deepak Kumar, Riccardo Paccagnella, Paul Murley, Eric Hennenfent, Joshua Mason, Adam Bates, and Michael Bailey. 2018. Skill squatting attacks on Amazon Alexa. In *27th USENIX Security Symposium (USENIX Security 18)*. 33–47.

[51] Eugenia Kuyda. 2019. replika. https://replika.ai/.

[52] Li Li, Tegawendé F Bissyandé, Jacques Klein, and Yves Le Traon. 2016. An investigation into the use of common libraries in android apps. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 1. IEEE, 403–414.

[53] Menghao Li, Wei Wang, Pei Wang, Shuai Wang, Dinghao Wu, Jian Liu, Rui Xue, and Wei Huo. 2017. LibD: scalable and precise third-party library detection in android markets. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 335–346.

[54] Long Lu, Zhichun Li, Zhenyu Wu, Wenke Lee, and Guofei Jiang. 2012. Chex: statically vetting android apps for component hijacking vulnerabilities. In *Proceedings of the 2012 ACM conference on Computer and communications security*. 229–240.

[55] Riyadh Mahmood, Nariman Mirzaei, and Sam Malek. 2014. Evodroid: Segmented evolutionary testing of android apps. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 599–609.

[56] Taylor Martin. 2017. You can now use any Alexa skill without enabling it first. https://www.cnet.com/home/smart-home/amazon-echo-you-can-now-use-any-alexa-skill-without-enabling-it-first/.

[57] MIT. 2021. Avoiding Plagiarism - Paraphrasing. https://integrity.mit.edu/handbook/academic-writing/avoiding-plagiarism-paraphrasing.

[58] Pandorabots. 2020. Mitsuku. https://www.pandorabots.com/mitsuku/.

[59] Pandorabots. 2021. Bot A.L.I.C.E - Pandorabots. https://www.pandorabots.com/pandora/talk?botid=b8d616e35e36e881.

[60] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 311–318. https://doi.org/10.3115/1073083.1073135

[61] Neil J. Perkins and Enrique F. Schisterman. 2006. The Inconsistency of "Optimal" Cutpoints Obtained using Two Criteria based on the Receiver Operating Characteristic Curve. *American Journal of Epidemiology* 163, 7 (01 2006), 670–675. https://doi.org/10.1093/aje/kwj063 arXiv:https://academic.oup.com/aje/article-pdf/163/7/670/175717/kwj063.pdf

[62] Vaibhav Rastogi, Yan Chen, and William Enck. 2013. AppsPlayground: automatic security analysis of smartphone applications. In *Proceedings of the third ACM conference on Data and application security and privacy*. 209–220.

[63] Alessandro Reina, Aristide Fattori, and Lorenzo Cavallaro. 2013. A system call-centric analysis and stimulation technique to automatically reconstruct android malware behaviors. *EuroSec, April* (2013).

[64] Get Riddles. 2018. Get Riddles. 1000+ Riddles With Answers for Kids and Adults. https://www.getriddles.com/.

[65] Borja Sanz, Igor Santos, Carlos Laorden, Xabier Ugarte-Pedrero, Pablo Garcia Bringas, and Gonzalo Álvarez. 2013. Puma: Permission usage to detect malware in android. In *International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions*. Springer, 289–298.

[66] Lea Schönherr, Maximilian Golla, Thorsten Eisenhofer, Jan Wiele, D. Kolossa, and Thorsten Holz. 2020. Unacceptable, where is my privacy? Exploring Accidental Triggers of Smart Speakers. *ArXiv* abs/2008.00508 (2020).

[67] Artificial Solutions. 2020. Elbot the Chatbot. www.elbot.com.

[68] Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. , 4444–4451 pages. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14972

[69] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. 2020. Light commands: laser-based audio injection attacks on voice-controllable systems. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*. 2631–2648.

[70] Wei Tang, Guang Jin, Jiaming He, and Xianliang Jiang. 2011. Extending Android security enforcement with a security distance model. In *2011 International Conference on Internet Technology and Applications*. IEEE, 1–4.

[71] Purdue University. 2020. Paraphrasing // Purdue Writing Lab. https://owl.purdue.edu/owl/research_and_citation/using_research/quoting_paraphrasing_and_summarizing/paraphrasing.html.

[72] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. 2015. Cocaine noodles: exploiting the gap between human and machine speech recognition. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*.

[73] Timothy Vidas and Nicolas Christin. 2013. Sweetening android lemon markets: measuring and combating malware in application marketplaces. In *Proceedings of the third ACM conference on Data and application security and privacy*. 197–208.

