UNDERSTANDING AND SECURING VOICE ASSISTANT APPLICATIONS

A Dissertation

by

YANGYONG ZHANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Guofei Gu
Committee Members,	James Caverlee
	Jeff Huang
	Jeyavijayan Rajendran
Head of Department,	Scott Schaefer

August 2021

Major Subject: Computer Engineering

ABSTRACT

Internet of Things (IoT) has evolved from a traditional sensor network to an increasingly cloud-dependent ecosystem. This transition empowers IoT devices with abundant outsourced computational power. However, securing IoT devices is still a challenging task. The reason is that many IoT devices nowadays perform complicated tasks (e.g., voice assistants or VA) and are connected to different third parties. This research targets popular VA services such as Amazon Alexa and Google Assistant, which are rapidly appifying their platforms to allow a more flexible and diverse voice-controlled service experience.

Unfortunately, third-party skills have been reportedly posing threats to user privacy and security. The goal of this research is to conduct a systematic security analysis for different stages of a VA system, i.e., acoustic channel, speech processing, intent extraction, and application processing. Moreover, based on the analysis, corresponding defense strategies are proposed and evaluated. First, I investigate speech re-use problems in the acoustic channel. I then propose a security overlay named AEOLUS to tackle the speech re-use threat. Second, I study the speech processing stage by evaluating adversarial attacks targeting VA's speaker recognition systems. I present a novel attention-based audio perturbation scheme to help improve the efficiency and imperceptibility of generating audio adversarial examples. Third, I assess the intent extraction of VA to understand the root cause of semantic misinterpretation. A linguistic-guided fuzzing scheme is then proposed to evaluate the problem systematically in a large scale. Fourth, for VA application (or skill) processing stage, I conduct a user study with Alexa users to learn about how users perceive existing warning messages for voice assistant applications.

ACKNOWLEDGMENTS

I would first like to express my sincere gratitude to my advisor Dr. Guofei Gu. It has been a great honor to be his student. His patience, motivation, and immense knowledge helped me during my Ph.D. study and related research. I appreciate his contributions of time, ideas, and funding to make my Ph.D. experience productive and exciting. Also. I would like to thank Dr. Walt Magnussen, Texas A&M University Internet2 Technology Evaluation Center director, for his generous support to fund my Ph.D. study and guidance in outreaching to the public safety communication industry.

I want to thank my committee members, Dr. Jeff Huang, Dr. Jeyavijayan Rajendran, and Dr. James Caverlee, for their insightful suggestions and comments on my projects. I would also like to thank my mentors, Dr. Maliheh Shirvanian and Dr. Sunpreet Singh Arora when interning at Visa Research. Lastly, I appreciate all the help and support from my friends and families during my Ph.D. study.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This dissertation work was supported by a dissertation committee consisting of Professor Guofei Gu (advisor), Professor James Caverlee and Professor Jeff Huang of the Department of Computer Science & Engineering and Professor Jeyavijayan Rajendran of the Department of Electrical & Computer Engineering.

Chapter 4 is originated from a research work co-authored by Dr. Maliheh Shirvanian, Dr. Sunpreet Arora, Jianwei Huang, and Dr. Guofei Gu, and will be published in Proceedings of The 24th International Symposium on Research in Attacks, Intrusions and Defenses. Chapter 6 is originated from a research work co-authored by Dr. Lei Xu, Dr. Abner Mendoza, Dr. Guangliang Yang, Phakpoom Chinprutthiwong, and Dr. Guofei Gu, and was published in Proceedings of the 26th The Network and Distributed System Security Symposium.

Funding Sources

This dissertation work is based upon work supported in part by the National Science Foundation (NSF) under Grant no. 1816497 and 1700544, and ONR Grant No. N00014-20-1-2734. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF and ONR.

NOMENCLATURE

VA	Voice assistants
Skills	A common name for voice assistant applications
Lapsus	Speech errors (in Latin)
Aeolus	A tool name in this dissertation; The keeper of the winds in Greek mythology
BN	Bayesian network
IR	Infrared (light)
MTurk	Amazon Mechanical Turk

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Introduction

Popular Voice Assistant (VA) services such as Amazon Alexa and Google Assistant are now rapidly appifying their platforms to allow a more flexible and diverse voice-controlled service experience. For example, as of September 2019, there are more than 100,000 skills[1] published in the Alexa store [1]. Voice-driven skills developed by third parties are now capable of performing various tasks such as answering questions, purchasing grocery, or controlling smart home devices.

However, third-party skills can be dangerous to user privacy and security. For example, Skill-Explorer [2] reports that some third-party skills have been requesting users' private information and have been eavesdropping without strictly adhering to developer and platform policies. Another study [3] demonstrates that undesirable policy-violating skills can bypass the skill certification process and get published in the Alexa skill store.

In this dissertation, the goal is to study potential vulnerabilities that reside in the VA ecosystem systematically. Next, based on the findings, corresponding defense strategies are proposed and evaluated. To achieve this goal, I first address how VA applications (or skills) are designed and operated. Second, I explore how different components of skills can be exploited. Third, I propose security analysis and defense strategies to help start mitigating the uncovered problems. Moreover, this dissertation focuses on Amazon Alexa due to its complete VA application design, such as using a permission system, a large skill population, and a dominant market share. In one of the most important VA device markets - Smart Speaker, Amazon has a 71.2% market share in 2018 [4] and 69.7% in 2019 [5]. Although Google Home is catching up, Alexa is still the dominant VA in the U.S. market.

[1] Skill is commonly used as the name of VA applications.

1.2 Skill Design Overview.

As shown in Figure 1.1, it takes four steps for a skill to interact with users. Users' speech input is first recorded (i.e., Step 1) and converted into textual data using Automated Speech Recognition (ASR) in Step 2. In Step 3, Natural Language Understanding (NLU) component then extracts users' intent from the textual data. Before skills can be interacted by users, as the last step, intent data is redirected to the skill hosts, which could be operated by third parties.

- **Acoustic Channel.** Users' voice commands operate a VA via the open acoustic channel. The voice command is processed by the VA, and a response is returned either verbally to the user (e.g., announcing the time) or by performing a specific action (e.g., turning lights on/off). Devices that use VA are triggered by a user using a pre-specified phrase (e.g., Alexa, Hey Siri) and operate in a turn-taking mode. The device actively listens to the user once triggered, and the user listens to the device when it responds.

- **Speech Processing.** Skills' speech processing components are located in the platform cloud. For example, VA platforms such as Amazon Alexa will direct all the recorded speech to Amazon's cloud back-end. ASR will transcribe the speech, and the output is in the form of text data. Also, if user authentication and authorization are needed, a speaker recognition system (SRS) may also be deployed in this stage.

- **Intent Extraction.** After acoustic inputs (i.e., voice commands received by VA devices) are transcribed into textual data, a typical VUI-based VA platform processes the textual data to understand the user's intents with NLU [6] [7]. As a result, intents are generated for later skill processing. To produce accurate intents, NLU performs a two-step procedure: NLP transformation and Intent Classification. The NLP part follows standard procedures such as Tokenization (word segmentation), Coreference Resolution (COREF), and Named Entity Recognition (NER) [8]. This gathers the syntax information of given textual data. Next, to understand the semantic meaning, NLU further matches the syntax data with a pre-built intent classification tree. In the tree, the branch with the highest confidence will be selected

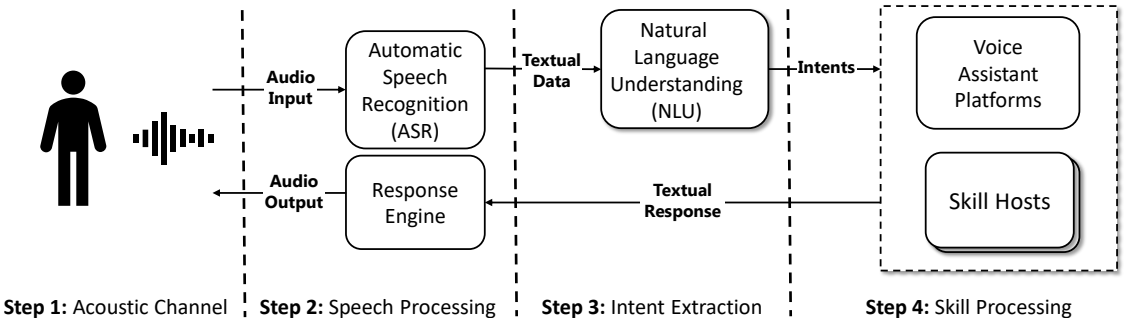


Figure 1.1: Skill pipeline overview.

to produce intents. Note that there may exist skill-irrelevant built-in branches for matching VA's default services, such as alarm and default music.

- **Skill Processing.** Skills developed by third parties are now capable of performing various tasks such as answering questions, purchasing groceries, or controlling smart home devices. However, third-party skills have been reportedly posing threats to user privacy and security. For example, SkillExplorer [2] reports that some third-party skills have been requesting users' private information and eavesdropping without strictly adhering to the developer and platform policies. Also, other studies [3, 9] demonstrate that such undesirable policy-violating skills can easily bypass the skill certification process and get published in the Alexa skill store.

1.3 Research Solutions

In this section, I conduct a series of security analyses to assess each of these steps, i.e., acoustic channel, speech processing, intent extraction, and skill processing. For each step, I study VA-related features, potential design weakness, and corresponding defense and protection scheme. Overall, my research covers four important topics throughout four phases, which are mapped to the four steps mentioned above.

1.3.1 Phase 1: Defending Open Acoustic Channel with Acoustic Nonce.

One of the infamous problems of using VA is that the speech input to the VA can be captured via the open acoustic channel (step 1 in Figure 1.1), e.g., using a microphone stealthily. Then, an attacker can re-use the recorded speech to impersonate the original user. Existing speech re-use prevention methods either use machine learning liveness detection algorithms to passively identify unique signatures produced by loudspeakers in the recorded speech or rely on external devices to ensure the freshness of the speech. Such techniques are limited in terms of accuracy, data dependency, and device availability.

To address these limitations, I propose a VA security overlay called AEOLUS that proactively embeds a dynamic acoustic nonce at the time of user interaction, and detects the presence of the properly embedded nonce in the recorded speech to ensure freshness. I design and examine the practicality of AEOLUS by investigating how an acoustic nonce can be embedded and reliably and imperceptibly to a VA user. I model acoustic nonce propagation as an optimization problem with considerations of both reliability and imperceptibility, and determine the optimal parameters (i.e., acoustic nonce's operating frequency, amplitude, and bitrate) in light of these considerations from a practical perspective. I evaluate AEOLUS using a microphone loudspeaker pair in real-world settings such as an office, dining hall, and gas stations. My experimental results show that AEOLUS works effectively in both public and semi-public environments up to a distance of 4 meters.

1.3.2 Phase 2: Analyzing and Attacking Voice Assistant Speech Processing.

In the speech processing step of the skill pipeline, SRS is widely used in identity verification and personalized services. However, it is concerning that SRSs are vulnerable to adversarial attacks. Existing attacks often apply a global perturbation strategy and end up with non-trivial noise addition. This greatly affects the practicability of launching real-world attacks. To start tackling the limitation of global perturbation, I propose a novel one pitch attack that perturbs only the most effective audio frames. The attack consists of two main components: a black-box attention es-

timation module and a stagano-filter inspired by audio steganography. As a result, within a few hundred queries, one pitch attack could achieve an attack success rate of up to 99% when targeting SRSs in the overlay mode. Also, it introduces only minimal imperceptible noise addition. The evaluation results show that this attack is model-agnostic and useful in boosting the efficiency and imperceptibility of traditional adversarial attacking algorithms.

