Semantic-Fuzzing-Based Empirical Analysis of Voice Assistant Systems of Asian Symbol Languages

Jian Mao, Ziwen Liu, Qixiao Lin, and Zhenkai Liang, *Member, IEEE*

***Abstract*—Recently, smart voice assistants (VAs) are widely deployed to provide control services via voice commands in IoT systems, e.g., smart home, industrial IoT systems, etc. However, due to the complexity of the application environment and the diversity of voice commands, more and more attacks against VAs cause severe security problems. As voice development platforms allow third-party voice skills to be accessed, adversaries are able to obtain users' private information by squatting attacks using confusing names. The existing work studied the exploitability of semantic misinterpretation in VA systems on phonetic languages such as English. However, due to the semantic structural difference between phonetic English and symbol-based Asian languages, such as Chinese, the linguistic-model-guided fuzzing tool proposed by the previous work is insufficient to conduct semantic analysis on the VAs of Asian Languages. In this article, we conduct a systematic analysis to evaluate the feasibility of voice misinterpretation attacks to typical Asian language VAs through semantic fuzzing. We develop Harmony-Fuzzer, the semantic fuzzing tool that the fuzzing process is under the guidance of fuzzing rules abstracted from phenomena of speech errors, disfluency, or semantically similar expressions in Chinese corpus. We use Bayesian networks to formulate fuzzing models statistically so that the fuzzing space can be controlled by the probability of fuzzing processing. We use our results to test VAs and design malicious skills to empirically verify the feasibility of squatting attacks. We found that squatting attacks on Chinese VAs are feasible when attackers leverage some linguistic phenomena delicately.**

***Index Terms*—Deep learning security, Internet of Things, IoT security, voice assistant (VA).**

I. Introduction

VOICE user interface (VUI), a human–computer interaction mechanism, understands user commands through voice recognition and uses voice to respond. Based on VUI, voice assistant (VA), an intelligent speech interaction system, is widely used in smart industrial and smart home scenarios, such as Google Assistant and Tmall Genie. VA-enabled smart devices carry out user commands, such as Internet searching, App activation, and device remote control, by recognizing intents in the user's voice [1], [2]. Since there is no need for users to type commands or perform physical interactions with devices, VAs are replacing traditional human–computer interface and becoming an ideal solution of humanized interactions. Currently, a variety of well-established voice service systems, such as Google Assistant, Amazon Alexa, and AliGenie, provides voice (control) services for smart devices, industrial equipment [3], etc.

The main component of a VUI-based VA system includes automatic speech recognition (ASR), natural language understanding (NLU), text to speech (TTS), etc. While VAs make human–machine interaction more convenient and natural, the complexity of the voice interaction environment and the diversity of human languages bring potential security risks to the users. Security issues of VAs include two aspects, in recognizing voice and in understanding intent, causing VAs to be threatened by *voice recognition attacks* and *voice misinterpretation attacks*. Voice recognition attacks leverage the insufficiency of user identity authentication in ASR to make malicious commands be recognized by VAs, including signal manipulation-based attacks, obfuscation-based attacks, and adversarial example-based attacks [4]. Voice misinterpretation attacks disturb the process of intent classification in NLU to trigger malicious operations [5]. These attacks may lead to security threats, such as denial of service, privacy leakage, and phishing. As functions of smart home VAs get improved and services are more diverse, for example, users can use VAs to shop or pay phone bills, attacks on VAs will cause more serious security consequences.

Adversaries carry out voice misinterpretation attacks by exploiting the weakness that users lack knowledge about the *skill* they are using, where a skill is a set of voice interaction procedures and service processing logic, i.e., third-party *apps* that offer a variety of services that the VA itself does not provide. Adversaries can design a malicious skill with an invocation name similar to a legal skill and wait for users to open their skill by mistake, which is called *squatting attack*. Through this attack, adversaries can obtain user's privacy information or provide misleading information.

Manuscript received March 26, 2021; revised June 20, 2021 and August 12, 2021; accepted September 4, 2021. Date of publication September 20, 2021; date of current version June 7, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62172027; in part by the Beijing Natural Science Foundation under Grant 4202036; in part by the National Key Research and Development Program of China under Grant 2020YFB1005601; in part by the National Natural Science Foundation of China under Grant U1733115 and Grant 61871023; and in part by the National Research Foundation (NRF) of Singapore through its NSoE DeST-SCI Programme under Grant NSoE_DeST- SCI2019-0006. *(Corresponding author: Jian Mao.)*

Jian Mao and Ziwen Liu are with the School of Cyber Science and Technology, Beihang University, Beijing 100083, China (e-mail: maojian@buaa.edu.cn).

Qixiao Lin is with the School of Cyber Science and Technology, Beihang University, Beijing 100083, China, and also with the School of Computing, National University of Singapore, Singapore 117417.

Zhenkai Liang is with the School of Computing, National University of Singapore, Singapore 117417.

Digital Object Identifier 10.1109/JIOT.2021.3113645

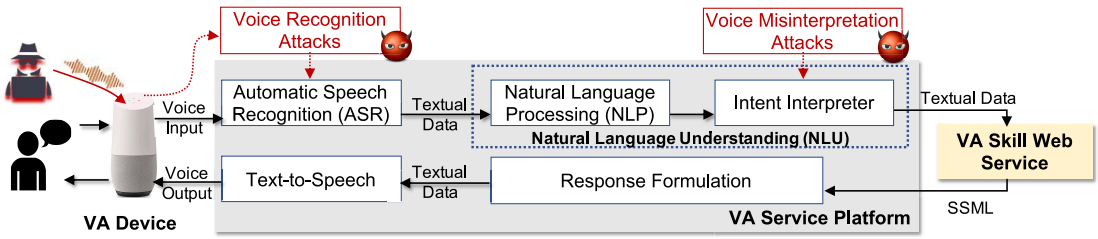

Fig. 1. VUI-based VA system and threats.

Researchers have worked on the exploitability of semantic misinterpretation in VAs on phonetic languages (such as English). Compared to phonetic languages, character-based Asian symbol languages (such as Chinese) are even more vulnerable to VA attacks. As the word in Asian symbol languages is composed of characters and there are dozens of characters under the same pronunciation, the distance in pronunciation between words in Asian symbol languages is much smaller on average. Moreover, the flexibility of spoken Asian language and various types of slips of the tongue make it difficult to use uniform rules to describe speech errors, which inevitably brings challenges to VA fuzzing test. As far as we know, there has been little discussion about squatting attacks on Asian-language-driven VAs and fuzzing rules extracted in others related works on English VAs are unavailable in this scenario. For example, previous research in English-driven VA semantic fuzzing only considered limited grammar-level linguistic information. Thus, the existed fuzzing rules are insufficient in expressing linguistic phenomena that are relevant to dependencies between words in Asian symbol languages.

In this article, we conduct a systematic analysis to evaluate the feasibility of voice misinterpretation (e.g., squatting) attacks to typical Asian language-driven VAs, e.g., Tmall Genie, through semantic fuzzing. We develop Harmony-Fuzzer, the semantic fuzzing tool of which the fuzzing process is under the guidance of 190 fuzzing rules abstracted from phenomena of speech errors, disfluency, or semantically similar expressions in Chinese corpus. Bayesian networks (BNs) are used to formulate fuzzing models statistically so that the fuzzing space can be controlled by the probability of fuzzing processing.

In the evaluation, we utilize 17 177 fuzzing commands generated by Harmony-Fuzzer for 100 sampled real commands. As a result, 12 927 fuzzing commands are misinterpreted by VAs, and 97 (out of 100) sampled commands are potentially vulnerable to squatting attacks. Furthermore, we design malicious skills to empirically verify the feasibility of squatting attacks. By leveraging some linguistic phenomena delicately, we successfully conduct squatting attacks on VAs in the Chinese language. Based on the evaluation, we summarize security problems of VAs in recognizing speech and understanding intent, and present defense suggestions.

Contribution: In summary, we make the following contributions in this article.

1) We analyze the mechanism of how VAs recognize user voices and understand user intents, and evaluate the feasibility of voice misinterpretation (e.g., squatting) attacks to typical Asian language-driven VAs.
2) We develop Harmony-Fuzzer, the semantic fuzzing tool whose fuzzing process is under the guidance of fuzzing rules abstracted from phenomena of speech errors, disfluency, or semantically similar expressions in Chinese corpus. To effectively reduce the space for fuzzing, we introduce BN formulation to calculate the probability of fuzzing processing.
3) We take fuzzing results of real commands to test VAs and design malicious skills to empirically verify the feasibility of squatting attacks. We analyze security problems in voice recognition and intent understanding of VAs and put forward defense suggestions.

Article Organization: The remainder of this article is organized as follows. In Section II, we introduce the technical mechanism of smart VAs, explain attacks and their feasibility in detail, and further give the overview of our approach. We demonstrate how to build our fuzzing model to generate fuzzing results for real voice commands in Section III. Specifically, we collect and process Chinese corpus to extract fuzzing rules and formulate the fuzzing model with BN. We conduct the evaluation and present the result of the fuzzing processing and squatting attack test in Section IV. We discuss related work in Section V, and finally, Section VI concludes this article.

II. Analysis on Security Risks of VAs

In this section, we discuss typical threats and attacks to VUI-based VA System.

A. VUI-Based VA System and Threats

Fig. 1 shows a typical structure of a VUI-based VA system [5], [6]. First, the user voice input is sent to the VA service platform and converted to the textual data through ASR. Then, in the NLU subsystem, natural language processing (NLP) performs text parsing to extract key features and then the intent interpreter analyzes the target service user asking. If a target service is found, the VA service platform will send a request with necessary parameters, such as intent ID and keywords in user command to the first-party or third-party service provider. Next, the Web service processes the request and replies the desired output that can be marked by the speech synthesis markup language (SSML) to indicate how the textual response is converted to synthesized speech. The VA service platform organizes the output and converts textual data into

audio data through TTS service. Finally, the voice output is played by the VA-enabled device.

1) Voice Recognition and Intent Understanding of VAs: In the voice interaction system of VAs, the voice recognition and the intent understanding refer to the process of converting user voice input into text, analyzing user intent, and extracting keywords (also called entities or slot values). The voice recognition and the intent understanding are realized through two essential techniques, respectively: 1) ASR and 2) NLU.

ASR: ASR is a machine-based, independent spoken speech decoding and translation process [7]. A typical ASR system analyzes the received voice data with a specific algorithm according to a preestablished linguistic model, and finally, gets the output that is usually in the text form [8].

NLU: NLU performs the semantically linguistic analysis to enable computers to understand the natural language. In VAs, the process of NLU can be summarized into two parts: 1) NLP and 2) intent interpreter [5]. First, NLP techniques, such as part-of-speech (POS) tagging, named entity recognition (NER), and coreference resolution (COREF) [9], [10], are used to analyze grammatical features of the text. The intent interpreter determines the semantic intent by matching the grammatical data with a prebuilt intent classification tree.