[74] Wikipedia. 2021. Determiner. https://en.wikipedia.org/wiki/Determiner.

[75] Wikipedia. 2021. Proper noun and specific noun. https://en.wikipedia.org/wiki/Proper_noun_and_common_noun.

[76] Wikipedia. 2021. Stop words - Wikipedia. https://en.wikipedia.org/wiki/Stop_words.

[77] Wikipedia. 2021. Unicode - Wikipedia. https://en.wikipedia.org/wiki/Unicode.

[78] Bruce Wilcox. 2017. Rose demo - Brillig Understanding. http://brilligunderstanding.com/rosedemo.html.

[79] Michelle Y Wong and David Lie. 2016. IntelliDroid: A Targeted Input Generator for the Dynamic Analysis of Android Malware.. In *NDSS*, Vol. 16. 21–24.

[80] Chiachih Wu, Yajin Zhou, Kunal Patel, Zhenkai Liang, and Xuxian Jiang. 2014. AirBag: Boosting Smartphone Resistance to Malware Infection.. In *NDSS*.

[81] Dong-Jie Wu, Ching-Hao Mao, Te-En Wei, Hahn-Ming Lee, and Kuo-Ping Wu. 2012. Droidmat: Android malware detection through manifest and api calls tracing. In *2012 Seventh Asia Joint Conference on Information Security*. IEEE, 62–69.

[82] Mingyuan Xia, Lu Gong, Yuanhao Lyu, Zhengwei Qi, and Xue Liu. 2015. Effective real-time android application auditing. In *2015 IEEE Symposium on Security and Privacy*. IEEE, 899–914.

[83] Xiaojun Xu, Chang Liu, Qian Feng, Heng Yin, Le Song, and Dawn Song. 2017. Neural network-based graph embedding for cross-platform binary code similarity detection. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 363–376.

[84] Fabian Yamaguchi, Nico Golde, Daniel Arp, and Konrad Rieck. 2014. Modeling and discovering vulnerabilities with code property graphs. In *2014 IEEE Symposium on Security and Privacy*. IEEE, 590–604.

[85] Lok Kwong Yan and Heng Yin. 2012. Droidscope: Seamlessly reconstructing the OS and dalvik semantic views for dynamic android malware analysis. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*. 569–584.

[86] Qiben Yan, Kehai Liu, Qin Zhou, Hanqing Guo, and Ning Zhang. 2020. SurfingAttack: Interactive Hidden Attack on Voice Assistants Using Ultrasonic Guided Waves. In *NDSS*.

[87] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, XiaoFeng Wang, and Carl A Gunter. 2018. Commandersong: A systematic approach for practical adversarial voice recognition. In *27th USENIX Security Symposium (USENIX Security 18)*. 49–64.

[88] Fangfang Zhang, Heqing Huang, Sencun Zhu, Dinghao Wu, and Peng Liu. 2014. ViewDroid: Towards obfuscation-resilient mobile application repackaging detection. In *Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks*. 25–36.

[89] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 103–117.

[90] Mu Zhang, Yue Duan, Heng Yin, and Zhiruo Zhao. 2014. Semantics-aware android malware classification using weighted contextual api dependency graphs. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 1105–1116.

[91] Nan Zhang, Xianghang Mi, Xuan Feng, XiaoFeng Wang, Yuan Tian, and Feng Qian. 2019. Dangerous Skills: Understanding and Mitigating Security Risks of Voice-Controlled Third-Party Functions on Virtual Personal Assistant Systems. *2019 IEEE Symposium on Security and Privacy (SP)* (2019), 1381–1396.

[92] Yangyong Zhang, Lei Xu, Abner Mendoza, Guangliang Yang, Phakpoom Chinprutthiwong, and Guofei Gu. 2019. Life after Speech Recognition: Fuzzing Semantic Misinterpretation for Voice Assistant Applications.. In *NDSS*.

[93] Cong Zheng, Shixiong Zhu, Shuaifu Dai, Guofei Gu, Xiaorui Gong, Xinhui Han, and Wei Zou. 2012. Smartdroid: an automatic system for revealing ui-based trigger conditions in android applications. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*. 93–104.

[94] Min Zheng, Mingshen Sun, and John CS Lui. 2013. Droid analytics: a signature based analytic system to collect, extract, analyze and associate android malware. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 163–171.

[95] Wu Zhou, Yajin Zhou, Xuxian Jiang, and Peng Ning. 2012. Detecting repackaged smartphone applications in third-party android marketplaces. In *Proceedings of the second ACM conference on Data and Application Security and Privacy*. 317–326.