1.3.3 Phase 3: Analyzing and Securing Intent Extraction of Voice Assistant Application.

I further study the intent extraction step of the skill pipeline. While previous works such as hidden voice attacks [10] mainly examine the problems of VA services' default ASR component, I analyze and evaluate the security of the succeeding component after ASR, i.e., NLU, which performs semantic interpretation (i.e., text-to-intent) after ASR's acoustic-to-text processing. In particular, I focus on NLU's Intent Classifier, which is used in customizing machine understanding for third-party VA Applications. I find that the semantic inconsistency caused by the improper semantic interpretation of an Intent Classifier can create the opportunity of breaching the integrity of skill processing when attackers delicately leverage some common spoken errors.

Specifically, I design a linguistic-model-guided fuzzing tool, named LipFuzzer, to assess the security of Intent Classifier and systematically discover potential misinterpretation-prone spoken errors based on skills' voice command templates. To guide the fuzzing, I construct adversarial linguistic models with the help of Statistical Relational Learning (SRL) and emerging Natural Language Processing (NLP) techniques. In evaluation, I have successfully verified the effectiveness and accuracy of LipFuzzer. I also use LipFuzzer to evaluate both Amazon Alexa and Google Assistant platforms. I have identified that a large portion of real-world skills is vulnerable based on my fuzzing result.

1.3.4 Phase 4: Analyzing Security and Privacy Risks of Skill Processing.

As the last step in the skill pipeline, the skill processing also introduces security and privacy risks related to skills' user resource handling. A third-party skill may access user resources (e.g., contact information) in three ways: skill permissions, account linking, and skill inputs/outputs. To

use a skill properly, Alexa users often need to grant permissions to third-party skills and allow them to access the requested resources in one or more aforementioned ways. For example, a skill may obtain a user's location information by requesting `Device Address` skill permission, linking to the user's Amazon account with a home address configured, or asking the user directly via skill inputs/outputs. While different types of skill security indicators (e.g., skill permission prompts) are observed to warn users against the potential risks of granting these permissions, little is known on how effective these indicators are themselves. I take an in-depth look at the skill security indicators and test their effectiveness in a two-fold user study: a user survey with 150 Alexa users and a skill experiment with 30 Alexa users.

The results from both the user survey and skill experiment indicate low attention rates to the skill security indicators, especially for the ones related to Alexa's unique VUI design. For example, no Alexa users noticed the security indicators regarding dynamic content in skills, and users often misunderstood Alexa-specific capabilities such as `Alexa Reminder`. Furthermore, I find many Alexa users perceive third-party skills as Amazon-owned applications. This results in inappropriate trust in third parties, which hence leads to potential confused deputy problems. These findings depict the fact that the skill security indicators do not help most Alexa users make correct security decisions in a VUI environment. As a result, Alexa users may engage with policy-violating or even malicious skills without a proper understanding of the underlying risks. Based on my findings, I also provide recommendations for improving user attention and comprehension and identify directions for future work.

1.4 Summary of Contributions

In this section, the contributions of this dissertation are highlighted as follows:

Phase 1. I aim to tackle the problem speech re-use in the first step of skill pipeline. Specifically, I propose to use acoustic nonce embedding to detect and prevent the re-use of SRS-related user interactions with skills.

(i) I design a skill security overlay called AEOLUS to ensure freshness of input speech without any extra hardware dependency.

(ii) Modeling acoustic nonce propagation as an optimization problem to address 1) reliability, i.e., ensuring successful embedding and retrieval of dynamic acoustic nonce without impacting VA functionality, and 2) imperceptibility, i.e., to have minimal impact on VA users' experience.

(iii) Comprehensive real-world evaluation to show that AEOLUS can work effectively (0.5% false reject rate (FRR) and 0% false accept rate (FAR) in detecting speech re-use) up to a range of 4 meters in three different environments.

(iv) User study with 120 subjects to demonstrate that the embedded acoustic nonce does not degrade the user experience.

Phase 2. I study efficient adversarial attacks and target SRS of speech processing step in the skill pipeline. Moreover, the imperceptibility property of generated audio adversarial examples is also considered.

(i) I propose a novel black-box attention-based adversarial attack, named one pitch attack, to fool SRSs efficiently and imperceptibly. This attack can work both in the default and overlay mode.

(ii) To help locate the focused perturbation area, I design a multi-head attention estimation module that greatly reduces 1) the required model queries, 2) perturbation areas when launching the adversarial attack.

(iii) I apply various audio steganography techniques to further ensure the imperceptibility of adversarial example generation.

Phase 3. I study the widely existing misinterpretation issues that may be caused by the intent extraction step in the skill pipeline. The goal is to pinpoint the root cause of such misinterpretation problems and start to mitigate them.

(i) I analyze the semantic inconsistency of speech interpretation in skills, and uncover that this problem is deeply rooted in NLU's Intent classifier.

(ii) I model existing voice command errors to enable systematic analysis. More specifically, with SRL and NLP techniques, my modeling process can convert linguistic knowledge into computational statistical relational models.

(iii) I design an automated linguistic-model-guided mutation fuzzing scheme, named Lip-

Fuzzer, to assess the Intent Classifier at a large scale. I have published the source code and associated linguistic models to help fortify the security of skill processing.

Phase 4. I find that many of the risks in using third-party skills are not well understood by the users. Hence, I design a user study to investigate whether users perceive these risks properly. The results can help understand the potential problems of current skill processing design in the skill pipeline.

(i) I demystify how skill security indicators are designed to warn Alexa users against the risks of installing and using third-party skills. Three types of skill security indicators are identified based on how third-party skills can request access to user resources: skill permissions, account linking, and skill inputs/outputs.

(ii) I scrutinize the effectiveness of skill security indicators with a two-fold user study. The result reveals a worrisome fact that most users do not understand the risks incurred from third-party skills due to a flawed and poorly executed security indicator design.

(iii) I recommend changes to skill security indicators to improve their effectiveness. Several open challenges are also introduced to shed light on future research in securing skills.

2. BACKGROUND

In this chapter, I introduce background knowledge that is related to VA and VA application pipeline.

2.1 Voice Assistant

VA is becoming increasingly popular for conducting a variety of tasks ranging from internet browsing to online commerce. In fact, it is predicted that by the end of 2020, 50% of all searches performed online will be driven by voice [11]. VA systems are widely used in both smartphone and Internet of Things (IoT) platforms. Major smartphone vendors (e.g., Apple and Samsung) provide native support for VA, e.g., Amazon's Alexa or Apple's Siri, in their operating systems and make them available as independent applications. These can be used to instruct third-party applications to perform specific tasks, e.g., to turn lights on/off.

A VA is operated by a user using voice commands. The service processes the voice command and a response is returned either verbally to the user (e.g., announcing the time) or by performing a specific action (e.g., turning lights on/off). Devices that use VA are triggered by a user using a pre-specified phrase (e.g., Alexa, Hey Siri) and operate in a turn-taking mode. The device actively listens to the user once triggered, and the user listens to the device when it responds.

A typical VA can have up to five different sub-components, (i) an input/output voice channel to listen to a user's command and send verbal responses, (ii) a text/speech processing engine to convert speech-to-text and text-to-speech, (iii) a speaker verification/identification module to only permit authorized users to execute privileged commands, (iv) a text/intent processing engine to extract voice commands' semantic meaning, and (v) an application backend processing unit to execute/respond to the voice commands. In this dissertation, we apply a 4-step model as mentioned above. Specifically, we merge (ii) and (iii) to be the speech processing of the skill pipeline.

2.2 Intent Extraction

In the intent extraction step, a classification tree is a result of aggregating a VA system's built-in services' voice commands (e.g., voice search and time query) and developer-defined voice command templates created through Interaction Model [12]. In this dissertation, my focus is to analyze how malicious third-party developers can affect an Intent Classifier. Thus, using high-level examples[1], I introduce how developers contribute to building the intent classification tree in two parts: developer-defined built-in intent generation, and custom intent definition. First, to generate built-in intents for a skill, developers need to define the installation and invocation names.

```
1   #1.Developer-defined
2   <@\textcolor{blue}{"skill Installation Name":}@>
3   "The True Bank Skill",
4   <@\textcolor{blue}{"skill Invocation Name":}@>
5   "True Bank",
6   #2.Auto Generated Intents
7   <@\textcolor{blue}{"SYSTEM.InstallIntent":}@>
8   {"Alexa, enable The True Bank Skill.",
9   "Alexa, install The True Bank Skill."},
10  <@\textcolor{blue}{"SYSTEM.LaunchIntent":}@>
11  {"Alexa, open True Bank.",
12  "Alexa, ask True Bank to ... ."}
```

Listing 2.1: Example template with built-in Intent.

As shown in Listing 2.1, an example skill template (defined through programming an Interaction Model) is defined with an installation name at Line 2-3, and the invocation name is defined at Line 4-5. The VA system will then automatically generate the voice command templates (from Line 6 to 12) to update the classification tree. For instance, Amazon Alexa typically follows a format like "Alexa, install `Installation Name`" for installing a skill. Other built-in intents, such as `SYSTEM.CancelIntent`, `SYSTEM.HelpIntent`, are tied to default words (e.g. "stop",

[1] Although low-level implementations of the classification process can be different, based on my study, both Amazon Alexa and Google Assistant's high-level architectures are almost the same.

"cancel") with no requirement for developer involvement.

```
1    #3 Custom Intents
2    <@\textcolor{blue}{"CUSTOM.BalanceQueryIntent":}@>
3    {"Tell me my balance",
4    "Alexa, ask True Bank about my account balance.",
5    "What is my balance?"},
6    <@\textcolor{blue}{"CUSTOM.TransferIntent":}@>
7    {"Alexa, ask True Bank to transfer money to <@\textbf{Alice}."@>,
8    "I want to send money to someone."}
```

Listing 2.2: Example Template with Custom Intent.

Then, as shown in Listing 2.2, a developer can also define customized intents with associated voice command templates. For example, a developer can define a custom intent `CUSTOM.BalanceQueryIntent` (at Line 2) with corresponding voice command templates (at Line 3-5).

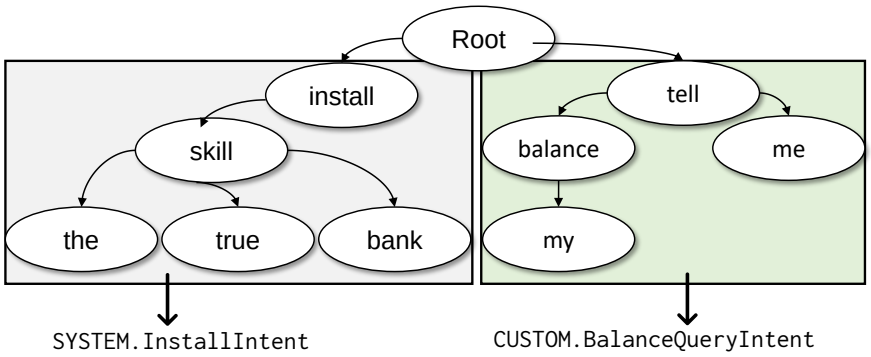


Figure 2.1: Intent classification tree example

In Figure 2.1, I show a simplified classification tree based on the dependency relationship of NLP-processed voice commands. The left nodes are generated from a built-in installation voice command, i.e., "install the True Bank Skill", defined at Line 7-9 in Listing 2.1. The right nodes are generated from a custom voice command listed in Listing 2.2. With more skill Interaction Models aggregated into the tree, additional nodes will be added. Any voice command input in a

VA platform will be processed with this tree.

Slot Values. A slot is a value associated with intent. Slots can be associated with intents by using tags. For example, with the "transfer money" voice command mentioned above, a developer can define a voice command with a slot type. For example, "Alexa, ask True Bank to transfer Money to *Alice*." (at Line 6-8 in Listing 2.2). "Alice" can be defined as `Contacts` slot value that is associated with `CUSTOM.TransferIntent`. This `Contacts` uses default slot type `SYSTEM.US_FIRST_NAME` which will force the NLU to automatically match common people's names used in the US, and hook the name value with intents. Slot values are usually important in processing voice commands as they contain the key data for deciding the next step. For example, in `SYSTEM.InstallationIntent`, a unique ID related to the installation name is associated with the slot value to decide which skill to install.