Slot Filling: The process of intent understanding also involves *slot filling*. A *slot* is a parameter associated with an intent. For instance, when a user asks "what is the weather in Beijing today," "today" and "Beijing" are parameters. They can be abstracted into two different categories, date and location, which are slots of a "check weather" intent. The slot filling means to fill in required slots so that user intent can be transformed into a clear command. Usually, labels are used to associate slots with intents, so the slot filling can also be seen as a process of label allocation [11].

Fuzzing Match: In addition, the *fuzzing match* in the speech recognition and the intent understanding can introduce the tolerance to various user voice commands. For example, with the fuzzing match, voice commands that have speech errors, synonym substitutions, or synonymous expressions can still be matched to target commands. Zhang *et al.* [5] showed thatd compared to Amazon Alexa, Google Assistant using fuzzing match is more tolerant of speech errors (such as wrong use of tense, singular, and plural) in user commands. A reasonable design of the fuzzing match algorithm can mitigate the impact of speech errors on understanding user intents, and improve the robustness of VAs.

2) Skill-Based Services and VA Ecosystem: Third-party voice services can be accessed by users as skills through the skill calling function provided by the native voice service platform of VAs. *Skill* refers to a set of voice interaction procedures and service processing logic. Taking Tmall Genie (the VA on device) and AliGenie (the platform) as an example, the sequence flow of calling a third-party skill is shown in Fig. 2. User's voice command is uploaded to AliGenie as audio via Tmall Genie. The platform uses ASR and NLP to parse the user's command and matches it to the corresponding skill and intent. The parsed parameters are sent to the service interface attached with the target skill provided by the third-party developer through an HTTP(S) request. The third-party service

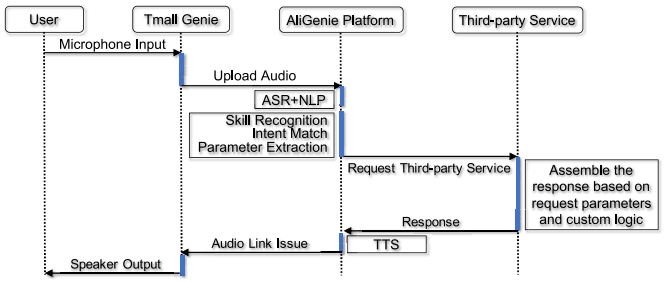

Fig. 2. Sequence flow of invoking third-party skills.

receives the request, executes service logic, and assembles a response. After the platform receives the result returned by the third-party service, it uses TTS to synthesize audio and sends the link of audio to the Tmall Genie device. The device broadcasts the issued audio to the user, and a round of dialogue is completed.

To create a skill, the third-party service provider must set the content of the skill on the platform, such as the skill invocation name, intents, entities, and the response logic. Before the skill is released on the skill market, it must pass the certification process in which the platform tests the skill to verify its function and security. Published skills can be accessed by smart device users via calling the invocation name without installation.

B. Attacks on the VA System

Typically, there are two types of attacks on the VA system: 1) voice recognition attacks and 2) voice misinterpretation attacks.

1) Voice Recognition Attacks: Voice recognition attacks include signal manipulation-based attacks, obfuscation-based attacks, and adversarial example-based attacks.

In signal manipulation-based attacks, malicious commands played by speakers in an open environment can start VAs even without any permissions. For example, the cartoon South Park mischievously triggered Amazon Echo VAs to fill viewers' Amazon shopping carts with risqué items by inserting voice commands in character's lines [12]. More hidden attacks make signals with frequencies beyond the range of human hearing be recognized by ASR [13], [14].

Obfuscation-based attacks [15] utilize the weakness of feature extraction in ASR to craft special voice signals that contain acoustic characteristics of malicious commands. Since the voice signal cannot be understood by humans, an adversary can control the device without the consent of users.

In adversarial example-based attacks [16], an unrelated voice signal can be recognized by ASR as a specific malicious command by continuously modifying the original audio according to the target command with slight disturbances, while users cannot tell the difference.

2) Voice Misinterpretation Attacks: Users usually organize their commands based on command templates provided by voice service developers. However, attackers can still leverage those commands that are not able to match correct skills because of systematic errors of VAs (caused by ASR when

a correct voice signal is received) [17] or speech errors, disfluencies, and synonymy in user commands [5] to launch malicious skills, which is called the *voice misinterpretation attack*. Voice misinterpretation attacks occur in the process of intent classification, as shown in Fig. 1. The impact of this attack would decrease as VAs' capabilities of voice processing and intent analysis are improved. Compared with voice recognition attacks, voice misinterpretation attacks are not limited by the physical distance between voice sources and victim devices and can be carried out on a large scale. Therefore, we could focus on voice misinterpretation attacks and further analyze their feasibility.

Specifically, the feasibility of voice misinterpretation attacks is manifested in the following aspects.

Linguistic Phenomena, Such as Pronunciation Confusions and Speech Errors, Are Common in User Voice Commands: Due to the existence of homophones, phonetic confusions, and compound words, speech recognition errors are inevitable. For example, in Chinese, "密封" (seal) and "蜜蜂" (bee) have the completely same pronunciation ("mìfēng"), which may be wrongly identified in speech recognition. Kumar *et al.* [17] pointed out that the systematic recognition error rate of words is 12.7%, of which 33.3% are caused by confusing pronunciation. In addition, speech errors of users may also cause their commands not be correctly understood by VAs. People often make speech errors in their daily speech [18]. A study on English speech errors shows that most people make 7–22 errors per day [19] and the frequency will be higher for nonnative speakers, children, and people who are tired or stressed. Adversaries can utilize these common linguistic phenomena to design malicious commands and carry out squatting attacks.

Large Number of Development Platforms Provide Technical Support for Third-Party Voice Services: More and more voice service providers build development platforms, such as Amazon Alexa, Google Assistant, AliGenie, and XiaoAi development platforms [20]–[24], and share their capabilities of ASR, NLP, and TTS with developers to enrich their voice services by third-party skills. The Amazon skill market has 25 000 skills, and Google also has more than 1000 skills (called actions by Google) that are used in the Google Home system [25]. In contrast, the number of skills for Chinese smart home VAs is smaller. As of June 2020, there are 459 skills for Tmall Genie and more than 100 skills for XiaoAi. While development platforms facilitate developers to efficiently create voice skills, adversaries are also able to release malicious skills and wait for users to open them by mistake, which will threaten user equipment safety and privacy security.

Voice Squatting Attacks Cannot Be Detected by the Certification Process of Platforms: The lax check of skill invocation names in the certification process gives opportunities for voice squatting attacks (VSAs) and lead to security risks. Amazon allows different skills to use the same invocation name. Although Google does not allow using the same names, it allows invocation names to be similar or have similar pronunciations, such as "cat facts" and "funny cat facts," which are similar to "cat fact" [25]. In addition, Alexa and Google assistant identify target skills by matching the longest string in voice commands. Suppose an adversary uses a phrase containing the invocation name of a victim skill as the invocation name of the malicious skill. In that case, the malicious skill will hijack the victim skill when the user command includes the phrase. As the result presented by previous studies, the certification process of voice service development platforms did not take such attacks into consideration.

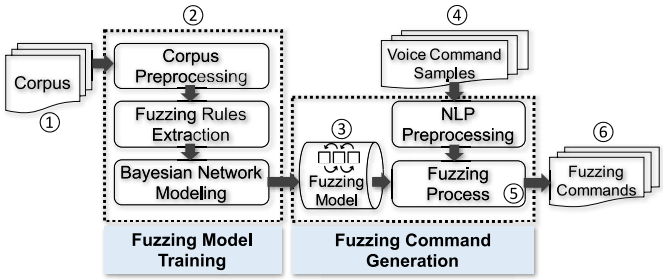

Fig. 3. Fuzzing tool architecture.

Lack of Mutual Awareness Between Users and Skills: On the one hand, the vulnerability of voice services comes from the lack of user identity authentication. It is difficult for VAs to tell whether the voice command comes from the user bound to it. On the other hand, users cannot distinguish whether the skill that they are speaking to is legal. The lack of mutual awareness enables adversaries to pretend to be one of two sides to carry out attacks [25].

III. Design and Implementation

Due to the diversity of user commands, smart home VAs have limited ability to accurately understand user intents. At the same time, skill development platforms are open to third-party developers, allowing adversaries to easily create malicious skills and obtain user privacy information through squatting attacks. To analyze the possibility of conducting squatting attacks on Chinese VAs, we use the method of fuzzing to generate possible commands to test the performance of Chinese smart home VAs in intent classification, and further analyze security problems based on test results.

A. Our Approach

The architecture of the fuzzing tool we develop is shown in Fig. 3, including fuzzing model training and fuzzing commands generation.

Fuzzing Model Training: To guide the fuzzing processing with linguistic rules, we collect a Chinese corpus (①) that contains linguistic phenomena, such as speech errors, disfluencies, and semantically similar expressions to build a fuzzing model. For the fuzzing model training (②), we utilize NLP tools to analyze the collected corpus and obtain fuzzing rules that describe the conversion between sentences through logical abstraction. Then, we use BN to formulate the fuzzing model and train weights of BN with a corpus of free speech in the real world.

Fuzzing Commands Generation: We conduct NLP preprocessing on collected command templates (④) given by developers and fuzz them (⑤) according to the fuzzing

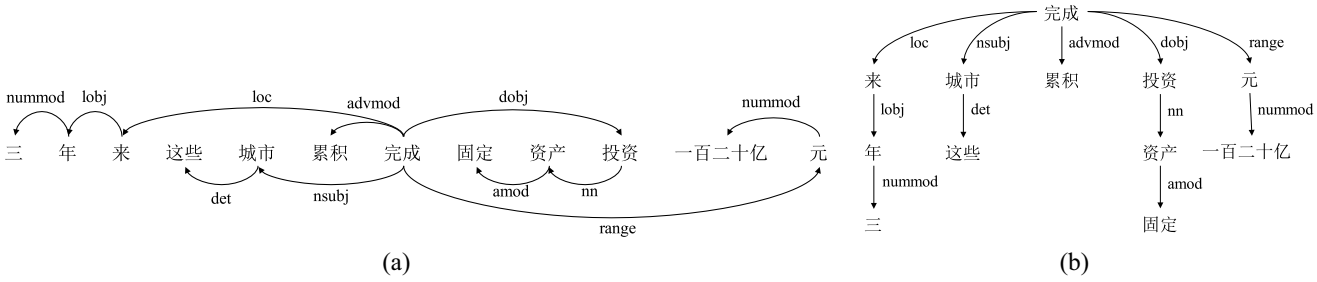

Fig. 4. (a) Dependencies between words and (b) syntactic parse tree.

model (③). The generated fuzzing commands (⑥) are used for the VA fuzzing test.

B. Fuzzing Model Training

The fuzzing model is built through the following steps: corpus collection and preprocessing; fuzzing rules extraction; and BN modeling.

1) Corpus Collection and Preprocessing: The Chinese corpus we collect can be divided into two parts: 1) speech errors and 2) expressions without speech errors. There are three levels of speech errors: 1) pronunciation errors; 2) vocabulary errors; and 3) grammar errors [26]. Expressions that are not speech errors but may lead to misunderstanding include *accents* that may cause voice recognition errors; *disfluencies* such as repeating or meaningless pausing, and *semantically similar words or phrases* such as synonyms, which may confuse the intent interpreter.