[96] Yajin Zhou, Zhi Wang, Wu Zhou, and Xuxian Jiang. 2012. Hey, you, get off of my market: detecting malicious apps in official and alternative android markets.. In *NDSS*, Vol. 25. 50–52.

[97] Ziyun Zhu and Tudor Dumitraş. 2016. Featuresmith: Automatically engineering features for malware detection by mining the security literature. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 767–778.

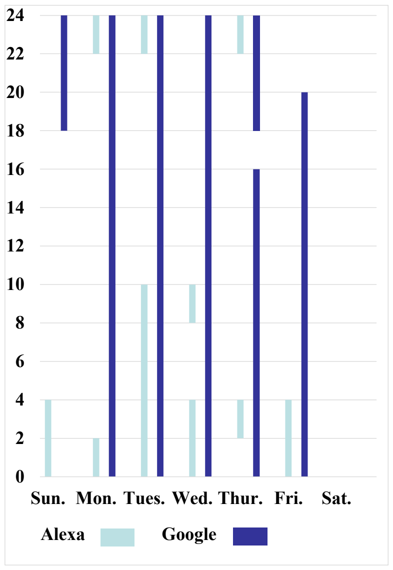
A Appendix

A.1 Answers of Chatbots

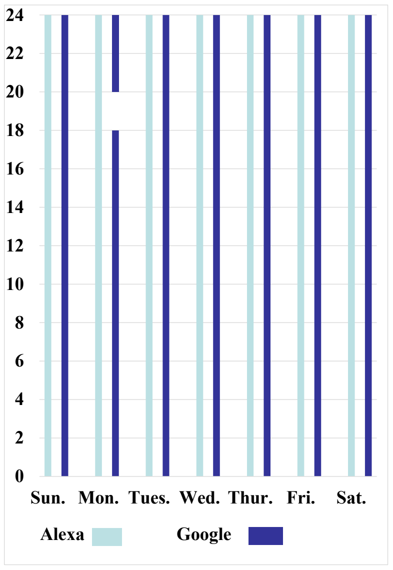
To see the answers, please visit:
https://github.com/DemystifyingVetting/DemystifyingVettingProcess

A.2 Vetting Time for Testers.

Figure 7 shows the time for human and machine testers to vet skills.



(a) Vetting time for human testers.



(b) Vetting time for machine testers.

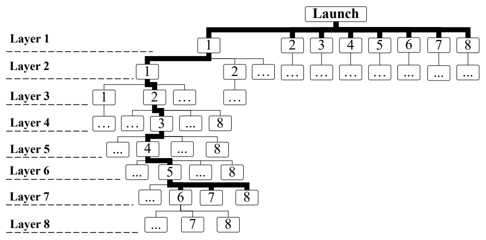
Fig. 7. Time for human and machine testers to vet skills. The rectangular bar within every two hours represents at least one tester vetted our skills in that period.

A.3 Interaction Model of Our Test Skill.

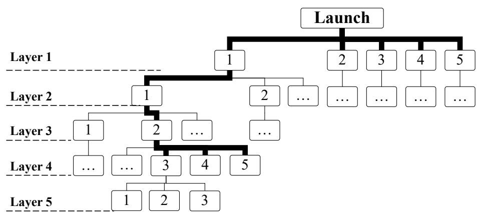
The interaction model of our test skill is shown in Table 8.

A.4 The Complement to Amazon's Traversal Strategy

The complement to Amazon's traversal strategy is shown in Figure 8.



(a) Traversal results for $n > 7$.



(b) Traversal results for no enough intents in kth layer.

Fig. 8. The complement to Amazon's traversal strategy. The bold lines indicate the traversal paths. We omitted some nodes from the figure due to space issues.

Table 8. Ten intents of the skill we built. Note that the sample utterance of the last three intents does not show in the responses of the *Launch* and *Help* intents, but can still be triggered by users.

Type	Name	Utterance	Response
Built-in intents	Launch		To get A, you can say A. To get B, you can say B. To get C. you can say C.[1]
	Help[2]	help	
	Stop	stop	Goodbye!
	Cancel	cancel	Cancel!
Custom Intents	VisibleA	A.	Got it.
	VisibleB	B.	Got it.
	VisibleC	C.	Got it.
	HiddenA	here is a hidden intent	You find me.
	HiddenB	I guess you can not find me	You find me.
	HiddenC	how can you find me	You find me.

[1] In Google, we put single quotation marks around the command just like "you can say 'A'."
[2] The Help intent is not a built-in intent in Google.