Fuzzy Matching. Another important feature enabled in NLU is the fuzzy matching of users' intent. It can tolerate minor LAPSUS. For example, a voice intent for launching "True Bank" can be interpreted correctly with the voice command "open True Banks" or "opening True Bank". Note that Google Assistant always applies fuzzy matching, but Amazon Alexa only enables this feature after a skill is installed. Hence, in Amazon Alexa, a skill Installation Name has to be spoken precisely with minimum tolerance for any LAPSUS.

2.3 Voice Payment

Voice-driven Payments. VAs are being increasingly used for payments, especially in retail environments [13]. Amazon and Exxon recently announced voice-driven payments at over 11,500 gas stations in the US [14]. A recent user study [15] indicates that while most users are comfortable conducting low-value purchases such as ordering a meal and shopping for groceries using voice-driven payment services, the majority of them do not have sufficient confidence in conducting high value purchases due to security concerns.

Most voice-driven payment services authenticate users through speaker verification. However, they often lack adequate protection mechanisms against speech re-use attacks where an attacker records a user interaction and re-uses it to attack the service either on-site or remotely. This is

analogous to card skimming [16] where an attacker re-uses a victim's credit card information.

Workplace Automation. Various digital assistants are being used for unsupervised or semi-supervised interactions in workplace environments [17]. In addition to accurate speaker recognition (identification and/or verification), it is important to have adequate protection mechanisms against speech re-use to protect sensitive resources in such environments [18].

2.4 Attention Mechanism

In traditional Recurrent Neural Network (RNN) models [19] such as sequence-to-sequence architecture, there is a known bottleneck that a single vector must capture the complete sequence of information. This creates problems of encoding long-range dependencies and memorizing the information since the beginning of a sequence. The advancement in attention-based [20, 21] models helps solve these problems by having the model focus on the more relevant parts of the input sequence for each output. Also, the attention mechanism introduces a new way to alleviate the vanishing gradient problem by providing a direct path to the inputs.

Self-attention. There are different forms of attention mechanisms. Self-attention, or intra-attention, is an attention mechanism aiming to compute a representation of the sequence by relating different positions of this sequence. Self-attention has been used successfully in various tasks, including question answering, reading comprehension, text summarization, etc.

In SRSs, given the speech signal, attention mechanisms are applied to extract a sequence of speaker embeddings from short segments. One of the commonly used architectures is to use self-attentive pooling [22, 23] for both text-independent speaker verification and other speaker recognition tasks. The goal of using self-attentive pooling is to use a trainable and more adapted layer for pooling. This method is shown [23] to be more scalable in SRSs than vanilla temporal averaging. Specifically, self-attentive pooling adopts a trainable layer to assign a weight over each representation of the sequence. As a result, the utterance level representation is obtained through the respective weighted average of these representations.

Multi-head Attention. However, the self-attention model has limitations, such as the global consideration of the weights. This means that each element has a limited weight-based interpre-

tation. To gain greater power of encoding multiple relationships and nuances for each element, a multi-head attention model is proposed in the Transformer design [20] which takes this one step further. In the Transformer design, the idea is that sequential models can be dispensed entirely. Moreover, the outputs can be calculated using only attention mechanisms.

2.5 Audio Steganography

Audio steganography is a technique used to transmit hidden information by modifying an audio signal in an imperceptible manner. It can hide secret information in a hosting or carrier message (e.g., text or audio). One of the features of steganography is to ensure that the host message before and after steganography embedding has the same characteristics.

Auditory Masking. The goal of applying steganography in phase 2 is to ensure the minimal human perception of adversarial example embedding. This means that the transmission of secret information (e.g., secret information encoding and decoding) is not considered. To achieve the goal of imperceptible audio adversarial example embedding, I apply a technique called auditory masking. In auditory masking [24] theory, if an embedded signal has similar or lower frequencies or intensity, human ears can barely differentiate the host and embedded signals. I learn lessons from two popular audio auditory masking methods in this research: echo hiding and spread spectrum. In echo hiding, the same or similar acoustic signal is embedded around the host signal to achieve this goal. Second, stealthy signal embedding can be done both simultaneously (i.e., simultaneous masking) or temporally separated (i.e., temporal masking) from the host or masking signal. Spread spectrum [25] is applied by spreading hidden data through the frequency spectrum. The origin of this technique is to ensure proper recovery of a signal sent over a noisy channel by producing redundant copies of the data signal [26]. In detail, the data are multiplied by an M-sequence code shared by both sender and receiver, then hidden in the host audio. Thus, if noise corrupts some values, the receiver will still have sufficient copies of each value to recover the hidden message. For audio steganography, the conventional direct sequence spread spectrum [27] technique was initially used to hide confidential information in MP3 and WAV signals. This approach applies a frequency-mask-based method [25] to ensure the quality of the embedded signal. Specifically, the

spread spectrum in the sub-band domain is combined with phase shifting to increase the robustness of transmitted data against additive noise and allow easy detection of the hidden data.

3. LITERATURE REVIEW

In this chapter, I review necessary related work to help understand the state-of-the-art research in VA security.

3.1 Audible and Inaudible Voice Command Attacks

First, for audible voice command attacks, Diao et al. [28] proposed to play prepared audio files using non-hidden channels that are understandable by a human listener. Tavish et al. [29] then presented Cocaine Noodles that exploits the difference between synthetic and natural sound to launch attacks that can be recognized by a computer speech recognition system but not easily understandable by humans. To achieve a better result, a white box method [10] was used based on knowledge of speech recognition procedures. With complete knowledge of the algorithms used in the speech recognition system, this improved attack guarantees that the synthetic voice commands are not understandable by a human. For the threat model, Audible Voice Command Attacks require the attackers' speakers to be placed at a physical place near victims' devices ([10] fails with distances over 3.5 meters). Also, CommandSong [30] was proposed to embed a set of commands into a song, to spread to a large amount of audience.

Second, more powerful attacks using inaudible voice commands were proposed in [31] [32]. They leverage non-linearity of microphone hardware used in almost all modern electric devices. Humans are not able to recognize sound with frequency over 20 kHz, and microphone's upper bound of recordable range is 24 kHz. For this threat model, Inaudible Voice Command Attacks also require the attackers' devices to be physically close to the victims' devices (in feet range).

Third, Kumar et al. [33] presented an empirical study of skill squatting attacks based on speech misinterpretation. Moreover, as a concurrent work, [34] also showcases a similar approach to exploit the way a skill is invoked, using a malicious skill with similarly pronounced name or paraphrased name to hijack the voice command meant for a different skill.

3.2 Speech Re-use

Speech Replay Detection Existing research related to speech replay detection can be classified into two categories, software-based and hardware-based methods. Software-based methods use machine learning to determine whether the input speech is produced by a human or replayed using a recording device [35] [36] [37]. Chen et al. [35] present a method that obtains 0% EER for distance (between user and VA input device) of only a few centimeters which limits its applicability in practice. Ahmed et al. [36]'s method, on the other hand, is reported to work well upto a distance of 2.6 meters. However, the EER varies significantly in different environments with it being as low as 11.6%. One of the best performing methods [37] reports EER of 6.7% on ASVSpoof 17 database. A caveat, however, is that these methods aim to address general replay attacks. In contrast, AEOLUS aims to prevent speech re-use and obtains 0% EER upto a distance of 4 meters in three different environments.

The second category of approaches use additional hardware (e.g., a vibration sensor on user's neck) to check that speech is produced in real-time by a user [38] [39]. Such approaches, in general, outperform machine learning-based methods. For example, Feng et al. [38] report 97% successful detection rate, on average, with no strict assumption on distance. However, there are two major practical limitations using the aforementioned approaches for detecting speech re-use, (i) requirement of extra hardware which has cost implications, and (ii) inconvenience to users.

Audio Watermarking. Audio watermarking is widely used for multimedia copyright protection [40, 41]. Both time domain [42] and frequency domain [43, 44] watermarking techniques have been proposed in the literature. Typically, audio watermark embedding and detection is self-synchronized [45]. Advanced watermarking methods use spread-spectrum modulation [40] to prevent watermark removal and provide imperceptibility [43, 46]. While useful, these techniques cannot be directly used in VA operational setting. They are designed for fixed length offline audio files, e.g., music recordings in CDs [40], and do not consider the impact of environmental factors, e.g., bitrate variability and background noise, in over-the-air transmission. Such environmental factors result in a dynamic and lossy acoustic environment in turn making watermark embedding

and retrieval significantly challenging. Further, they are not designed for real-time speech and speaker recognition systems where input speech is unknown apriori and can be of variable length. AEOLUS is designed to take these pragmatic considerations into account. It is able to successfully embed and retrieve acoustic nonce in the voice channel without degrading user experience in practice.

3.3 Warning Research

Researchers have been studying various warnings/indicators in different platforms such as browsers, websites, mobile apps. A common model, known as the Communication-Human Information Processing (C-HIP) model [47], is often used to learn how humans process warning messages. C-HIP formalizes the steps between warning message delivery and users' final behavior. In the C-HIP model, users are assumed to act quickly upon the warning if an appropriate warning design is provided. This model has been adopted to learn different security indicators, including web [48] and mobile [49]. The steps being studied are usually different among different targeted platforms. For example, a seven-step approach is used in studying web browser phishing warnings [48]. Felt et al. [49] proposed to use a C-HIP-oriented five-step model; however, the results indicate that only the most critical three steps (i.e., attention, comprehension, and behavior) matter if low attention and comprehension rates are observed. A similar abstraction is also presented and discussed by Laughery et al. [50]. In this dissertation, I learn from these work [49, 50] and apply a similar three-step model in the user survey to scrutinize skill security indicators.

4. DEFENDING OPEN ACOUSTIC CHANNEL WITH ACOUSTIC NONCE*

4.1 Introduction

In this chapter, as shown in Figure 4.1, I focus on speech re-use issues in the acoustic channel of the skill pipeline. This is an important problem because VA such as Amazon Alexa and Google Assistant can now perform tasks like smart home control, hands-free online commerce, as well as automated voice response systems for customer support. While this increasing ubiquity can be primarily attributed to improved real-world performance of deep learning driven speech and speaker recognition, security is an equally important consideration in operational settings. To secure VA in practice, one of the common security mechanisms used is "voice password". For example, applications such as Samsung Bixby, Wechat [51, 52, 53] prompt users to speak a password and perform two factor authentication by checking (i) if the spoken passphrase is correct, and (ii) it was spoken by the authorized user.

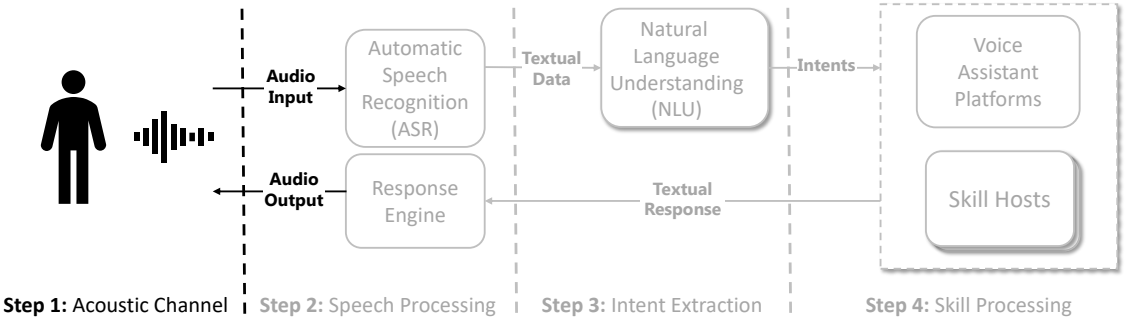


Figure 4.1: Skill Pipeline - Step 1

Voice passwords, however, do not provide security against voice spoofing techniques such as speech replays [54, 39, 55] and synthesis [56]. While there are known practical limitations in

* Reprinted with permission from "Practical Speech Re-use Prevention in Voice-driven Services" by Yangyong Zhang, Maliheh Shirvanian, Sunpreet Arora, Jianwei Huang, Guofei Gu, 2021. The 24th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '21), Copyright 2021 by Zhang et al., publication rights licensed to ACM.