Before rules extraction, the collected corpus (Fig. 3 ①) is preprocessed to adequately annotate the features in terms of pronunciation, vocabulary, and grammar, which is required for abstracting linguistic fuzzing rules.

As Chinese is written using characters without spaces between words, sentences are need to be split into words through word segmentation. Each word should be marked with its POS tag indicating the function of the word in this sentence for further vocabulary and grammar error extraction. The pronunciation of each character is described by a triple, including an initial, a final, and a tone for pronunciation error extraction.

To obtain the grammatical structure of a sentence, we utilize the dependency parser to generate a syntactic parse tree that describes dependencies between words. For example, for this sentence, "sān niánlái zhèxiē chéngshì lěijī wánchéng gùdìng zīchǎn tóuzī yībǎi èrshí yì yuán" ("三年来这些城市累积完成固定资产投资一百二十亿元," i.e., "Over the past three years, these cities have invested 12 billion yuan in fixed assets"), dependencies between words and the syntactic parse tree are shown in Fig. 4(a) and (b), respectively. Each dependency between words is described by the governor word, the dependent word, and their relationship, which correspond to the starting node, pointing node, and edge in the syntactic parse tree. Types of dependencies, such as "dobj," "nsubj," "amod," and other grammatical relations, refer to the Stanford typed dependencies manual [27].

2) Fuzzing Rules Extraction: There are four forms of linguistic phenomena in the corpus we collected: 1) speech error; 2) accent; 3) disfluency; and 4) semantically similar expression.

Speech Error: Speech errors, or slips of the tongue, deviate from the speaker's intention, so accents, nonstandard Mandarin, and synonyms are not included in speech errors. According to the real scenario of using smart home VAs and the feasibility of describing fuzzing rules, speech errors related to the context, mental activity, and subconsciousness of users are not considered in this article. Speech errors can be classified into two types: 1) "phoneme" and 2) "combination of phoneme and meaning." In terms of initial consonants, final consonants, and tones, speech errors on phoneme can be divided into "single component errors" and "compound component errors." The "combination of phoneme and meaning" includes morphemes, words, and phrases according to the size of a linguistic unit. Meanwhile, behaviors of speech errors include replacing, exchanging, preplacing, postplacing, mixing, adding, and deleting [28]. In this article, we consider these phenomena and abstract fuzzing rules into pronunciation errors, vocabulary errors, and grammar errors.

Accent: An accent is an individual's characteristic tone or inflection. However, due to the limitation of the pronunciation annotation approach we choose, more complex features, such as loudness, pitch, or length, cannot be extracted. Therefore, we only consider the accent in terms of initial consonants, final consonants, tones, and their combinations. The rules of these accents have a similar logic to speech errors; hence, we use the same approach for rule extraction.

Disfluency: Common phenomena of disfluency are adding meaningless filled pauses, such as "en" ("嗯," i.e., "hum"), "jiùshì" ("就是," i.e., "just"), and "zhège" ("这个," i.e., "this"), or repeating some characters, words, or phrase.

Semantically Similar Expression: Semantically similar expressions of users are related to their habits of language. Except for synonym replacing, only some of the semantically similar phenomena that are easy to describe are considered in this article, such as omitting "de" ("的," i.e., "of") after an adjective, or adding "yíxià" ("一下," i.e., "once") after a verb.

To abstract these linguistic phenomena mentioned above into rules, we leverage the NLP annotation obtained from preprocessing to describe the precondition in which the phenomenon may happen and the way they happen. Mathematically, a fuzzing rule r is denoted as $r : s_r \Longrightarrow f_r$, where s_r is the precondition, f_r is the fuzzing function, and

TABLE I
EXAMPLES OF FUZZING RULE EXTRACTION

Phenomenon	Example	Description	Abstracted Fuzzing Rule
Speech Error	"wǒ méi bǎ huāpíng dǎ suì" ("我没把花瓶打碎") $\Longrightarrow$ "wǒ bǎ huāpíng méi dǎ suì" ("我把花瓶没打碎")	A verb's negative adverb that should be in other positions is moved before this verb.	$\exists x, y, x \in AD^1, y \in VV^2, dep(y, x, neg)$, $\nexists x$ before $y \Longrightarrow pop\_insert(x, x$ before $y)$
Accent	"Fāshēng shénme shìle" ("发生什么事了") $\Longrightarrow$ "Fāshēng sénme shìle" ("发生什么事了")	Pronounce the initial consonant "sh" as "s".	$\exists initial($"sh"$)$ $\Longrightarrow change($"sh", "s"$)$
Disfluency	"Wǒ xiǎng shì de" ("我想是的") $\Longrightarrow$ "Wǒ xiǎng, ǹg, shì de" ("我想，嗯，是的")	Insert a filled pause "嗯"("Hum").	$\forall$sentence $\Longrightarrow insert($"嗯",sentence$)$
Semantically Similar Expression	"Yóuxì guīzé" ("游戏规则") $\Longrightarrow$ "Yóuxì dè guīzé" ("游戏的规则")	If a noun has a noun modifier, insert "的" after the modifier.	$\exists x, y \in NN^3, dep(x, y, compound:nn)$ $\Longrightarrow insert($"的" after $y)$

[1] AD: adverb
[2] VV: general verb except for copulas, "有" ("have"), etc.
[3] NN: common noun except for temporal noun, proper noun, etc.

Algorithm 1: Fuzzing a Given Command

Input: Command T, Probability Threshold ζ, Fuzzing Rule Set $\mathcal{R} = \{r_1, ..., r_n\}$ where $r_i = (s_{r_i}, f_{r_i}, p_{r_i})$.
Output: Fuzzing Result Set $\mathcal{O} = \{O_1, ..., O_m\}$ where $O_j = (T_{O_j}, P_{O_j})$.

$\mathcal{Q} \leftarrow \{(T, 1)\}$;
$\mathcal{O} \leftarrow \emptyset$;
while $\mathcal{Q} \neq \emptyset$ **do**
  $Q \leftarrow selectFirst(\mathcal{Q})$;
  for $r_i \in \mathcal{R}$ **do**
    if T_Q *not satisfy* s_{r_i} **then**
      continue;
    end
    if $p_{r_i} * P_Q \geq \zeta$ **then**
      $T_{new} \leftarrow f_{r_i}(T_Q)$;
      $Q_{new} \leftarrow (T_{new}, p_{r_i} * P_Q)$;
      $\mathcal{Q}$ append Q_{new};
    end
  end
  $\mathcal{O}$ append Q;
  $\mathcal{Q}$ remove Q;
end

r indicates that if precondition s_r is satisfied then conduct fuzzing process f_r. The fuzzing policy of using rules to generate candidate commands will be detailed in Algorithm 1.

To describe a precondition s_r required by a fuzzing rule r, we take following conditions into account. Pronunciation conditions are presented as the existence of phonemes, i.e., "$\exists$ phoneme(x)" where x can be an initial, a final, a tone, or their combination. POS conditions specify the function of a target word in a sentence, which is denoted as "$\exists\, x, x \in \boldsymbol{A}$" where $\boldsymbol{A}$ is a word class represented as a POS tag. Dependency conditions describe the grammatical relationship between two words. Denoted as "$\exists$ dep(y, x, d)," dependencies are presented as the governor word y, the dependent word x, and their grammatical relationship d. Position conditions are presented as "$\exists$ x before/after y" to explain the order of words in a sentence. To include reverse conditions, "$\nexists$" is used instead of "$\exists$." If there is not precondition should be satisfied to perform a fuzzing processing, we denote s_r as "$\forall$ sentence."

The fuzzing functions f_r, presenting the processing approach of fuzzing, can be divided into six types: 1) *pronunciation modification*; 2) *vocabulary modification*; 3) *vocabulary replacement*; 4) *insertion*; 5) *deletion*; and 6) *repetition*. Pronunciation modification functions mainly focus on pronunciation errors in speech errors and accents, e.g., the function *change*() can be used to modify initials, finals, tones, or their combinations. The other five types of fuzzing function are used to generate vocabulary errors, grammatical errors, disfluency, and semantically similar expressions. All of them support optionally restricting the linguistic unit in which level the processing is performed (e.g., characters or words) and positional relationships between words, which can achieve a more complex processing. For example, one can specify the function *pop_insert*() to pop a word and insert it before or after another word in a sentence. Moreover, replacing morphemes or vocabulary in sentences with synonyms, similar words, or antonyms is achieved by searching preestablished libraries.

Table I provides examples for four linguistic phenomena with their literal descriptions and abstracted fuzzing rules. For instance, the grammar error in the sentence "wǒ bǎ huāpíng méi dǎ suì" ("我把花瓶没打碎," i.e., "I made the vase not broken") can be described as "a verb's negative adverb that should be in other positions is moved before this verb." Then, the fuzzing rule can be extracted like "$\exists x, y, x \in AD, y \in VV$, dep($y$, x, neg), $\nexists x$ before $y \Longrightarrow$ *pop_insert*(x, x before y)," which means that if a general verb (*VV*, which is different from other verbs such as copulas) y has a negative adverb (*AD*) x in the sentence and x is not before y, then a fuzzing sentence can be generated by function *pop_insert*(), which will pop the negative adverb x and insert it before the verb y.

3) Bayesian Network Modeling: BN has been successfully used in the field of statistical relational learning (SRL) and has solved many AI problems including NLU [5], [29]. In terms of NLP, Bayesian methods have been adopted for spell checking [30], which is similar to our fuzzing task. While spell checking aims at calculating the probability that a word is the correct formation of the given error word, we estimate the probability that an expression is the fuzzing result of the given origin expression.

However, fuzzing probability estimation differs from naive spell correction tasks because fuzzing processing may change the features of an expression. For example, a fuzzing function that inserts "zhège" ("这个," i.e., "this") before nouns will

provide the precondition for another fuzzing rule of mistakenly pronouncing the initial "zh" as "z." Therefore, we treat a single fuzzing processing as a state transition, and define a state as follows.

Definition 1 (State): A *state* X is that a possible expression includes a set of fuzzing characteristics $\boldsymbol{C} = \{c_1, \ldots, c_t\}$, where $c_k (k \in \{1, \ldots, t\})$ is a linguistic phenomenon created by a single fuzzing processing. Note that if a linguistic phenomenon characteristic is not included in $\boldsymbol{C}$, it means that the existence of this linguistic phenomenon in the possible expression is unknown.

We assume that the fuzzing rule set $\boldsymbol{R} = \{r_1, \ldots, r_n\}$ has a corresponding precondition set $\boldsymbol{S} = \{s_1, \ldots, s_n\}$. For a possible expression in state X, it is related to a Boolean variable V_{s_i} of whether the expression satisfies the precondition s_i required by the rule r_i or not. A state transition (a single fuzzing processing) can be defined as follows.