A.5 Special Case of Hidden Intent Being Checked

In very rare situations, the users' hidden intents will be checked. On Amazon's side, if we change the question in *Launch* or *Help* to a WH-question, such as "What's your name?" (must end with a question mark), one of the custom intents will be checked by the machine testers, even though the *Launch*/*Help* intents have not mentioned it. Similarly, the machine testers in Google will do the same if they cannot parse the question they received.

A.6 Details of the End-to-end Attack

Below are the details of the conversation in the end-to-end attack. Sentence (7) is the entry sentence, and (8) is the malicious sentence. The skill only replies (7) when a user gives a specific answer sequence (i.e., AABCD for Amazon and EEEEE for Google). Otherwise, it will reply (6) and end the session.

(1) Welcome to Extraversion Score Test! This free personality test is based on Carl Jung's and Isabel Briggs Myers' personality type theory. You will receive 5 questions in turn, and will receive the test results after completion. The first question is Am the life of the party. From very inaccurate to very accurate, you can say A, B, C, D, and E.
(2) Next question is Don't talk a lot. From very inaccurate to very accurate, you can say A, B, C, D, and E.
(3) Next question is Feel comfortable around people. From very inaccurate to very accurate, you can say A, B, C, D, and E.
(4) Next question is Keep in the background. From very inaccurate to very accurate, you can say A, B, C, D, and E.
(5) Next question is Start conversations. From very inaccurate to very accurate, you can say A, B, C, D, and E.
(6) Here is your test result. Your Extraversion score is 15. Individuals who score high on Factor I one are outgoing and social. Individuals who score low tend to be shut ins. (End the session)
(7) Here is your test result. Your Extraversion score is 15. Individuals who score high on Factor I one are outgoing and social. Individuals who score low tend to be shut ins. Next, to end the session, you can say

cancel, quit, stop, and exit. If you are interested in your score, you can say tell me more details if you want to know more about your personality. (Wait for reply)

(8) Hi, based on your test results, we suspect that you have a mild mental illness. To learn more about your situation, we need to get your social security number. What is your social security number?

A.7 Punctuation's Pause Time

To obtain each punctuation's pause time, we used AWS TTS service to convert a long text containing the numbers from one to one hundred and one into audio. Then we inserted specific punctuation between every two numbers and then converted it to another audio. By comparing the difference in the two audios' duration, we got each punctuation's pause time, as shown in Table 9. According to the pause time t, we divided the punctuation into three categories: long-pause punctuation ($t > 0.5$ s), short-pause punctuation ($0.1 < t \leq 0.5$) s, and no-pause punctuation ($t \leq 0.1$ s).

Table 9. Each punctuation's pause time.

Category	Name	Symbol	Time
Long-pause	Period	.	0.76s
	Question	?	0.70s
	Exclamation	!	0.75s
	Braces	{}	0.57s
Short-pause		{	0.43s
		}	0.42s
	Parentheses	()	0.43s
		(	0.43s
		)	0.42s
	Brackets	[]	0.43s
		[	0.43s
		]	0.42s
	Comma	,	0.43s
	Semicolon	;	0.43s
	Colon	:	0.44s
	Dash	-	0.43s
	Ellipsis	...	0.43s
No-pause punctuation	Apostrophes	‘’	0.02s
		‘	0.02s
		’	0.03s
	Quotations	“”	0s
		“	0s
		”	0s

A.8 Filtering Rules for Command Extraction

Since a sentence may contain much irrelevant content (non-command content), we propose two filtering rules based on prior knowledge to reduce irrelevant content's impact. First, since commands are placed after key verbs in most cases, we only extract the content after each key verb (e.g., "say," "tell," and "ask"). Second, as for the case where there is no conjunction, we judge whether multiple contents are parallel commands through sentence structure similarity. We find that when multiple contents are with similar structures (parallelism sentence), users can know that they are parallel commands even without conjunctions. Specifically, we use the spaCy library [32] to do tokenization and part-of-speech (pos) tagging on each content. Starting from the first token, we compare the pos tags of each token in each content. When two or more consecutive tags are the same, we regard these sentences as structure-similar. For example, in the sentence "You can say: Give me an attraction. Tell me about Hamilton Wenham. Tell me the top five things to do. What would you like to do," there are four contents, "give me an attraction," "tell me about Hamilton Wenham," "tell me the top five things to do," and "what would you like to do." The first tag in the first three content is "VB," while that in the last content is "WP." So we regard the last one as irrelevant.

A.9 Supplying Documents for Surveys

To see the documents for survey, please visit:
https://github.com/DemystifyingVetting/DemystifyingVettingProcess