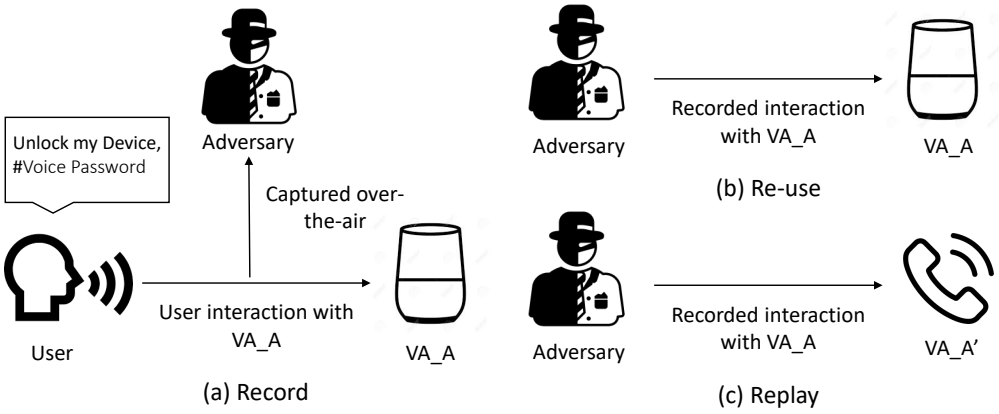


Figure 4.2: Difference between re-use and generic speech replays.

conducting speech synthesis attacks (e.g., the attack proposed in [56] requires 24 hours of high quality training data), it is easier to record and replay user speech in practice. In this chapter, I aim to address a subset of malicious speech replays, called *speech re-use* attacks. I define speech re-use attacks to be specific to a certain type of VA. As shown in Figure 4.2, let A and A' be the VA types for the capture and attack steps, respectively. Speech replays typically consider a variety of VA types for capture and attack steps (A = A' and A $\neq$ A'), while re-use attacks are a subset of such replays where the VA types in consideration for the capture and attack steps is similar (A = A'). An example of a re-use scenario is replaying a recorded passphrase to fool the authentication process of an Amazon Alexa service, and then conduct online shopping or controlling a smart home device. A generic speech replay assumes this passphrase can be used for other VA which is not a practical assumption because such passphrases are supposed to be VA-specific or device-specific. For example, reciting the phrase "Hey Siri" will not trigger an Amazon Alexa service.

Existing defense mechanisms can be categorized into two classes. The first class of methods use machine learning to determine whether a speech sample is produced by a human or replayed using a playback device [35, 57]. The second category of techniques use external devices (e.g., a vibration sensor) to check if the speech is produced by a human in real-time [38]. For machine learning techniques, real world performance usually depends on training data and its relevance to the test environment. A state-of-the-art method Void [36], as an example, has significantly

different equal error rates (EER), 0.3% and 11.6%, on two speech datasets collected in two different environments. Another practical limitation is unreliability of playback device signatures used by machine learning techniques [57]. In addition, for methods that use extra hardware, there are usability and cost implications for practical deployment.

I present AEOLUS[1], a security overlay to prevent re-use attacks on protected VA in a proactive and device-agnostic manner. In detail, the proposed security overlay proactively embeds a dynamic acoustic nonce in the voice channel via the microphone at the time of user interaction, and detects the presence of the embedded nonce in the speech recorded by the speaker to ensure speech freshness. For example, if the security overlay is integrated with each Alexa's VA, it can prevent speech captured from user interaction with one Alexa's VA from being re-used on another Alexa's VA. This is similar to the device specific random pattern used in Face ID to counter digital and physical spoofs [58] (see Figure 4.3). AEOLUS is designed as a software solution, and can be integrated by a vendor with any closed-loop VA that involves real-time user interaction. It is useful for providing additional security in critical applications such as user authentication and payments. AEOLUS does not require any extra hardware and works with in-built speaker and microphone in consumer products. AEOLUS is secure by design against attacks that remove the embedded nonce for speech re-use. It uses Frequency-hopping spread spectrum (FHSS) [59] [60] technique for dynamic acoustic nonce embedding. Similar techniques have been previously used to watermark audio recordings for copyright purposes [40] [42]. However, their application in offline audio watermarking is fundamentally different from VA operational setting where the length and content of the input speech is unknown apriori. AEOLUS is designed for real-time over-the-air use without prior knowledge of the input speech.

AEOLUS addresses two key practical challenges. The first challenge is to reliably embed an acoustic nonce over-the-air and retrieve it successfully from the recorded speech. Over-the-air acoustic nonce propagation is affected by factors such as background noise and distance between microphone and loudspeaker [61]. The latter, for instance, results in distorted and attenuated

[1]Named after Aelous, the "Keeper of the Winds" in Greek mythology.

speech signal which increases the likelihood of errors while extracting the embedded acoustic nonce. The second challenge is to embed the acoustic nonce such that it is imperceptible to a VA user. Achieving imperceptibility is non-trivial in light of the first challenge. This is because an important consideration for reliability is embedding acoustic nonce of certain strength which, in turn, makes the acoustic nonce perceptible. Therefore, I model acoustic nonce generation as an optimization problem and compute the set of optimal parameters, e.g., nonce's operating frequency, amplitude, bitrate in different environments using differential evolution.

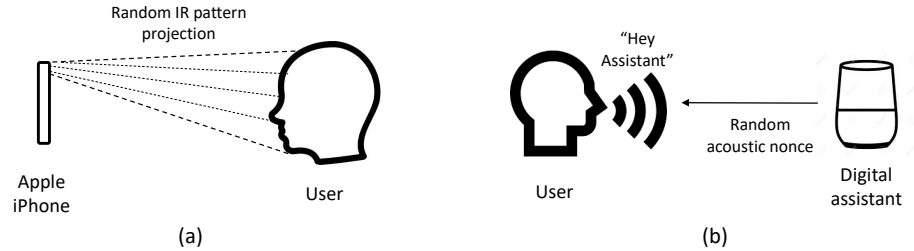


Figure 4.3: Face ID's device-specific random IR pattern projection and AEOLUS using a random acoustic nonce.

I evaluate AEOLUS in three different environments and show that it works reliably upto a range of 4 m. Additionally, I conduct a user study involving 120 subjects (approved by the institutional human study review board) to evaluate the imperceptibility of the embedded nonce. The results of the study show that majority of users find the embedded acoustic nonce to be either imperceptible or non-disruptive.

4.2 Proposed Security Overlay

In this section, I present the AEOLUS design by first introducing its threat model including both adversary capability and hardware assumptions. Next, as shown in Figure 4.4, I illustrate the core design of AEOLUS which consists of VA-compatible acoustic nonce generation, embedding, and detection.

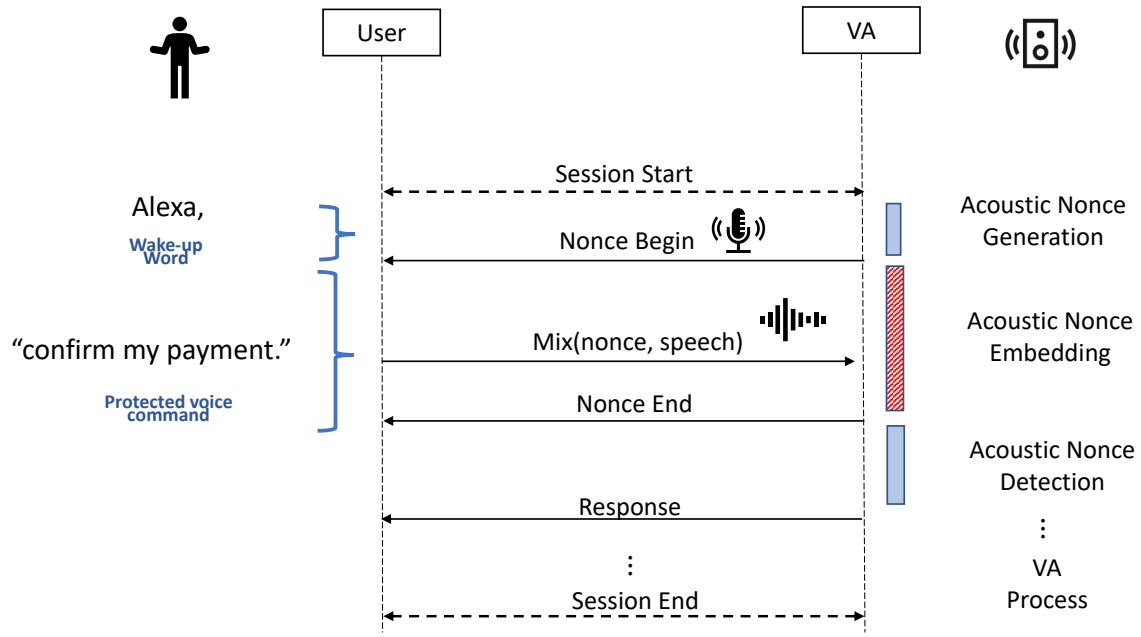


Figure 4.4: Using AEOLUS to prevent speech re-use.

4.2.1 Threat Model

At the time of user interaction, the method used to trigger a VA, e.g., words or phrases like "Alexa" and "Hey, Siri" are not protected but can initiate the proposed overlay to prevent re-use of voice commands and confidential passphrases. For example, to prevent re-use of a user's voice password to access recent account transactions when the user is interacting with a voice banking VA [62].

The following assumptions are made while designing AEOLUS to address the speech re-use attack scenario.

- **Adversary Capability.** We aim to address speech re-use in the context of VA protected with AEOLUS. An adversary uses a commodity recording device to record prior user interactions with a AEOLUS-enabled VA, and re-use the recorded interaction to attack a AEOLUS-enabled VA. Hence, attack scenarios where an adversary can record or access clean user speech samples (i.e., recordings without embedded acoustic nonce) from a different con-

text to launch replay or speech synthesis attacks are not considered. The key underlying assumptions are that (i) users typically interact with VA using specific commands (e.g., a user-defined passphrase [51]) and these commands can be protected with AEOLUS, and (ii) it is difficult to record clean samples of these commands from interactions with different VA without AEOLUS protection or in human conversations. Also, it is assumed that AEOLUS can be used in conjunction with other defense mechanisms for replay or speech synthesis attacks.

- **Hardware Assumptions.** We study the problem of speech re-use prevention wherein speech is presented over-the-air. It is assumed that the hardware device (loudspeaker and microphone) used by a user to interact with VA is trusted, secure and functioning, and an adversary cannot disable the speaker or the microphone. Furthermore, I assume that the channel between hardware and other VA modules is secure/protected. Similar hardware assumptions are made by other software-based security mechanisms such as CAPTCHA. The proposed framework does not address hardware integrity which can be addressed using other security mechanisms, e.g., hardware attestations.

4.2.2 Core Components

To prevent speech re-use in VA, AEOLUS uses three core components, nonce generation, nonce embedding and nonce detection.