Definition 2 (State Transition): For the expression in state X, if V_{s_i} is true, then a fuzzing expression in state X' can be generated according to the rule r_i. Assuming that the fuzzing processing introduces a linguistic phenomenon characteristic c' into the fuzzing expression, the state X' means that the expression has a set of characteristics $\boldsymbol{C}' = \boldsymbol{C} \cup \{c'\}$. The *state transition* can be represented as $X \xrightarrow{r_i} X'$. Note that the transition is independent of the existing characteristics in a possible expression but related to the precondition s_i required by r_i.

Accordingly, we use a BN model to represent the transitions between states and the corresponding probabilities. Our definition of the BN model is as follows.

Definition 3 (Bayesian Network Model): A *BN model BN* is represented as a directed acyclic graph $G = (\boldsymbol{E}, \boldsymbol{V}, \boldsymbol{P})$, including the vertex set $\boldsymbol{V}$, the directed edge set $\boldsymbol{E}$, and the edge weight set $\boldsymbol{P}$. In our model, each vertex in the graph representing a state X. The directed edge $e_{XX'}$ refers to the state transition from one state X to another state X', and the weight $p_{XX'}$ serves as the corresponding transition possibility.

For instance, the state X represents that an expression includes a speech error of pronouncing the initial "zh" as "z"($\boldsymbol{C} = \{c_1\}$). After fuzzing based on fuzzing rule r, a new state X' represents the fuzzing expression that has the previous pronunciation error and a filled pause "嗯"("Hum") at the same time ($\boldsymbol{C}' = \{c_1, c_2\}$). Note that the specific location where the transition happens matters. For example, a sentence has two characters with an initial "zh," then pronunciation errors of them will be denoted as two different characteristics.

The transition from the state X to one of its next state X' has conditional probability

$$P(X'|X) = P\big(X'|V_{s_i} = \text{true}, X\big)P\big(V_{s_i} = \text{true}|X\big) \tag{1}$$

because if the precondition s_i is not satisfied, the transition under r_i will not happen, i.e., $P(X'|V_{s_i} = \text{false}, X) = 0$. As transitions are independent of existing characteristics $\boldsymbol{C}$, we have $p_{s_i} = P(X'|V_{s_i} = \text{true}, X) = P(c'|V_{s_i} = \text{true})$, which is the probability of getting characteristic c' based on rule r_i for any expression satisfying precondition s_i. Thus, we denote this probability as transition probability p_{s_i}, which is the weight that needs to be learned through observation.

Assume the probabilities of all the parent nodes of X' are known. Then, $P(X') = P(X'|X_{pi})P(X_{pi})$ where $X_{pi} \in \{X_{p1}, \ldots, X_{pk}\}$ is any of one-hop parent nodes of X', because various paths from initial vertex to X' is caused by different execution order of the same set of fuzzing functions. Also, the initial vertex of *BN* is a state X_0 representing any possible fuzzing expression of a given sentence, thus $\boldsymbol{C}_0 = \emptyset$ and $P(X) = 1$. Therefore, we can compute any $P(X')$ if probabilities of transition under the guidance of each fuzzing rule are known. Note that we calculate $P(V_{s_i} = \text{true}|X)$ in (1) based on specific expressions during fuzzing processing.

For transition probability estimation, the training Chinese voice data are required to contain a large number of free dialogues or discussions on free topics and is better to contain text form data with speech errors or disfluencies that have not been manually fixed. By counting how many times a transition occurs, its corresponding transition probability is estimated as the frequency.

C. Fuzzing Commands Generation

Fuzzing Algorithm: The basic structure of our fuzzing algorithm is shown in Algorithm 1. The input of the algorithm includes a given command T, a probability threshold ζ to cut off rare fuzzing processes, and the set of fuzzing rules $\mathcal{R} = \{r_1, \ldots, r_n\}$ where rule r_i is a triple including its precondition s_{r_i}, fuzzing function f_{r_i}, and transition probability p_{r_i}. The fuzzing function is considered to be a map from a given sentence T to its fuzzing result T', denoted as $T' \leftarrow f_{r_i}(T)$. The output of the algorithm includes a set of fuzzing result $\mathcal{O} = \{O_1, \ldots, O_m\}$ where the element O_j is two tuples of fuzzing sentence T_{O_j} and corresponding transition probability P_{O_j}.

Beginning from a given command $Q = (T, 1)$ in the unprocessed command set $\mathcal{Q}$, the algorithm generates fuzzing commands of Q according to r_i if T_Q satisfies the precondition s_{r_i} and the probability of the fuzzing processing is greater than the threshold ζ. The algorithm continues fuzzing the first unprocessed command Q in $\mathcal{Q}$, removing Q from $\mathcal{Q}$ after the processing, and generating the commands into output set $\mathcal{O}$ until $\mathcal{Q}$ is empty. By adjusting the threshold ζ or fuzzing times, the algorithm can control the number of fuzzing results to limit the searching space.

To explain how this algorithm works, take the sentence "打开台灯" ("please open the lamp") as an example and assume that there are two fuzzing rules: $r_1 : \exists\ x \in VV, \nexists$ "一下" after $x \Longrightarrow$ *insert*("一下" after x); $r_2 : \exists\ x \in NN, \nexists$ "这个" before $x \Longrightarrow$ *insert*("这个" before x), where *NN* represents nouns. The *BN* is demonstrated in the left-hand side of Fig. 5 in which the state of expressions is described as fuzzing features c_1 or c_2 it has. Since the sentence "打开台灯" (please open the lamp) has a verb "打开" ("open") without "一下" ("once") after it, it satisfies precondition s_1. Harmony-Fuzzer will insert "一下" after "打开" with a transition probability p_{s_1} (if $p_{s_1} \geq \zeta$) and generate a new sentence "打开一下台灯." Similarly, "打开一下这个台灯" will be generated with a transition probability $p_{s_1} * p_{s_2}$ (if $p_{s_1} * p_{s_2} \geq \zeta$). While if the probability threshold $\zeta \geq p_{s_1}$, X_1 and all its child nodes will be cut off.

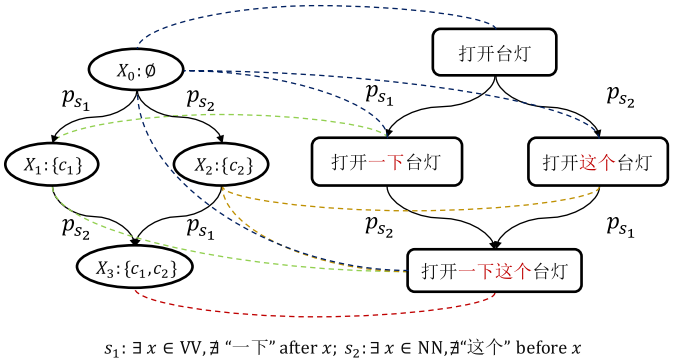

Fig. 5. Example of fuzzing a given command.

Algorithm Optimization: To solve some specific problems in the fuzzing processing, the algorithm is optimized based on the above basic structure.

1) *Reusing Preprocessing Results:* Each sentence is preprocessed by NLP tools before a single fuzzing process, which consumes a long time. To avoid preprocessing the same sentence repeatedly, the result of preprocessing will be stored and reused in the next fuzzing process on the same sentence to reduce the time cost.
2) *Processing Text and Pronunciation (i.e., Chinese Pinyin) in Parallel:* The text and pinyin are processed in parallel to avoid the mutual interference between the fuzzing process of pronunciation errors and other types during multistep fuzzing. It ensures that the text with pronunciation errors will still be processed according to the original word segmentation, POS tagging, and dependency analysis in the next step. Thus, except for pronunciation changes, the characteristics of the original vocabulary are completely retained, and the subsequent fuzzing process is not affected.
3) *Outputing Multiple Results for a Single Rule:* A fuzzing rule can get different fuzzing results for a same sentence, such as randomly inserting a filled pause "en" ("嗯," i.e., "Hum"). The fuzzing process for a sentence is required to return all possible fuzzing results for a rule.
4) *Merging the Same Results:* There must be some cases where the execution order of fuzzing rules changes but the fuzzing results are the same. The algorithm must merge the same fuzzing results, obtained under the same times of the fuzzing processing, to ensure the process follows the fuzzing model.

D. Implementation

Corpus Collection and Preprocessing: The Chinese corpus we collected includes speech errors [28], [31], [32], antonyms [33], accents [34], [35], synonyms [36], [37], semantically similar words [38], semantically similar sentences [39], and disfluencies [35].

We processed the collected corpus with Stanford Core NLP, pypinyin (a Python tool for phonetic notation of Chinese), and other tools to obtain four types of linguistic data: 1) words segmentation (Seg); 2) POS tags; 3) dependencies between words; and 4) pronunciations. Take this sentence, "shǐyòng gāi yǔjù cèshì yù chǔlǐ gōngjù de shūchū jiéguǒ" ("使用该语句测试预处理工具的输出结果," i.e., "Use this sentence to test the output of the preprocessing tool"), as an example, the processing output is shown in Table II.

TABLE II
EXAMPLE OF PREPROCESSING RESULTS

	Output
Seg	['使用', '该', '语句', '测试', '预', '处理', '工具', '的', '输出', '结果']
POS	['VV', 'DT'[1], 'NN', 'VV', 'JJ'[2], 'NN', 'NN', 'DEG'[3], 'NN', 'NN']
Dependencies[5]	[['使用', 'VV', 0], ['语句', 'NN', 2], 'dobj'][4] [['使用', 'VV', 0], ['测试', 'VV', 3], 'conj'], [['语句', 'NN', 2], ['该', 'DT', 1], 'det'], [['测试', 'VV', 3], ['结果', 'NN', 9], 'dobj'], [['工具', 'NN', 6], ['预', 'JJ', 4], 'amod'], [['工具', 'NN', 6], ['处理', 'NN', 5], 'compound:nn'], [['工具', 'NN', 6], ['的', 'DEG', 7], 'case'], [['结果', 'NN', 9], ['工具', 'NN', 6], 'nmod:assmod'], [['结果', 'NN', 9], ['输出', 'NN', 8], 'compound:nn']]
Pronunciation	[[['sh', 'i', '3'], ['y', 'ong', '4']], [['g', 'ai', '1']], [['y', 'u', '3'], ['j', 'u', '4']], [['c', 'e', '4'], ['sh', 'i', '4']], [['y', 'u', '4']], [['ch', 'u', '3'], ['l', 'i', '3']], [['g', 'ong', '1'], ['j', 'u', '4']], [['d', 'e', '']], [['sh', 'u', '1'], ['ch', 'u', '1']], [['j', 'ie', '2'], ['g', 'uo', '3']]]

[1] DT: determiner
[2] JJ: adjective
[3] DEG: associative "的"(i.e., "of")
[4] ['a', 'b', 'dobj']: a noun 'b' is the direct object ('dobj') that is directly affected by the action of a verb 'a'
[5] For explanations about other types of dependencies shown in this table, please refer to Stanford typed dependencies manual [27]

To facilitate the further process, the dependency of two words was denoted as three parts: 1) the governor word with its POS tag and index in the sentence; 2) the dependent word with its POS tag and index in the sentence; and 3) the relationship between two words. Moreover, pronunciation was denoted as initials, finals, and tones of each character in each word generated by segmentation.