Nonce Generation. The nonce generation module is invoked upon initiation of a new user interaction session (see Figure 4.5 ①). Akin to nonce-based authentication [63, 64], the module generates a digital nonce (a random number of a length L) that is valid for the current session. Nonce length L should be sufficiently large to prevent use of duplicate nonces in different sessions. Given limited bandwidth of over-the-air channel and estimated user base for popular VA (e.g., Alexa skill has 300,000 daily users [65]), the minimum permitted value of L is set to 32 bits. 32-bit long nonce is sufficient to protect short speech samples (e.g., a voice passphrase) in a particular session. Nonce repetition is permitted to protect longer speech samples during the session. The nonce δ

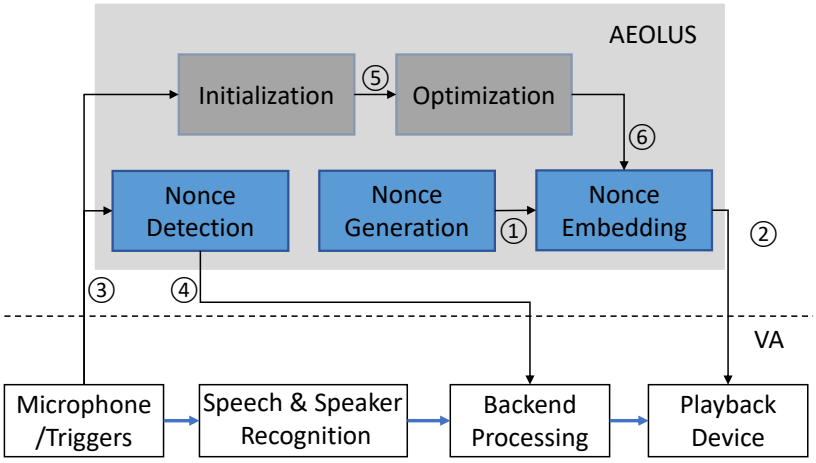
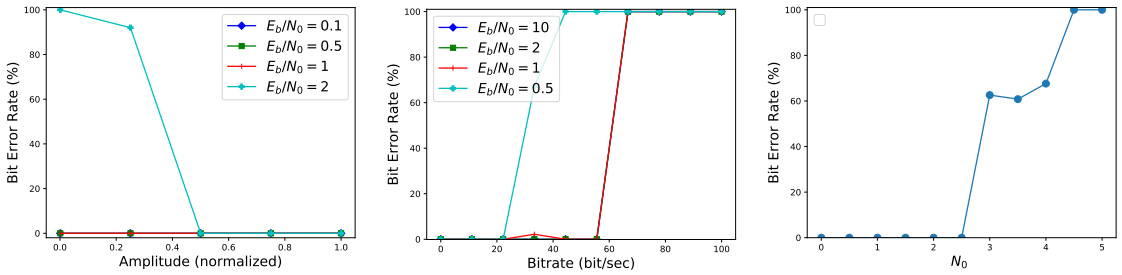


Figure 4.5: AEOLUS-enabled VA components with the associated data flows.

for session S is stored in a set S_{set}. S_{set} is subsequently used by the nonce detection module to determine if the embedded nonce is current.



(a) Frequency range = 4-5kHz, hopping = 2, bitrate = 30 bits/sec

(b) Frequency range = 4-5kHz, hopping = 2, amplitude = 1

(c) Frequency range = 4-5kHz, hopping = 2, bitrate = 30 bits/sec, amplitude = 1

Figure 4.6: Variation of bit error rates.

Nonce Embedding. Post nonce generation, an encoding algorithm, Binary Frequency Shift Keying (BFSK) [66, 67], is used to encode the generated digital nonce as an acoustic nonce. Each bit is independently encoded as a cosine wave ω_i. Bit "0" is encoded as a 1kHz cosine wave, and bit "1" as a 5 kHz cosine wave. All cosine waves in the set ω are concatenated to yield the acoustic nonce δ. The acoustic nonce δ is embedded in over-the-air channel using VA playback device (

Figure 4.5 ②). An important parameter for nonce embedding is the time duration of the embedded acoustic nonce relative to the time duration of user interaction. Using a real-world dataset with 27,000 samples [68], I estimate that the average user interaction is 3 sec. As an example, let the minimum duration of embedded nonce be half of the average user interaction. In this case, the acoustic nonce embedding would require a bitrate of 14 bits/sec, and the maximum duration of each component cosine wave w_i can be 42 ms. Note that this estimate does not consider the idle time in user interaction. Additionally, to prevent an adversary from obtaining clean speech sample by removing the embedded nonce, the nonce embedding module leverages Frequency-hopping Spread Spectrum (FHSS) [60, 59]. FHSS is a well-known technique that provides high robustness against standard embedded signal removal attacks. For high robustness, the module uses FHSS with a set of different frequencies and periodically selects the operating frequency at random.

$$\arg\min_{\mathcal{P}} f(\mathcal{P}) \text{ subject to } g(h(\mathbf{x} * \delta)) - g(h(\mathbf{x})) \leq \theta \tag{4.1}$$

Nonce Detection. Once user interaction ends, the recorded audio is processed using the nonce detection module (Figure 4.5 ③). The module decodes the acoustic nonce using BFSK, and checks that (i) the nonce is completely decoded, and (ii) the decoded nonce is current. The recorded speech is subsequently processed using standard VA modules (Figure 4.5 ④). If nonce detection fails or if the decoded nonce has a high Bit Error Rate (BER), a recourse action (e.g. a retry or reject) can be taken. Furthermore, if speech is re-used by an adversary and both the current nonce and the obsolete nonce from re-used speech are detected, information about re-used speech such as when and where it was recorded can be determined from S_{set}. Another scenario can be signal interference between the current and obsolete nonce resulting in inappropriate decoding result. In such case, pertinent recourse action can be taken.

4.3 Practical Realization

In this section, I first illustrate two key challenges while implementing AEOLUS in practice: reliability and imperceptibility. Then, I discuss the root causes of these challenges. Lastly, I present an optimization scheme to tackle the challenges.

4.3.1 Challenges

Reliability. The first challenge is to reliably embed an acoustic nonce in over-the-air channel, as well as to accurately detect nonce from the recorded speech. Because over-the-air acoustic nonce transmission is impacted by factors such as background noise and distance between microphone and playback device, it is difficult to achieve this in operational settings. It is also important to determine if any previously embedded acoustic nonce is present in the recorded speech to ascertain speech re-use. The metric used to measure reliability is the bit error rate (BER) between the embedded and detected nonce. Recall that BFSK is used for acoustic nonce embedding and detection. Under the assumption that only additive white Gaussian noise (AWGN) is present in over-the-air channel, BER can be computed using the complimentary error function $erfc()$ [69] as follows:

$$BER = \frac{1}{2} erfc(\sqrt{E_b/N_0}) \tag{4.2}$$

Note that BER computation in Eqn. 4.2 involves the normalized per bit signal-to-noise ratio E_b/N_0 [70] which depends on the frequency f and amplitude α of the carrier wave. However, other types of noises besides AWGN are typically present in over-the-air channel. Hence, I conduct Room Impulse Response (RIR) simulation experiments to study the impact of carrier wave amplitude and frequency on BER.

Imperceptibility. The second challenge is to ensure that the embedded acoustic nonce does not degrade VA user experience. For this, the acoustic nonce should be as imperceptible as possible to a VA user. The nonce embedding and generation modules presented earlier do not adequately address this challenge because (i) they do not optimize any objective metric for imperceptibility, and (ii) they do not account for dynamic and noisy environments where it is difficult to achieve both reliability and imperceptibility simultaneously. For measuring imperceptibility, Sound Pressure Level (SPL) is computed using the following equation:

$$SPL = 2\pi^2 f^2 \alpha^2 \rho c / v \tag{4.3}$$

Here, f represents the frequency and α is the amplitude of the carrier wave. ρ represents the density of the transmission channel (e.g., over-the-air channel), and c denotes the speed of acoustic transmission. Given that the average SPL for human speech is 60dBm [71], f and α should ideally ensure that SPL is below this threshold for imperceptibility. Like Equation 4.2, while this equation provides a theoretical basis to understand SPL, I study how the aforementioned parameters impact SPL using RIR simulation.

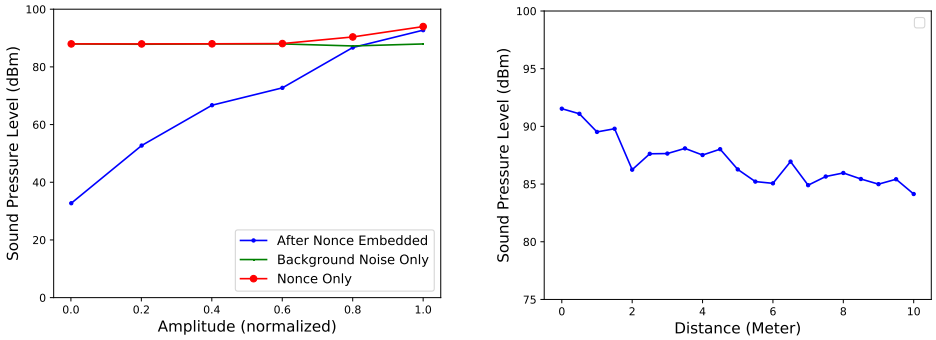
4.3.2 Key Parameters

There are two key parameters that impact reliability and imperceptibility: (i) acoustic nonce generation and embedding parameters that include frequency and amplitude of the carrier wave used for acoustic nonce generation and bitrate used for acoustic nonce encoding, and (ii) environmental parameters that include the distance between the microphone and user, the room size, and background noise among others. To understand the impact of the these parameters on reliability and imperceptibility, I implement AEOLUS as a prototype and setup an RIR environment called MCRoomSim in MATLAB [72]. RIR is a commonly used tool for estimating the performance of signal processing applications in over-the-air channel [61, 73].

4.3.2.1 Nonce Generation and Embedding Parameters

Amplitude. According to RIR simulation, a lower amplitude yields lower E_b/N_0 ratio and consequently, higher BER (see Figure 4.6a). This is consistent with Equation 4.2. Furthermore, as shown in Figure 4.7, SPL is proportional to the amplitude of the carrier wave.

Frequency. BER and SPL are evaluated in the frequency range at which typical speaker recognition systems operate (5 Hz - 8 kHz; sampling rate = 500 Hz). The other parameters are fixed as follows: E_b/N_0 = 1, frequency hopping = 2, amplitude (normalized) = 1, bitrate = 30 bits/sec, distance = 1 meter. It is observed that the carrier wave frequency neither impacts BER nor SPL. The later is in contrast to existing research [74, 75] which shows that the frequency of an acoustic wave affects the loudness as perceived by humans in a non-linear manner. Hence, the frequency to SPL



(a) E_b/N_0 = 1, carrier frequency range = 4-5kHz, hopping = 2, bitrate = 30 bits/sec, distance = 1 meter

(b) E_b/N_0 = 1, carrier frequency range = 4-5kHz, hopping = 2, bitrate = 30 bits/sec, Amplitude (normalized) = 1

Figure 4.7: Variation of sound pressure level.

mapping from IoSR toolkit [76] is used as reference.

Bitrate. Figure 4.6b shows the impact of acoustic nonce bitrate on BER. BER increases significantly as bitrate increases beyond 20 bits/sec. One of the causes is asynchronous signal overlapping in the time domain when two bits are transmitted too close to each other.

4.3.2.2 Environmental Parameters

Frequency Shifting. Over-the-air acoustic nonce transmission induces differences in the received frequencies and the transmitted frequencies called *frequency shifting*. Frequency shifting can also occur because of hardware limitations, e.g., a microphone with a 1.8 kHz frequency limit being used to capture a 2 kHz wave. Since the received frequency is used to decode the acoustic nonce, frequency shifting needs to be accounted for by AEOLUS. We investigate frequency shifting in the simulation setting (operating frequency range: 500 Hz-8 kHz, cardioid loudspeaker type, dipole microphone type) and determine the frequency shifting range to be +/- 15 Hz.

Distance. Distance between a VA user and VA input impacts imperceptibility of the acoustic nonce. Figure 4.7b indicates that this distance is inversely proportional to SPL.

Background Noise. Figure 4.6c and Equation 4.2 suggest that the background noise is proportional to BER. In an environment with high background noise, the nonce generation and embedding

parameters need to be configured appropriately (e.g., decrease bitrate, increase amplitude) to limit the BER.