Fuzzing Rules Extraction: According to the corpus, a total of 190 fuzzing rules was extracted, including 98 pronunciation errors (including accent phenomena), 39 vocabulary errors, 8 grammar errors, and 45 nonerror linguistic phenomena (disfluency and semantically similar expressions).

Bayesian Network Modeling: To train weights in BN, we chose CCMT 2019-BSTC [35], a Chinese speech data set in the real world, to calculate transition probabilities. We manually processed 5-h audio data sampled from CCMT 2019-BSTC and estimated the transition probabilities of each fuzzing rule by counting how many times a transition already occurred and how many times the precondition is met. Taking

the insertion of "zhège" ("这个," i.e., "this") before a noun as an example, we observe n nouns of which m nouns have "zhège" before it; thus, the probability of inserting "zhège" is m/n. In the 5-h audio, we hunted 1852 times of linguistic phenomena extracted in this article and 458 456 times of meeting preconditions.

Fuzzing Commands Generation: Finally, we obtained 1483 examples of skill commands from the official website of Tmall Genie [40] through the crawler and randomly selected 100 of them to implement the fuzzing processing with a probability threshold of 0.0005. By limiting the probability of the fuzzing process, we can get speech errors or language expressions that are more likely to occur in the user speech, to reduce the space of fuzzing. Our fuzzing tool generated 17 177 fuzzing results.

IV. Evaluation

To evaluate the risk of performing voice misinterpretation attacks on a VA system, we take the following questions into account.

Q1: How many fuzzing commands obtained in Section III can no longer invoke their target skills? If a command can still open its target skill after fuzzing, it means that when parsing this command, VAs can handle or ignore some possible speech errors or other language habits of users. Fuzzing commands that cannot invoke the correct skill can be used by adversaries to design the invocation name of malicious skills and perform squatting attacks. These fuzzing commands are *effective* in this article.

Q2: Which features of the fuzzing process are related to the effectiveness of fuzzing commands? Analyzing the influence of different parameters or features of fuzzing on the effectiveness of fuzzing results can help us find suitable parameters for fuzzing processing and understand what kinds of fuzzing commands that VAs are more tolerable or sensitive to.

Q3: In a real scenario, will users accidentally invoke the malicious skill designed based on the method of squatting attacks? We collected voice commands to invoke a target skill from the public, and then test if these voice commands can open malicious skills that have a similar invocation name. Through this test, we analyze the feasibility of squatting attacks.

Q4: Can skills that use malicious invocation names be released on the skill market and available to users? We submitted some skills to three different skill development platforms to see whether the certification process can detect the possibility of squatting attacks from skill invocation names or not.

Q5: How might malicious skills affect users? We analyzed possible methods of using squatting attacks to perform malicious operations on different platforms and the harms they may cause.

We conducted fuzzing commands recognition evaluation (in Section IV-A) to answer *Q1* and *Q2*, squatting attack feasibility evaluation (in Section IV-B) to answer *Q3* and *Q4*, and analysis of possible behaviors and impacts of malicious skills (in Section IV-C) to answer *Q5*. Futhermore, we summarized the security problems of VAs (in Section IV-D) and proposed defense suggestions (in Section IV-E).

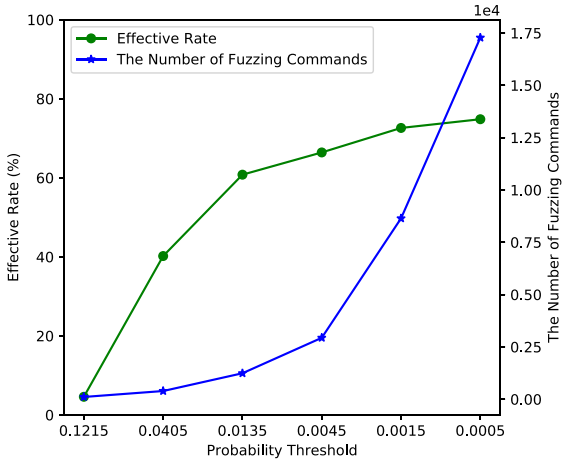

Fig. 6. Number of fuzzy commands and the effective rate for different probability threshold.

A. Fuzzing Commands Recognition Evaluation

In this section, we evaluate the effectiveness of fuzzing commands and analyze how different factors influence the effective rate.

Experiment Methodology: The test was conducted in a quiet environment. The pinyin of fuzzing commands was converted into text and then converted into audio using TTS to play to a Tmall Genie smart speaker. During the test, a device wake-up name was played first, and fuzzing commands were played after the device was awakened and its response ended. Since TTS might be unable to synthesize some spoken sentences or speech errors like a human, we read these fuzzing commands to test the VA at the same time. If a fuzzing command failed to get a response or get an error response from the device in both cases, the command would be regarded as an effective one.

Experimental Results: Finally, 17 177 fuzzing commands were tested, with an average fuzzing time of 1.80. There were 12 927 effective fuzzing commands, with an effective rate of 75.26%, and an average fuzzing times of 1.88, which was slightly higher than the average fuzzing times of commands we tested. We analyzed the influencing factors of the effectiveness of fuzzing commands from the following four aspects: 1) probability threshold; 2) fuzzing times; 3) command types; and 4) fuzzing types.

Probability Threshold: As shown in Fig. 6, a decrease of the probability threshold from 0.1215 to 0.0135 significantly improves the effective rate of fuzzing results. When the threshold is less than 0.0015, the effective rate increases slowly, while, at the same time, the number of fuzzing commands increases rapidly. This result shows that appropriately reducing the probability threshold of fuzzing results can make it more difficult for VAs to understand. But the threshold should not be too small, because the improvement of effective rate is not obvious while too many fuzzing commands are generated. The threshold we chose was sufficient for the discussion on fuzzing results; further decreasing the threshold would expend computing resources whereas the effective rate would increases indistinctively.

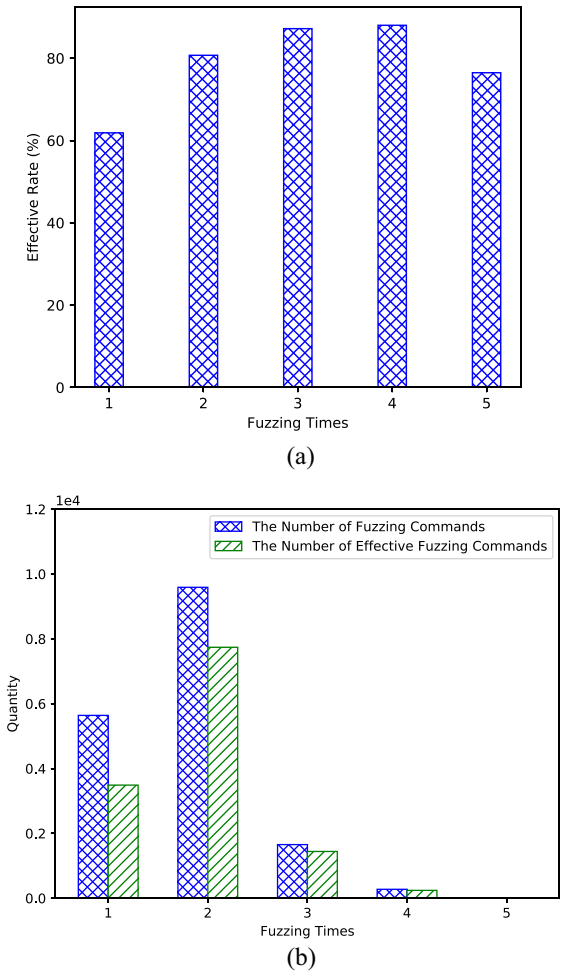

Fig. 7. Number of fuzzy commands and the effective rate for different fuzzing times. (a) Effective rate for different fuzzing times. (b) Number of fuzzing commands for different fuzzing times.

Fuzzing Times: As shown in Fig. 7, with the limitation of the probability threshold, less than six times single fuzzing processes are conducted on each command template. The effective rate of fuzzing commands obtained by performing two to four times of fuzzing can reach more than 80%. Moreover, since commands must satisfy more preconditions required in fuzzing rules when more single fuzzing processes are conducted, the number of fuzzing commands with larger fuzzing times gradually decreases. The result shows that appropriately increasing times of fuzzing can improve but cannot indefinitely increase the effective rate.

Command Types: Among 100 sampled command templates, only three have no effective fuzzing results, which means that 97% of commands may be exploited to carry out squatting attacks. To analyze effective rates of fuzzing results for different commands, sampled commands are classified into seven types, and effective rates of their fuzzing results are calculated separately. As shown in Fig. 8, the effective rate of commands using skill invocation names (② ③ ⑤ ⑥) is relatively higher. In practice, users may have speech errors when using invocation names, and VAs will be unable to match commands to correct skills, in this case, thus fuzzing results of these commands have higher effective rates. For commands using invocation names to enter skills, commands provided by Tmall

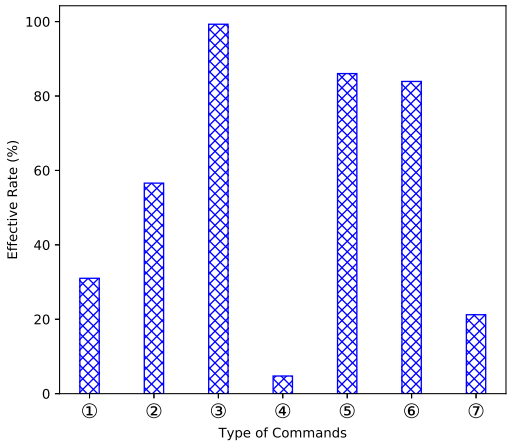

Fig. 8. Effective rates for different types of commands. We considered commands ① for Tmall Genie skills without invocation name; ② for Tmall Genie skills with (“打开/开 始/来个,” i.e., “open/start/come some” +) invocation name; ③ for Tmall Genie skills with invocation name and intent; ④ for third-party skills without invocation name; ⑤ for third-party skills with (“打开/开始/来个,” i.e., “open/start/come some” +) invocation name; ⑥ for third-party skills with invocation name and intent; and ⑦ other commands (such as “下一集”, i.e., “next one”).

Genie (② ⑤) have lower effective rates, which means that they are easier to match correct skills. This is because Tmall Genie can set several invocation names for a skill, but third-party developers usually are allowed to set only one invocation name for a skill. Moreover, the effective rate of commands using invocation name and intent for Tmall Genie skills (③) is high. We found that there are only a few sampled commands of this type, and these command templates cannot invoke correct skills due to ASR errors, which is the primary cause for the high effective rate instead of fuzzing processes. In addition, fuzzing results of commands without invocation names (① ④) is less effective. Since VAs will directly extract the value of required entities from commands without invocation names, as long as values of required entities in fuzzing results are correct, expected responses will be returned from VAs.

Fuzzing Types: We consider four types of fuzzing processes corresponding to four linguistic phenomena: pronunciation errors (①), vocabulary errors (②), grammar errors (③), disfluencies, or semantically similar expressions (④). We calculate the effective rate of fuzzing results that have or only have this type of linguistic phenomena. As shown in Fig. 9, fuzzing results of grammar errors (③) achieve a higher effective rate. Since most grammar errors we consider in this article are errors like omitting a *noun compound modifier*, which makes commands more difficult to match correct skills. For pronunciation errors (①), fuzzing results that only have pronunciation errors are less effective, which is consistent with the actual situation during the test: as long as pronunciation errors in commands can be accurately restored, correct skills can be launched.

B. Squatting Attacks Feasibility Evaluation

In this section, we verify the feasibility of squatting attacks and test whether the certification process of different platforms can detect the possibility of squatting attacks from skill invocation names or not.

TABLE III
SQUATTING ATTACK RESULTS

Skill	Invocation Name	According to	The number of commands that open the skill	Commands that contain the invocation name but cannot open the skill
Target Skill	"shuì qián yīnyuè" (睡前音乐)	The Original Invocation Name	36	"qǐng dǎkāi shuì qián yīnyuè" (请打开睡前音乐) "wǒ yào tīng shuì qián yīnyuè" (我要听睡前音乐) "wǒ xiǎng tīng shuì qián yīnyuè" (我想听睡前音乐) "lái diǎn shuì qián yīnyuè ba" (来点睡前音乐吧) "bòfàng shuì qián yīnyuè" (播放睡前音乐)
Malicious Skill	"yīgè shuì qián yīnyuè" (一个睡前音乐)	Fuzzing Result	0	"fàng yīgè shuì qián yīnyuè" (放一个睡前音乐)
	"shuì qián yīnyuè ba" (睡前音乐吧)		1	"lái diǎn shuì qián yīnyuè ba" (来点睡前音乐吧)
	"lái gè shuì qián yīnyuè" (来个睡前音乐)	Command Template	0	"lái gè shuì qián yīnyuè ba" (来个睡前音乐吧)
	"shuì qián yīngyǔ" (睡前英语)	Recognition Error	1	-
	"shōucáng yīnyuè" (收藏音乐)		2	-

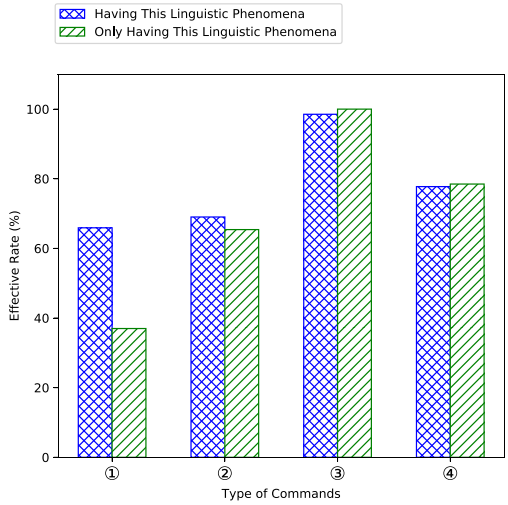

Fig. 9. Effective rates for different types of fuzzing rules.

Experiment Methodology: To create customized skills for squatting attacks on the AliGenie development platform, we took the skill "shuì qián yīnyuè" ("睡前音乐," i.e., "Music Before Sleep") released in Tmall Genie skill market as a target skill and selected some commands from fuzzing results of its command templates to design invocation names. In addition, the platform provided a function for developers to test skill invoking, dialogue processing, and skill responding on real devices, which could be used to test squatting attacks without releasing skills. In other words, we used uncertified skills for this evaluation.

To test the effect of squatting attacks, it was necessary to collect user voice commands to invoke the target skill. Since real user voice data were protected from disclosure, we collected voice data from the public. We guided volunteers to complete data collection by describing the scenario of using smart home VAs. In detail, volunteers were informed of the invocation name "shuì qián yīnyuè" ("睡前音乐," i.e., "Music Before Sleep") of the target skill and were asked to start this skill through voice commands. Two command templates, "shuì qián yīnyuè" ("睡前音乐," i.e., "Music Before Sleep") and "dǎkāi shuì qián yīnyuè" ("打开睡前音乐," i.e., "open Music Before Sleep"), were given to volunteers. These command templates came from the skill description on the skill market, so the process of data collection was as close to the real scene of using VAs as possible. Finally, 88 pieces of audio were collected after removing completely unrelated data.

Experimental Results: The test results are shown in Table III. We will discuss the feasibility of squatting attacks according to different types of malicious invocation names.

1) Using Fuzzing Results as Invocation Names: In the test, we find that Tmall Genie is strict in the use of invocation names. For example, the command "dǎkāi shuì qián yīnyuè" ("打开睡前音乐," i.e., "open Music Before Sleep") can start the correct skill, but "qǐng dǎkāi shuì qián yīnyuè" ("请打开睡前音乐," i.e., "please open Music Before Sleep") cannot. We consider that there are some preset rules for Tmall Genie to find invocation names in voice commands, such as the pattern of the verb, such as ⟨*"dǎkāi"/"láigè"/"kāishǐ" ("打开," i.e., "open" / "来个," i.e., "come some" / "开始," i.e., "start")* + *the invocation name*⟩, but these rules cannot cover all possible expressions. Only 36 of 88 voice commands start the target skill. These 36 commands are the same as given command templates with few of them have pronunciation errors, while other expressions such as "wǒ xiǎng tīng shuì qián yīnyuè" ("我想听睡前音乐," i.e., "I want to listen to Music Before Sleep") cannot open the target skill. Although extracting invocation names by strict pattern causes some user commands to fail to open the correct skill, the possibility of users accidentally turning on malicious skills will also decrease. For instance, the customized skill "yīgè shuì qián yīnyuè" ("一个睡前音乐," i.e., "a Music Before Sleep") cannot be open by the voice command "fàng yīgè shuì qián yīnyuè" ("放一个睡前音乐," i.e., "play a Music Before Sleep"), which includes its invocation name. The invocation name "shuì qián yīnyuè ba" ("睡前音乐吧," i.e., "Music Before Sleep") appears twice in voice commands, in which "lái gè shuì qián yīnyuè ba" ("来个睡前音乐吧," i.e., "come some Music Before Sleep") successfully starts the skill, but "lái diǎn shuì qián yīnyuè ba" ("来点睡前音乐吧," i.e., "come some Music Before Sleep") fails.

2) Using Command Templates as Invocation Names: We used command templates, such as ⟨*"bòfàng"/"dǎkāi"*

TABLE IV
SUBMITTED SKILLS AND THEIR CERTIFICATION RESULTS

Platform	Target Skill	Malicious Invocation Name	According to	Result of Certification	Feedback
AliGenie	"shuì qián yīnyuè" (睡前音乐)	"shuì qián yīnyuè ba" (睡前音乐吧)	Fuzzing Result	Fail	Highly similar to the online skill "shuì qián yīnyuè" (睡前音乐)
		"shuì qián yīngyǔ" (睡前英语)	Recognition Error	Released	-
XiaoMi XiaoAi	Míngxiǎng yīnyuè (冥想音乐)	Míngxiǎng de yīnyuè (冥想的音乐)	Fuzzing Result	Released	-
	Kāfēi guǎn bái zàoshēng (咖啡馆白噪声)	Kāfēi guǎn bái zàoyīn (咖啡馆白噪音)	Recognition Error	Fail	Invocation name has been used
DuerOS	Xià yǔ de shēngyīn (下雨的声音)	Xià yǔ shēngyīn (下雨声音)	Fuzzing Result	Fail	Skill name and invocation name are common
	Línyù shēng (淋浴声)	Tīng yǔ shēng (听雨声)	Recognition Error	Fail	

("播放," i.e., "play"/"打开," i.e., "open") + the invocation name ⟩, provided for opening the target skill to design invocation names. As a result, malicious skills cannot be opened by voice commands, even there are voice commands exactly the same as the invocation name we set. For example, using "lái gè shuì qián yīnyuè" ("来个睡前音乐," i.e., "come some Music Before Sleep") (marked with * in Table III) as the malicious invocation name, even if the device recognizes a command that is exactly the same as the invocation name, the malicious skill cannot be turned on. It indicates that Tmall Genie may analyze and delete the verb before the invocation name when processing a command.

3) Using Recognition Errors as Invocation Names: According to VAs recognition errors of collected voice commands, we designed some malicious invocation names, such as "shuì qián yīngyǔ" ("睡前英语," i.e., "English Before Sleep") and "shōucáng yīnyuè" ("收藏音乐," i.e., "collect the music"). As a result, skills with these invocation names are successfully opened by voice commands. At the same time, the skill "shōucáng yīnyuè" ("收藏音乐," i.e., "collect the music") also hijacks the music collection function of Tmall Genie. However, since our skills have not been submitted, it is still unclear whether these skills will be rejected or not in the certification process.

Certification Process Evaluation: Through the above test, we have proved that even if users know command templates, they may give voice commands that are different from the expectation of the target skill and some commands include the invocation name of malicious skills. To further verify whether these squatting attack skills can pass the certification process of platforms, we submitted six customized skills on three different Chinese voice service platforms, DuerOS, XiaoAi, and AliGenie. We chose some invocation names that are similar to the invocation name of released skills on the skill market, while not easy to be detected. Since we only considered whether skills would be launched by mistake, the skills we released do not contain any malicious functions.

Among the three platforms, DuerOS and XiaoAi allow configuring the response logic on the platform directly without server maintenance. While for AliGenie, the response logic can only be processed by the developer server. Thus, when developing skills on AliGenie, we used Apache and mod_wsgi to build a local server and create a simple Web application using Flask to process the response logic. When an event occurs, AliGenie will initiate an HTTP request to the URL that we provide.

Submitted skills and their certification results are shown in Table IV. Two of the submitted skills could pass the certification process and be released to skill markets, which proves that squatting attacks can be achieved in real scenarios to a certain extent. Besides, although some skill releasing applications were rejected, only one of the four rejected skills, "shuì qián yīnyuè ba" ("睡前音乐吧," i.e., "Music Before Sleep") submitted on AliGenie, was clearly pointed out to be highly similar to its target skill. The XiaoAi development platform pointed out that the name of the submitted skill "kāfēi guǎn bái zàoshēng" ("咖啡馆白噪声," i.e., "White Noise in Coffee Shop") already exists, but there was indeed no skill in the skill market that had the same skill name or invocation name that we proposed. We think the platform might have detected that the name we proposed is similar to the name of an existing skill. The two skills submitted on DuerOS were both pointed out as "common." We believe that because there is already a large number of skills with similar content and names, and the platform could prompt developers to optimize the invocation name to distinguish it from other skills and make it easier for users to call.

C. Analysis of Possible Behaviors and Impacts of Malicious Skills

In this part, we put forward some methods of using squatting attacks to perform malicious operations based on our knowledge of real skill development platforms and analyze their threats to privacy security.

1) Eavesdropping by Setting Slots: There have been some studies [25], [41] suggesting that third-party skills can eavesdrop on users without their consent by disguising the end of skills. However, in our tests on Chinese VAs, we found that implementing eavesdropping faces some challenges.