4.3.3 Mathematical Formulation

To address reliability and imperceptibility simultaneously, I model over-the-air acoustic nonce transmission as an optimization problem. Let a user's recited speech be denoted by x, and the acoustic nonce embedded by AEOLUS be δ. Also, let the function that accounts for variations induced by over-the-air channel, e.g., due to acoustic signal air absorption and distance, be $h(\cdot)$. The speech recorded by the microphone is thus represented as $h(x)$. When δ is embedded at the time of user interaction with the VA, x mixes with δ. As a result, the microphone records $h(\mathbf{x} * \delta)$, where $*$ denotes signal convolution between x and δ.

Assume that AEOLUS detects nonce δ' in $h(x * \delta)$. Let us define an objective function $f(\cdot)$ denoting BER that takes the parameter set $\mathcal{P} = \{< f, \alpha, \tau >, ...\}$ as input, and a function $g(\cdot)$ denoting SPL. The goal is to find the optimal parameter set $\mathcal{P}$ that minimizes f (for reliability) subject to the constraint that the SPL difference between speech with and without embedded nonce is less than or equal to a threshold θ (for imperceptibility):

Algorithm 1: Computing Optimal Parameters

initialization:

1. Create candidate population: $\mathcal{P} = \{P < f, \alpha, \tau >, ...\}$

2. Initialize: $\mathcal{P}_{output} = \{\}$, $i = 0$, $MAX_i = 100$, $MAX_{output} = 20$, $\beta = 0.5$

2. Acquire environmental parameters: d, N_0

3. Set up Goals: $BER < 10^{-9}$, $SPL^{'} - SPL \leq \theta$

SPL : the average loudness value measured before the nonce embedding; $SPL^{'}$: the average loudness value measured during the nonce embedding

optimization:

while $i <= MAX_i$, $\mathcal{P}_{output}.Size <= MAX_{output}$ **do**

  foreach *individual* $P_i (i = 1, ...) \in \mathcal{P}$ **do**

    if P_i *satisfies the Goals* **then**

      $\mathcal{P}_{output} \leftarrow P_i$

    end

    Create candidate C from parent P_i, d, N_0.

    Evaluate the candidate C and acquire SPL_C and BER_C.

    if $SPL_C < SPL_{Parent}$ *and* $BER_C < BER_{Parent}$ **then**

      $P_{feasible} \leftarrow$ Candidate C

      Candidate C replaces the parent

    else

      the candidate is discarded

    end

  end

  Randomly select next P in $\mathcal{P}$.

end

output: $\mathcal{P}_{output}$

Candidate Creation:

input : P_{parent}, d, N_0, β

Randomly select individuals P_1 , P_2 from $\mathcal{P}$

Calculate candidate C as $C = P_{parent} + \beta(P_1 - P_2)$,

where β is a scaling parameter decided by d and N_0.

output: Candidate C

4.3.4 Computing Optimal Parameters

Initialization. The environmental parameters are first initialized (Figure 4.5 ⑤) using the VA input. The background noise energy is calculated by inspecting the ambient sound. Frequency shifting is calculated by transmitting and decoding a nonce that covers the working frequency range (500 - 8 kHz, sampled every 500 Hz). It is assumed that the distance between the user and

VA input device is set by the VA device vendor.

Optimization. Computing the gradients of the optimization parameters directly using gradient-based methods is inefficient [77]. This is because the VA's acoustic environment is dynamic (e.g., due to occlusions and varying background noise levels). In addition, the formulated optimization is a multi-dimensional problem with considerations of both reliability and imperceptibility. Hence, I use differential evolution (DE) [78, 79] which is a gradient-free method. DE generates diverse candidate solutions in each iteration which avoids local minima and aids in convergence to a global minima. It does not require apriori knowledge of the underlying acoustic encoding method or acoustic environment. Algorithm 1 summarizes the use of DE for obtaining optimal parameters. The initial set of candidate parameters $\mathcal{P}$ includes feasible frequency f, amplitude α, and bitrate τ, estimated using RIR simulation. Once AEOLUS is deployed, the initialization module first estimates the environmental parameters, e.g., N_0 and d. Subsequently, the optimization module uses the pre-calculated RIR parameters for efficient convergence to optimal parameters (see Section 4.4.1).

BER and SPL Calculation. The BER is calculated by comparing the generated nonce that is embedded over-the-air and the decoded nonce from the captured speech. With synchronization techniques [80] used in the nonce embedding and detection, AEOLUS is able to calculate how many error bits are received. For example, if bit sequence "0010" is transmitted and "0110" is received, BER is 25%. SPL is calculated using Equation 4.3. AEOLUS first calculates the frequency and amplitude of the captured speech. These parameters along with other constant values (e.g., density of transmission medium) are fed into Equation 4.3 to obtain SPL. In Algorithm 1, BER threshold of 10^{-9} is used because it is a commonly used threshold for a good BER in communication systems. Similarly, threshold value of 5% is used for determining small SPL difference.

Candidate Comparison. A candidate's parameter set is compared with its parent based on their performance, i.e., the resulting BER and SPL. If both the BER_C and SPL_C of a candidate is better than the parent's BER_{parent} and SPL_{parent}, the candidate is added to the feasible set and the parent is discarded. If not both of the BER_C or SPL_C of a candidate outperform BER_{parent}

and SPL_{parent}, the candidate is discarded.

4.4 Experimental Evaluation

(a) Conference Room. **(b)** Dining Hall

(c) Gas Station (Car). **(d)** Setup in the Car.

Figure 4.8: Experimental setup to evaluate AEOLUS in different environments.

AEOLUS is implemented as a prototype in MATLAB and evaluated in three different environments (see Figure 4.8): (i) a conference room (approx. 5.5 m by 4 m) selected as a reference for a semi-public indoor environment, (ii) a campus dining area (approx. 25m by 50m) chosen to mimic a noisy indoor public area, and (iii) inside a car parked at a gas station to simulate the use case of voice-driven payments at gas stations [14]. For (i) and (ii), a FIFINE K669B microphone is used to record speech, and a Logitech S120 Speaker System is used as playback device to simulate users. The evaluation is performed at different distances (0.5-4 m) between the microphone and playback device. For (iii), the microphone is identical but the car's in-built loudspeakers are used instead of the Logitech loudspeakers. Also, the evaluation is performed at two different distances 0.5 and 0.7

m.

AEOLUS is evaluated both on speech data captured from live participants in ReMASC dataset [68] and synthesized speech from Amazon Polly. A total of ten different speaker profiles are used (four of which are synthesized). For each profile, three different speech samples are used. Microsoft Azure's speech and speaker recognition services [81] are used for speech-to-text and speaker recognition. Text-independent speaker verification service [82] is used for speaker verification experiments. Each speaker's profile is first enrolled using a speech sample longer than 20 seconds. Verification is performed using the speaker's speech samples with and without acoustic nonce, and yields one of the following outputs: acceptance with "high", "normal", or "low" confidence, or a rejection.

4.4.1 Performance

Table 4.1: Optimal parameters for achieving reliability and imperceptibility.

Location	SPL (dB-A)	Frequency ($\pm$ 200 Hz)	Amplitude (normalized)	Bitrate (bits/sec)
Conference Room	41	4000 Hz	0.52	35
Dining Hall	58	5200 Hz	1	25
Gas Station (Car)	47	4800 Hz	0.61	33

Optimal Parameters. First, I estimate optimal parameters for achieving reliability and imperceptibility simultaneously in a given environment using Algorithm 1. The frequency step size is set to 200 Hz and θ is set as 0.1 to ensure that the generated nonce is below the average human speech SPL of 60 db. It is observed that lower amplitudes and frequencies and higher bitrates are comparatively optimal for acoustic nonce embedding in low noise environments, e.g., conference room (see Table 4.1). Relatively higher amplitudes and frequencies, and lower bitrates are more optimal in environments with higher noise, such as the dining hall. This is because the background noise in such environments typically vests as a low amplitude low frequency noise.

Table 4.2: Acoustic nonce recovery bit error rate (BER).

Location	Dis.	BER/FAR/FRR	Location	Dis.	BER/FAR/FRR	Location	Dis.	BER/FAR/FRR
Conference Room	0.5m	0%/0%/0%	Dining Hall	0.5m	0.2%/0%/0.5%	Gas Station (Car)	0.5m	0%/0%/0%
	1m	0%/0%/0%		1m	1.2%/0%/ 1%		0.7m	0.5%0%0.5%
	4m	1.2%/0%/0.5%		2m	75%/0% /78%			

Computational Overhead. Algorithm 1 iteratively optimizes the parameter set $\mathcal{P}$ for the given environment. For conference room, a single iteration is sufficient. However, for dining hall and gas station environments, a few iterations (< 10) are required and the computation takes a few seconds. This does not cause a delay at the time of user interaction because it overlaps with the time a user typically takes to set up a VA. The acoustic nonce embedding and decoding process occurs while user is interacting with VA. The computational overhead is negligible because the nonce size is small, and the playback device is used only when the user is speaking.

Reliability Next, I measure the reliability of embedding and retrieving acoustic nonce in the three environments. For this, I compute the average BER over 5 independent trials at different distances between the VA input device (microphone) and user. Figure 4.9 shows that in all three environments, the average BER upto a distance of 1.5 m is 0%. Note, however that BER increases significantly beyond 1.5m in dining hall because the environment is relatively noisy.

Speech Re-use Detection. In this experiment, I measure the efficacy of AEOLUS in detecting speech re-use (see Table 4.2). The input speech is considered re-used if (i) the current nonce is not decoded correctly, and (ii) if any trace of an obsolete nonce is detected. FRR indicates the proportion of falsely rejected samples because of incorrect nonce decoding or because the detected nonce is obsolete. FAR indicates how many re-used speech samples with obsolete nonces are falsely accepted. AEOLUS achieves 0.5% FRR at 0% FAR for speech re-use prevention upto a distance of 4 meters in the three environments.

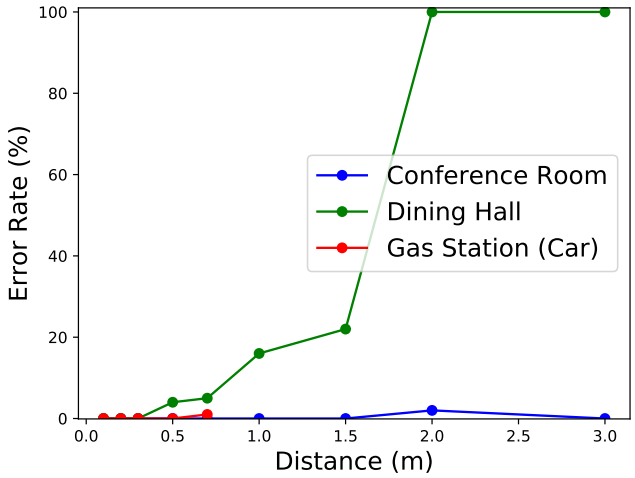


Figure 4.9: Variation of bit error rate (%) with different distances.

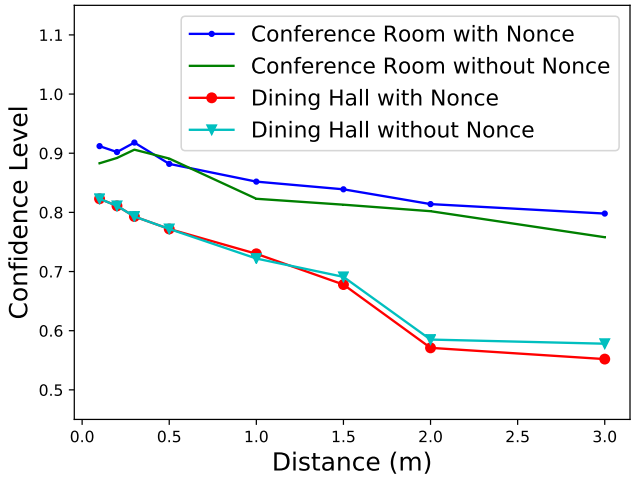


Figure 4.10: Impact of the AEOLUS on Microsoft Azure Speaker Recognition system.