For all the three platforms we investigate, the original text of the user voice command will only be included in the request sent to the third-party server when the command matches one of the preset intents. In other words, a voice command matching no intent will not be obtained by third-party developers. If adversaries want to achieve eavesdropping only through setting intents, the content of user speech is required to be included in the corpus which the developer considers can hit a skill intent;

thus, the content and length of speech that can be eavesdropped on are limited.

To achieve eavesdropping by malicious skills, we propose to obtain the textual speech of users by setting slots. The adversary can set a slot (for example, "city"), and remove the questioning sentence in the returned slot questioning request or replace it with a sentence reminding the user that the skill ends. These deletions or modification of the code on the third-party server can be carried out after the skill has passed the certification process. The malicious skill seems to end from the user's point of view, but it actually still records and analyzes the user speech and will send the text of the speech to the third-party server as long as it contains a slot value set by the adversary (for example, a "city" is mentioned in the user speech). In this way, adversaries could achieve temporary eavesdropping of user speech that contains specific words.

Tests show that the method we proposed is feasible on the three platforms, but the degree of implementation is different. On DuerSO and AliGenie, as long as the user voice command contains a system-defined slot value, we can receive the textual user speech. However, if we set slots for customized parameters, there is a need of adding corpus to match these slots; otherwise, the system cannot recognize these parameters. The addition of corpus will inevitably limit the form of user voice speeches that can be received by adversaries; thus, the slot filling of customized entities is not as flexible as it of system entities. For the XiaoAi platform, the slot filling of both customized or system-defined entities embraces the limit caused by adding corpus. This restriction might prevent our eavesdropping attack to a certain extent.

2) Obtaining Privacy by Imitating Benign Skills: Malicious skills that perform squatting attacks can deceive users and obtain their private information, such as the city-level (AliGenie) or street-level (XiaoAi and DuerOS) address information and the gender (XiaoAi), by imitating the behavior of benign skills. In addition, the DuerOS platform also supports skills to use Baidu account information of users, including nicknames, e-mail addresses, phone numbers, avatars, printer services, etc.

3) Denial of Service: Malicious skills can directly terminate user's conversations when they are triggered to realize denial of service attacks. For instance, a malicious skill that has a similar invocation name to a smart lock managing skill can ignore a user's intent of locking the door, which will threaten personal safety. In the same way, voice command misinterpretation will cause denial of service risks in general IoT scenarios, since user's important commands might be missed by VAs.

D. Security Problems Summary

Combining the above evaluations, we summarize the security problems of Chinese VAs that could be leveraged by adversaries to conduct squatting attacks.

1) VAs Are Easy to Misunderstand User Intents, Which Makes Squatting Attacks Possible: Compared with other forms of interaction, input signals in the voice interaction are more uncertain. Although developers of skills will inform users of some command templates for invoking skills, user commands are still diverse in the real scene. Therefore, commands for invoking skills are not always correctly matched to desired skills, and adversaries can design malicious invocation names according to possible expressions to carry out squatting attacks.

In the test of squatting attack, malicious skills using fuzzing results as the invocation name are opened by 1 of 88 collected voice commands. Some voice commands contain malicious invocation names although they cannot invoke corresponding skills. It indicates that it is feasible to use fuzzing commands to design squatting attacks. To increase the number of successful attacks, adversaries can create and release several malicious skills at the same time, or conduct such attacks against popular skills to obtain user information or provide users with wrong phone numbers or URLs for phishing on a large scale.

2) VAs Recognition Errors Can Be Used to Carry Out Squatting Attacks: The accuracy of speech recognition may be affected by many factors, such as accents, fast speech rates, or environmental noises, which may cause the speech recognition result to be inconsistent with user commands. Adversaries can design malicious skills based on common recognition mistakes in VAs. These skills may be unintentionally invoked by users and threaten user privacy security.

Our tests show that there are many misrecognition phenomena in Tmall Genie, as 38 among the 88 collected voice commands are misrecognized. Adversaries can utilize these numerous speech recognition errors to design invocation names of malicious skills. The squatting attack test shows that skills exploiting speech recognition errors as invocation names can be successfully invoked by the user's voice commands.

3) Skill Development Platform Has Loopholes in Check Skill Invocation Names: The skill development platform usually checks whether the invocation name is duplicated or similar to the invocation name of existing skills. However, the platform lacks an effective method to detect whether the invocation name is a suspicious variant of an existing skill invocation name. These variants come from possible user expressions or possible recognition errors of VAs. Adversaries can generate variants based on certain rules, for example, leveraging the fuzzing tool proposed in this article, to generate variants of user commands. Moreover, adversaries can select some seemingly reasonable invocation names to further increase the difficulty of certification processes. For example, in the test, the invocation name "shuì qián yīngyǔ" ("睡前英语," i.e., "English Before Sleep") designed based on speech recognition errors is not easily seen as a variant of "shuì qián yīnyuè" ("睡前音乐," i.e., "Music Before Sleep"), but it can be invoked by voice commands which expect to invoke the latter one.

E. Defense Suggestions

Finally, we provide two suggestions, i.e., standardizing the syntactic structure of skill invocation commands and improving the certification process of skills, to mitigate the risk of squatting attacks.

1) Standardizing the Syntactic Structure of Skill Invocation Commands: Standardizing the syntactic structure of invocation commands to invoke skills explicitly can prevent squatting attacks to some degree. Common patterns of invocation commands include ⟨*verb (can be omitted)* + *invocation name*⟩ and ⟨*preposition (can be omitted)* + *invocation name* + *intent*⟩, which correspond to two ways of invocation, "only invoking the skill" and "invoking the skill and explaining purpose," respectively.

The probability of launching malicious skills can be reduced by limiting valid verbs or prepositions that may appear in commands. For example, in the test, some commands containing malicious invocation names cannot open malicious skills: "lái diǎn shuì qián yīnyuè ba" ("来点睡前音乐吧," i.e., "come some Music Before Sleep") cannot invoke the malicious skill "shuì qián yīnyuè ba" ("睡前音乐吧," i.e., "Music Before Sleep"), because "lái diǎn" ("来点," i.e., "come some") may not be recognized as a verb, or is not within verbs considered by Tmall Genie.

In addition, using strict patterns to parsing commands can prevent squatting attacks that use ⟨ *verb* + *invocation name* ⟩ or ⟨ *preposition* + *invocation name* ⟩ as malicious invocation names. The test results indicate that Tmall Genie may have similar operations, that is, verbs may be deleted when processing skill invoking commands, so squatting attacks through these malicious skills can be prevented.

2) Improving the Certification Process of Skills: The similarity evaluation of skill invocation names can prevent adversaries from using variants of invocation names to conduct squatting attacks. Usually, the skill development platform prohibits developers from using invocation names similar to existing skills, but only the case of completely same or similar pronunciation is considered, and other types of variants cannot be detected.

Improving the detection process requires the development platform to be aware of possible variants of existing invocation names. For example, the fuzzing tool designed in this article can be used to generate variants. By comparing the similarity between variants and invocation names provided by third-party developers, the platform can detect suspicious skills, and then further analyze whether the suspicious skill has malicious behaviors. Through such a strict mechanism of certification, the platform can avoid most squatting attacks.

V. Related Work

A. Voice Recognition Attacks

Voice recognition attacks include signal manipulation-based attacks, obfuscation-based attacks, and adversarial example-based attacks, aiming at three processes of ASR: 1) preprocessing; 2) feature extraction; and 3) text generation based on language models.

Signal Manipulation-Based Attacks: Diao *et al.* [42] proposed to leverage malicious software installed in mobile phones to activate VAs and use speakers on the mobile phone to play malicious voice signals. Malware can achieve this process without any permissions of devices, but the problem is that users can perceive the malicious voice signal played by their phone. Roy *et al.* [13] proposed to exploit the nonlinearity of microphone diaphragms and power amplifiers to make two signals with frequencies beyond the range of human hearing (such as 40 and 50 kHz) produce a beat frequency when passing through the receiving system. The low-frequency beat signals can occupy the receiver. On the positive side, this method can be used to prevent eavesdropping or recording. But on the other hand, adversaries can use it to conduct denial-of-service attacks, preventing users from using VAs or recording functions. Dolphinattack proposed by Zhang *et al.* [14] also uses signals with frequencies beyond the range of human hearing. The malicious voice command is modulated on the ultrasonic carrier in the range of 20–24 kHz. The nonlinearity of the receiver circuit amplifier will make malicious signals demodulate to low-frequency before passing through the low-pass filter. It has been verified that Dolphinattack can launch the voice recognition function of devices with signals that are not perceptible to a human, and cause the recognition system to execute the command in malicious signals. At the same time, Zhang *et al.* also proposed corresponding defense measures, such as classifying the audio by support vector machines and distinguishing the audio generated by Dolphinattack from normal human voices.

Obfuscation Based-Attacks: Vaidya *et al.* [15] analyzed the difference between speech recognition and human language understanding and proposed to use sound sources in a public environment to play voice signals that can be recognized by speech recognition systems but cannot be understood by humans to make VAs execute malicious commands. In ASR, since voice recognition systems only need extracted features in voice signals to generate text, the input audio containing enough features can make the system recognize expected text.

Adversarial Example-Based Attacks: Based on [15], Carlini *et al.* [43] added the analysis of acoustic models and language models in ASR. They took the state sequence of the hidden Markov model (HMM) formed by the phoneme sequence as the target to find the input state that matches this HMM sequence, which will make a malicious command be recognized. Yuan *et al.* [16] used songs as the carrier of malicious voice signals. By inputting the command voice and the song into the ASR system, and analyzing the necessary features of the voice signal, they modified the acoustic model of the song to include necessary information.

Compared to voice recognition attacks, the speech misinterpretation-based squatting attack analyzed in this article is not limited by physical distance, thus may bring privacy threatens to VA users on a large scale.

B. Voice Misinterpretation Attacks

Kumar *et al.* [17] analyzed recognition errors in speech recognition and proposed to utilize characteristics of words that are easy to be confused in pronunciation, such as homophones, phonetic similar words, and compound words, to generate malicious commands that are confused with the original one to achieve squatting attacks. Zhang *et al.* [5] considered the inconsistency between voice commands given

by users and command templates, which may be caused by pronunciation, regional vocabulary, synonyms, and phoneme exchange. They proposed a fuzzing tool, LipFuzzer, under the guidance of a language model. Command templates are fuzzed at three levels: 1) pronunciation; 2) vocabulary; and 3) grammar. By calculating the frequency of linguistic phenomena in commands given by real users, they could limit the number of fuzzing results by adjusting the probability threshold.

Zhang *et al.* [25] analyzed security risks caused by third-party skills of smart home VAs, and proposed VSA and voice masquerading attack (VMA). Malicious skills in VSA use similar or paraphrased sentences to hijack a legal skill command.

Different from previous works on voice misinterpretation attacks, we propose a fuzzing tool for Asian language-driven VAs. Moreover, in terms of fuzzing policy extraction, previous research in English-driven VAs only considered grammar-level linguistic information, such as PoS tagging, stemming, and lemmatization, while we provide grammar-level preprocessing to analyze dependencies between words, which will be more expressible for fuzzing rule extraction.

C. Skill Behavior Analysis

Guo *et al.* [41] proposed the first systematic research on behaviors of skills, based on a suite of grammar-based methods, including utterance extraction, question understanding, and answer generation for specific skills. They built an interactive system: SkillExplorer and analyzed skills from Amazon and Google market. They found that over 1000 skills request users to provide personal information without declaring them in the privacy policy of skills or using specific APIs to configure permissions; 68 skills eavesdrop on user's conversations after user giving commands to stop.

While, in this article, we analyze the security risks on Chinese VA services, and point out some differences in the extent of privacy leakage for a variety of platforms.

VI. Conclusion

In this article, we systematically analyzed the misinterpretation-based voice attacks to Asian-language-driven VA systems. Taking typical Chinese-VAs as our target systems, we developed Harmony-Fuzzer, a voice fuzzing tool, which processes corpus containing language phenomena, such as speech errors, disfluencies, and synonymy. With NLP tools, the logic of language phenomena is abstracted into fuzzing rules. Using BN to describe the conversion between features of sentences, we reduced the space for fuzzing effectively by setting a probability threshold for the fuzzing processing. Finally, by analyzing the result of the fuzzing and squatting attack test, we further pointed out security problems in user intent analysis and skill certification process of Chinese VAs and provide defense suggestions.

Acknowledgment

The authors thank the anonymous reviewers for their valuable comments. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the funding agencies.

References

[1] Google. *Discover What Google Assistant*. Accessed: Oct. 2020. [Online]. Available: https://assistant.google.com

[2] T. G. A. Union. *Tmall Genie Controllable Device Query*. Accessed: Oct. 2020. [Online]. Available: https://bot.tmall.com/equipment

[3] Y. Liang, Z. Cai, J. Yu, Q. Han, and Y. Li, "Deep learning based inference of private information using embedded sensors in smart devices," *IEEE Netw.*, vol. 32, no. 4, pp. 8–14, Jul./Aug. 2018.

[4] Y. Chen *et al.*, "Devil's Whisper: A general approach for physical adversarial attacks against commercial black-box speech recognition devices," in *Proc. 29th USENIX Security Symp. (USENIX Security)*, Aug. 2020, pp. 2667–2684. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presenta-tion/chen-yuxuan

[5] Y. Zhang, L. Xu, A. Mendoza, G. Yang, P. Chinprutthiwong, and G. Gu, "Life after speech recognition: Fuzzing semantic misinterpretation for voice assistant applications," in *Proc. 26th Annu. Netw. Distrib. Syst. Security Symp.*, Feb. 2019, pp. 1–15.

[6] B. MacCartney and C. Potts. *Natural Language Understanding Course Overview*. Accessed: Oct. 2020. [Online]. Available: http://web.stanford.edu/class/cs224u/2019/materials/cs224u-2019-intro.pdf

[7] J. Levis and R. Suvorov, "Automatic speech recognition," in *The Encyclopedia of Applied Linguistics*, C. A. Chapelle, Ed. Oxford, U.K.: Blackwell Publ. Ltd., 2012.

[8] J. Lai, C.-M. Karat, and N. Yankelovich, "Conversational speech interfaces and technologies," in *The Human-Computer Interaction Handbook*. Boca Raton, FL, USA: CRC Press, 2007, pp. 407–418.

[9] S. CoreNLP. *Stanford Corenlp Natural Language Software*. Accessed: Oct. 2020. [Online]. Available: https://stanfordnlp.github.io/CoreNLP/

[10] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford CoreNLP natural language processing toolkit," in *Proc. 52nd Annu. Meet. Assoc. Comput. Linguist. Syst. Demonstrations*, 2014, pp. 55–60.

[11] P. Xu and R. Sarikaya, "Convolutional neural network based triangular CRF for joint intent detection and slot filling," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand.*, 2013, pp. 78–83.

[12] J. P. Pullen. *Amazon Echo Owners Were Pranked by South Park and Their Alexas Will Make Them Laugh for Weeks*. Accessed: Oct. 2020. [Online]. Available: http://fortune.com/2017/09/14/ watch-south-park-alexa-echo/

[13] N. Roy, H. Hassanieh, and R. R. Choudhury, "Backdoor: Making microphones hear inaudible sounds," in *Proc. 15th Annu. Int. Conf. Mobile Syst. Appl. Services*, 2017, pp. 2–14.

[14] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "DolphinAttack: Inaudible voice commands," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 103–117.

[15] T. Vaidya, Y. Zhang, M. Sherr, and C. Shields, "Cocaine noodles: Exploiting the gap between human and machine speech recognition," in *Proc. 9th USENIX Workshop Offensive Technol.*, 2015, p. 16.

[16] X. Yuan *et al.*, "CommanderSong: A systematic approach for practical adversarial voice recognition," in *Proc. 27th USENIX Security Symp.*, 2018, pp. 49–64.

[17] D. Kumar *et al.*, "Skill squatting attacks on Amazon Alexa," in *Proc. 27th USENIX Security Symp.*, 2018, pp. 33–47.

[18] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 2, pp. 766–775, Apr.–Jun. 2020.

[19] J. E. Pincott. *Slips the Tongue*. Accessed: Oct. 6, 2020. [Online]. Available: https://www.psychologytoday.com/intl/articles/201203/slips-the-tongue

[20] Amazon Alexa. *Amazon Alexa Developer Support*. Accessed: Oct. 2020. [Online]. Available: https://developer.amazon.com/en-US/alexa

[21] Google. *The Developer Platform for the Google Assistant*. Accessed: Oct. 2020. [Online]. Available: https://developers.google.com/assistant

[22] AliGenie. *Aligenie Skills*. Accessed: Oct. 2020. [Online]. Available: https://www.aligenie.com/skillst

[23] Xiaomi. *Xiaoai Open Platform*. Accessed: Oct. 2020. [Online]. Available: https://xiaoai.mi.com/skill/create/index

[24] X. Zheng, Z. Cai, and Y. Li, "Data linkage in smart Internet of Things systems: A consideration from a privacy perspective," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 55–61, Sep. 2018.

[25] N. Zhang, X. Mi, X. Feng, X. Wang, Y. Tian, and F. Qian, "Dangerous skills: Understanding and mitigating security risks of voice-controlled third-party functions on virtual personal assistant systems," in *Proc. IEEE Symp. Security Privacy (SP)*, 2019, pp. 1381–1396.
[26] R. Pfau, *Grammar as Processor: A Distributed Morphology Account of Spontaneous Speech Errors*, vol. 137. Amsterdam, The Netherlands: John Benjamins Publ., 2009.
[27] M. C. De Marneffe and C. D. Manning, *Stanford Typed Dependencies Manual*. Accessed: Oct. 2020. [Online]. Available: https://downloads.cs.stanford.edu/nlp/software/dependencies_manual.pdf
[28] Q. Liu, "Research on the tongue slips in modern chinese language," M.S. thesis, College Arts, Northeast Normal Univ., Changchun, China, 2009.
[29] I. Chaturvedi, E. Ragusa, P. Gastaldo, R. Zunino, and E. Cambria, "Bayesian network based extreme learning machine for subjectivity detection," *J. Franklin Inst.*, vol. 355, no. 4, pp. 1780–1797, 2018.
[30] A. R. Golding and D. Roth, "A winnow-based approach to context-sensitive spelling correction," *Mach. Learn.*, vol. 34, no. 1, pp. 107–130, 1999.
[31] X. Yang, "An analysis of the slip of the tongue in speech communication," M.S. thesis, School Chin. Lang. Lit., Shandong Normal Univ., Jinan, Shandong, China, 2010.
[32] X. Chen, "Modern chinese types of the slip of the tongue and the analysis of cause," M.S. thesis, School Chin. Lang. Lit., Central China Normal Univ., Wuhan, China, 2008.
[33] *Chinese Dictionary*. Accessed: Oct. 2020. [Online]. Available: https://github.com/guotong1988 /chinese_dictionary
[34] C. I. P. S. China. *Chinese Linguistic Data Consortium*. Accessed: Oct. 2020. [Online]. Available: http://www.chineseldc.org/resource_list.php
[35] Baidu. *CCMT 2019—BSTC*. Accessed: Oct. 2020. [Online]. Available: https://ai.baidu.com/broad/subordinate?dataset=bstc
[36] S. Yu, H. Duan, and Y. Wu. *Corpus of Multi-Level Processing for Modern Chinese*. Accessed: Oct. 2020. [Online]. Available: https://doi.org/10.18170/DVN/SEYRX5
[37] C. L. R. O. Institute of Applied Linguistic Ministry of Education. *Modern Chinese Corpus*. Accessed: Oct. 2020. [Online]. Available: http://corpus.zhonghuayuwen.org
[38] W. Che, Z. Li, and T. Liu, "LTP: A Chinese language technology platform," in *Proc. 23rd Int. Conf. Comput. Linguist.*, 2010, p. 13.
[39] Z. Jinming, "A brief explanation of synonymous sentence patterns," *Chin. Teach. World*, vol. 23, no. 1, pp. 26–32, 1993.
[40] AliGenie. *Aligenie Skill Market*. Accessed: Oct. 2020. [Online]. Available: https://bot.tmall.com/skills
[41] Z. Guo, Z. Lin, P. Li, and K. Chen, "SkillExplorer: Understanding the behavior of skills in large scale," in *Proc. 29th USENIX Security Symp. (USENIX Security)*, Aug. 2020, pp. 2649–2666. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentat-ion/guo
[42] W. Diao, X. Liu, Z. Zhou, and K. Zhang, "Your voice assistant is mine: How to abuse speakers to steal information and control your phone," in *Proc. 4th ACM Workshop Security Privacy Smartphones Mobile Devices*, 2014, pp. 63–74.
[43] N. Carlini *et al.*, "Hidden voice commands," in *Proc. 25th USENIX Security Symp.*, 2016, pp. 513–530.

Jian Mao received the B.S. and Ph.D. degrees from Xidian University, Xi'an, Shanxi, China, in 1997 and 2004, respectively.

She is an Associate Professor with the School of Cyber Science and Technology, Beihang University, Beijing, China. Her research interests include IoT security, Web security, mobile security, and social network security.

Ziwen Liu received the B.S. degree in electronic and information engineering from Beihang University, Beijing, China, in 2020, where she is currently pursuing the master's degree with the School of Cyber Science and Technology.

Her research interests include learning-based security analysis, IoT security, and SDN security.

Qixiao Lin received the B.S. degree in electronic and information engineering from Beihang University, Beijing, China, in 2018, where he is currently pursuing the Ph.D. degree with the School of Cyber Science and Technology.

His research interests include learning-based security analysis, social network privacy preserving, and IoT security.

Zhenkai Liang (Member, IEEE) received the B.S. degree from Peking University, Beijing, China, in 1999, and the Ph.D. degree from Stony Brook University, Stony Brook, NY, USA, in 2006.

He is an Associate Professor with the Department of Computer Science, National University of Singapore, Singapore. His research interests include software security, Web security, and mobile security.