4.4.2 Robustness

Impact on Speaker and Speech Recognition. The goal of this experiment is to assess the impact of acoustic nonce embedding and decoding on the performance of a commercial speaker and speech recognition system (Microsoft Azure). The results show that there is limited impact on the performance of the speaker recognition system for the conference room and dining hall environments upto a distance of 1.5 m (see Figure 4.10). The embedded nonce marginally increases the average word error rate (WER) of the speech recognition system to 4.4%. The WER is close to the

Table 4.3: Performance of Microsoft Azure speaker recognition system.

Removal Technique	4-5kHz, 2 Hop., Avg. of 30 samples		1-8kHz, 10 Hop., Avg. of 30 samples	
	Speaker Verification (Score, 0-1)	Speech Recognition (WER %)	Speaker Verification (Score, 0-1)	Speech Recognition (WER %)
Resampling	0.233	29.73%	0.170	91.83%
Amplitude Compression	0.217	34.31%	0.184	63.4%
Filtering	0.623	17.12%	0.319	62.21%
Additive Noise	0.318	53.13%	0.192	96.34%
Lossy Compression	Reject	Reject	Reject	Reject
Equalization	Reject	Reject	Reject	Reject

reported average WER of 5.1% for the same system in the NIST 2000 Switchboard test set [83]. Manually reviewing the results indicates that most errors are related to incorrect suffix "-ing". A few errors are due to small word omissions such as "an" and "a". Relatively more errors occur in the dining hall environment due to the background noise.

Acoustic Nonce Removal. An adversary may attempt to remove the embedded acoustic nonce and obtain a "clean" speech sample to conduct re-use attacks. To test the robustness of AEOLUS in this adversarial setting, I use the following 6 common audio watermark removal techniques [45]:

- Resampling. Samples the audio at a different frequency (e.g. 44.1 KHz) to remove sampling-dependent watermark.
- Amplitude Compression. Alters the amplitude of recorded audio to bypass amplitude-related watermark detection.
- Filtering. Uses a high pass or low pass filter to remove a specific range of frequency in the audio. For example, separate audible signals from inaudible frequencies (e.g. ultrasound).
- Lossy Compression. Leverages data encoding methods that uses inexact approximations and partial data discarding to compress the audio. For example, MP3 [84] or MPEG-1 Audio Layer III reduce audio file size by taking advantage of a perceptual limitation of human hearing.
- Equalization. Modifies the frequency response of an audio system using linear filters. Frequency response is the measure of the audio output in comparison to the input as a function of frequency. For example, a 5kHz wave input may have a 4kHz output response.

- Additive Noise. Adds noise to decrease SNR (signal-to-noise ratio) so that watermark cannot be extracted successfully.

Each technique is iteratively used until the acoustic nonce cannot be detected in a speech sample. The removal attempts are repeated for all speech samples in the dataset mentioned above. Table 4.3 shows that nonce removal techniques are disruptive and ineffective in removing the acoustic nonce. This is because they reduced the average recognition performance of the commercial speaker recognition system by 79.52%, and increased the average word error rate of the speech recognition system by 33.57%. This shows that AEOLUS is robust against these removal techniques.

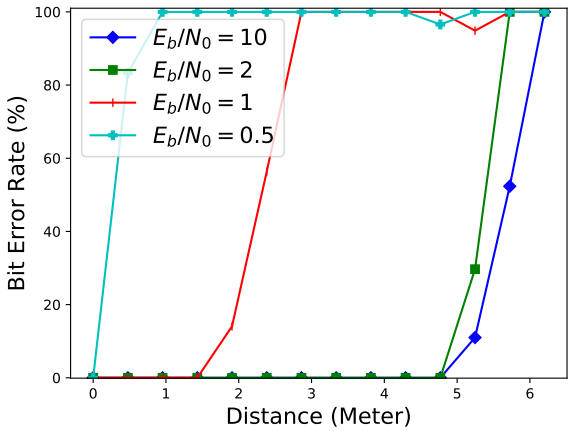


Figure 4.11: Impact of distance between the VA loudspeaker and an adversary.

4.4.3 Distance between Adversary and VA Speaker.

To launch a speech re-use attack, an adversary first records a user's interaction with VA. In this experiment, I measure how effectively the acoustic nonce is preserved with varying distance between the VA speaker and adversary's recording device (microphone) at the time of capture. Figure 4.11 shows that this distance significantly impacts BER. The computed BER is 0% upto a distance of 4 m when the signal to noise ratio is low. If the captured speech is being re-used,

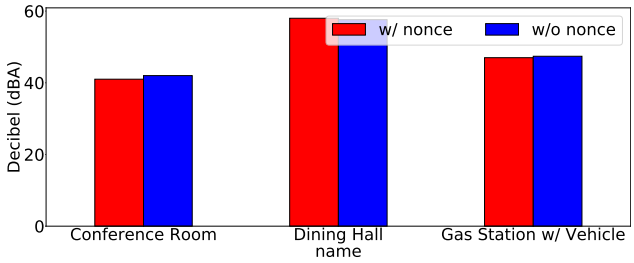


Figure 4.12: Loudness of speech with and without embedded nonce.

assuming a lossless audio input to VA, the entire nonce can be recovered and used for determining the particular user interaction session the speech was previously used in.

4.4.4 Human Perception

4.4.4.1 Empirical Measurement.

In this experiment, I empirically measure the change in SPL due to the embedded nonce. The distance between VA output and the recording device is 1 m in dining hall and conference room, and 0.7 m in the car. The results (see Figure 4.12) show that the average SPL difference in each environment is less than 1.4%. Non-parametric Wilcoxon Signed Rank test [85] (at 95% confidence level) is used to measure the statistical significance of the difference in SPL before and after acoustic nonce embedding. The results indicate that the differences are statistically insignificant.

Table 4.4: Perception of speech samples. .

Location	Imperceptible (Overall/Sample%)	Non-disruptive (Overall/Sample%)	Disruptive, Nonce-caused (Overall/Sample%)
Conference Room	34.87%/50%	50%/50%	1.31%/0%
Dining Hall	32.64%/25%	41.67%/75%	7.64%/0%
Gas Station (Car)	32.14%/25%	42.86%/75%	8.57%/0%

User Study. Next, I study the impact of AEOLUS on VA usability. For this, 120 participants (with prior experience using a VA) from various demographics are recruited on Amazon Mechanical Turk. The demographics for the participants are shown in Table 4.6. There are more female

Table 4.5: User study survey questions.

Question	Options
Q1. Pretend you are using voice-based service in a dining area, can you hear the speech in this audio clearly?	A. Very Clear B. Clear. C. It is neither clear or unclear but I am fine with it. D. I can hear the speech but it is very unclear. E. I cannot hear any speech.
Q2. Please transcribe the speech.	[Text]: ____
Q3. How noisy is this audio, rate from 0 - 10? (10 is most noisy)	[0-10]: ____
Q4. Do you find any noise affects your user experience?	A. I can hear something, but nothing bothers me. B. Yes, there is something troubles me. C. Nothing at all
Q5. If there are some noises affect your user experience for the voice service, what is it?	[Text]: ____

Table 4.6: User study participant demographic information.

Category	Gender Groups			Age Groups			
	Male	Female	Others	18-25	25-35	35-45	45-55
Number	65	53	2	32	56	21	11

participants than male participants, and 46.7% of the participants are between age 25 to 35.

The survey consists of three parts: device adjustment, preliminary questions, and survey questions. At the beginning of each survey, I guided the participants to adjust the loudness of their playback devices to a comfortable level. This is because the participants are recruited online, it is difficult to explicitly know or enforce the setup the participants use. Also, I pre-processed the recorded audio samples to ensure all audio samples were played with reasonable and consistent loudness.

Each participant is then asked to enter their demographic information. The participant is then instructed to play a test sample and adjust the playback volume to a comfortable level. Following this, the participant is presented with a survey with five speech samples and follow up questions.

Table 4.7: Average perceived noise in speech samples with and without acoustic nonce.

Location	w/o Nonce	w/ Nonce	t-test, $\alpha = 0.05$	Statistically Significant
Conference Room	3.8	4.8	$t = 4.2022, p = 0.0001489$	Yes
Dining Hall	5.7	6.1	$t = 1.4595, p = 0.1524$	No
Gas Station (Car)	5	5.6	$t = 3.0697, p = 0.00389$	Yes

Included in the five samples is one sample with only background noise (and no speech) to check if the participant is attentive. The participants are asked to provide speech transcriptions for all samples and the provided transcriptions are used to check the participants' attentiveness. Eleven participants failed to pass this test and hence were excluded.

Next, each participant is presented with a survey that takes approximately 10 minutes to complete. Post completion of the survey, a participant is paid 4 USD as incentive. The survey questions are designed to ensure bias is not introduced in the results (no leading questions). I showcase the survey questions for each audio in Table 4.5. When designing the questions, I do not ask participants directly about the imperceptibility. Instead, I ask them whether they can transcribe the speech command in the audio. Second, I ask how noisy they find the audio (from 0-10, and 10 is the noisiest score). Third, I ask if any available noise affects the user experience. Fourth, if the noise affects their user experience, what does the noise sound like. This question is designed to verify whether it is the acoustic nonce that bothers them.

The test samples are uniformly selected at random from a dataset of 24 speech recordings, 12 recordings each with and without embedded nonce from the three environments. The average length of the speech samples is 6.4 seconds. To ensure the user study is unbiased, all samples presented to a participant are from the same speaker and pertain to the same voice command. After listening to a speech sample, a participant is asked to rate the noise level of the speech sample on a scale of 0 - 10 (10 being the highest noise level). The participant is also instructed to report any noticeable noise source that affects their experience, and to specify the characteristic of the noise that resulted in this disruption (e.g., human conversation, train or car noise). The answer is used to ascertain the source of noise that impacted the usability (e.g., the embedded nonce).

Results. Table 4.4 reports the average human perception ratings. The acoustic nonce does not degrade overall user experience for 94.16% of speech samples, on average, in the three environments. Since this does not adequately measure the participant's perception of nonce embedding for each sample, per speech sample perception is also reported. No speech sample with embedded nonce is perceived to be disruptive by the majority of participants.

I show some example comments from participants who reported that the noise in the audio affects their user experience,

> *Compared to the other samples, the background voices are much louder. In fact, a man speaking almost overpowered the voice I is supposed to listen to.* (P-004, G2[2])

I find some participants reported other noise sources but the acoustic nonce. For example, P-004 says the people chatting in the dining hall affected his user experience of VA.

> *Car noises, but the singing is also very far away and hard to hear.* (P-072, G3)

Similarly, P-072 is complaining about street noise in Q5.

> *An echo, or like it is far away.* (P-019, G1)

Only 2 audios were reported to contain acoustic nonce (in Q5). For example, P-019 reported the echo sound and claimed it is affecting her user experience of VA. However, I notice that the acoustic nonce didn't affect their transcriptions' correctness (i.e., Q2 in Table 4.5).

Perceived Noise Level. Table 4.7 shows that the speech samples containing the acoustic nonce are perceived to have a higher noise level. The dependent samples paired t-test (at 95% confidence) indicates a statistically significant difference (between samples with and without nonce) in the conference room and car. However, this difference is statistically insignificant in the dining hall due to higher background noise level compared to the conference room and car. Despite perceiving higher noise level, the majority of participants do not perceive any speech sample to be disruptive (see Table 4.4).

[2]G1, G2, G3 are respectively the participant groups for testing audios collected in the conference room, dining hall, and gas station.

4.5 Discussion

In this section, I discuss AEOLUS's susceptibility to Denail of Service (DoS) attacks, its potential use in a layered VA security solution and as a security indicator, and future work.

Denial of Service (DoS) Attacks. An adversary can launch a type of DoS attack called *signal jamming* on AEOLUS-enabled VA by injecting acoustic waves with characteristic frequencies and amplitudes similar to AEOLUS's. However, this type of attack can be countered by frequently changing AEOLUS's operating frequency range. This would make it difficult for an attacker to predict the operating frequency at a given time and consequently launch effective real-time signal jamming attack. A smarter adversary can potentially inject acoustic waves in the entire spectrum of audible frequencies and amplitudes. Unlike common DoS attacks (e.g., link-flooding attacks [86, 87] in the network security domain) where it is usually challenging to pinpoint the attack, such an attack can be easily detected and mitigated. This is because to conduct the attack, the adversary would use the same frequency range as AEOLUS's (i.e., human audible frequencies), and these frequencies can be captured and analyzed without requiring any extra hardware.

Multiple AEOLUS-enabled VA devices used around the same time in each other's neighborhood may accidentally result in a DoS scenario. A potential way to prevent this could be through a centralized coordination mechanism between different VA devices to ensure they use different frequency ranges and nonce at the same time. Another reactive prevention method could be to allow AEOLUS to change it's operating frequency range if a DoS situation is detected.

Layered VA Security. AEOLUS is designed to prevent speech re-use as a security overlay. When used in conjunction with other security mechanisms, AEOLUS can greatly enhance VA usability as a passive and near-imperceptible defense mechanism. Consider, for example, active liveness detection methods such as audio CAPTCHA [88] or challenge-response methods which require users to respond with dynamic words or passphrases. Although such methods provide additional security, they add friction from a usability standpoint, especially for short user interaction sessions. A potential risk-based layered approach could proactively use the proposed framework in every user interaction, and invoke challenge-response, to mitigate transaction risk as necessary.

Use as security indicator. The proposed framework is designed to embed near-imperceptible acoustic nonce in the human audible frequency range. However, it can be tailored for use as a slightly perceptible security indicator similar to electric vehicle warning sound [89] or the light ring [90, 91] used in Amazon Echo devices. Presence of a gentle perceptible noise at the time of user interaction could indicate to the user that the interaction session with VA (e.g., Amazon Alexa) is secured using AEOLUS.

Future Work. In future, I plan to conduct in-depth investigation of methods for generation and encoding of acoustic nonce, e.g., using variations of music or other forms of sound along with different encoding schemes. I also plan to conduct large-scale live subject testing to further evaluate reliability and imperceptibility.

4.6 Summary

In this chapter, I present a security overlay called AEOLUS that can be integrated with any VA to prevent speech re-use. AEOLUS reliably embeds a dynamic acoustic nonce that is non-disruptive to a VA user, and detects the embedded nonce from the recorded user speech. Experiments conducted in three different environments show that AEOLUS can be used in operational VA scenarios with minimal overhead.

5. ANALYZING AND ATTACKING VOICE ASSISTANT SPEECH PROCESSING

5.1 Introduction

In the speech processing (shown in Figure 5.1) of the skill pipeline mentioned in Chapter 1, speaker recognition systems (SRSs) are used to automatically identify a person based on the recorded utterances' audio characteristics. They are widely used in different real-world scenarios, ranging from personalized services, speaker authentication to diarization [92] in speech transcription tasks. SRSs are usually implemented based on various machine learning techniques. However, they are known to be vulnerable to different adversarial attacks [93, 94, 10]. While many of them demonstrated successful attacks on SRSs, their practicality has not been scrutinized.

Various adversarial attacks have been proposed and studied for machine learning models used in speech-related tasks such as speech and speaker recognition. First, existing work shows that both white-box [95, 30, 96] and black-box adversarial attacks [97, 98] feasible in attacking speech recognition systems. Next, different adversarial attacks on SRSs are studied [99, 77]. The goal of such attacks is to craft a sample from a voice uttered by some source speaker. Therefore, it is misclassified as one of the enrolled speakers (untargeted attack) or a target speaker (targeted attack) by the system under attack. It is still correctly recognized as the source speaker by ordinary users. While white-box attacks [99] on SRSs are shown to be feasible, they are recognized to be less practical than a black-box approach due to the SRSs' closed-loop design. Subsequently, a new

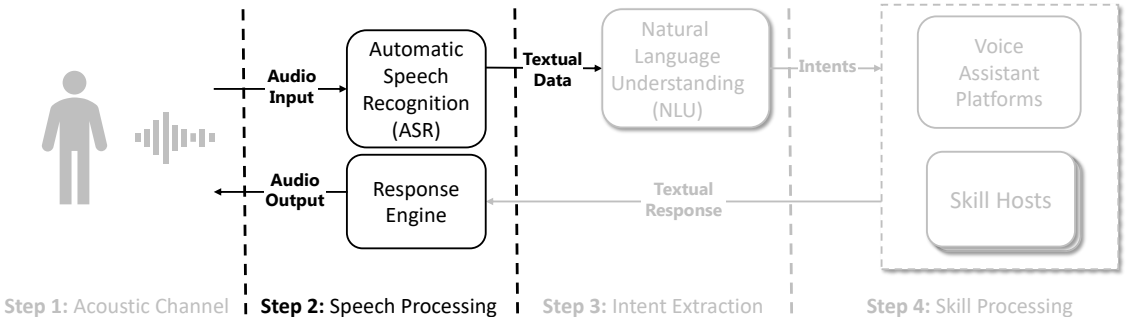


Figure 5.1: Skill Pipeline - Step 2

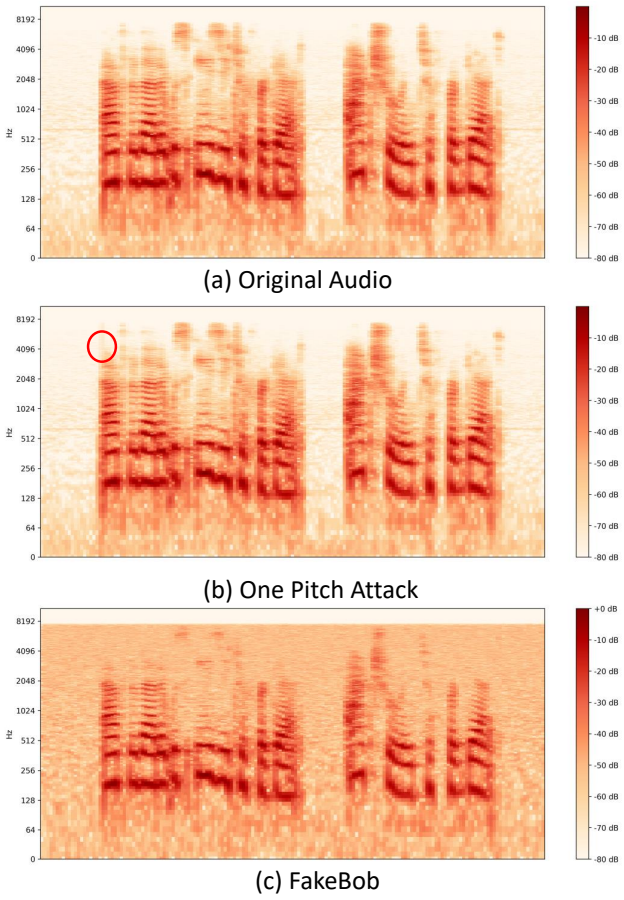


Figure 5.2: Spectrogram of one pitch attack compared with FakeBob attack.

attack named FakeBob [77] is proposed and demonstrated attack SRSs with a black-box setting.

However, FakeBob still falls short of practicality in terms of two important criteria. First, for each attack attempt on a piece of speech data, FakeBob takes many model queries to launch the attack. For example, for ivector models, FakeBob claims to take more than 40,000 queries and 38 minutes execution time to launch a successful black-box attack on an SRS. Second, FakeBob applies a global perturbation strategy (i.e., inject noise into the whole length of a targeted audio file), and the injected noise incurs obvious noise addition. For example, the user study in FakeBob work shows that, when comparing non-adversarial scenarios and adversarial attacks, 64.6% more users recognized the noise addition in the adversarial attacks. As shown in Figure 5.2, the perturbed space with FakeBob's approach is widespread over the whole audio file.

In this phase, I aim to improve the practicality of launching adversarial attacks on SRSs. I propose a new attack named one pitch attack, which leverages a multi-head attention estimation mechanism to improve the efficiency and imperceptibility of black-box adversarial attacks on SRSs. This attack consists of two parts. First, by leveraging the state-of-art attention mechanisms used in speech and speaker recognition, I estimate how a model focuses on certain frames or features. As a result, an attention map is built for further adversarial example generation. Moreover, to further generalize this approach, an attention map tuning is applied to assist the process of locating the attention of the models. In specific, a differential evolution (DE) [78] is used to focus on the most model-attended frames of speech data. Second, I design a stegano-filter which can be used to apply various audio steganography technique in the adversarial example generation procedure. Various steganography techniques such as spread spectrum and echo hiding are used to ensure the produced audio adversarial examples are as imperceptible as possible.

I evaluate one pitch attack with the VoxCeleb dataset [100, 101] and four different SRSs systems from both open-source or commercial tools. Among them, three of them are implemented based on Transformer-based (with attention mechanism), ivector-PLDA, GMM-UBM, respectively. The commercial tool (i.e., Microsoft Azure SRS) has no implementation details exposed. The result shows that one pitch attack achieves a 99% success rate on attention-based SRS and up to 99% success rate for other systems (in overlay mode mentioned in Section 5.3). By evaluating with other criteria such as query count, audio similarity, and signal to noise (SNR) ratio, I show that one pitch attack outperforms existing approaches in terms of both efficiency and imperceptibility.

5.2 Overview

5.2.1 Threat Model

In this phase, I study black-box adversarial attacks on SRSs. Hence, I consider the following assumptions.

- **Adversary Capability.** First, the attackers cannot poison SRS models or any of the SRS implementations. Second, the internal details of targeted SRSs are unclear to the attackers.

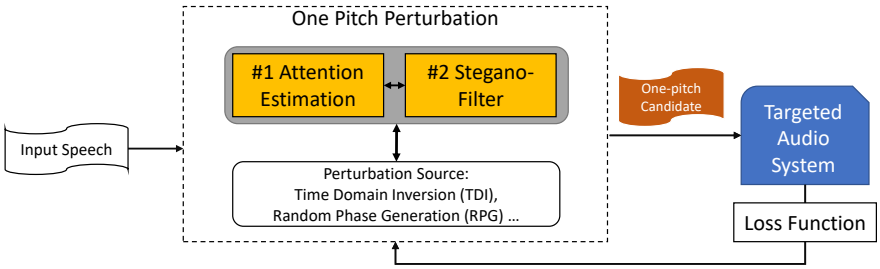


Figure 5.3: Illustration of one pitch attack overview.

Third, the attackers can conduct unlimited queries. However, to launch realistic attacks, it is ideal that the number of model queries and the execution time is minimal.

- **Attack Goals.** The targeted SRSs include both speaker verification and speaker identification tasks. A speaker verification system only has one user registered, and the system will either accept or reject one's identity claim. Speaker identification is designed to identify an unknown speech's identity. In this case, multiple users could be registered. The adversarial attacks have two goals: targeted attacks and untargeted attacks. First, when attacking a speaker verification system, an untargeted attack aims to disable an SRS so that a legit speech input will be misclassified. A targeted attack is to spoof the registered user's identity with any other speech input. Second, for speaker identification systems, an untargeted attack will result in misclassification of input speech to be any other registered users. A targeted attack for speaker identification ensures that an input speech is always being misclassified to be a targeted identity. In this phase, I focus on targeted attacks. Also, I consider a close-set setting for speaker identification tasks.

5.2.2 Problem Statement

I raise two research questions to start addressing the challenges of efficiency and imperceptibility of SRS adversarial attacks.

- How to find the "focused" area of an SRS model to conduct efficient audio perturbations? My idea is to find efficient local perturbation. I learn the lesson from one pixel attack [102]

that data-dependent models could have bias and in favor of certain parts of data or features. With the novel attention estimation mechanism, one pitch attack aims to guide the audio adversarial perturbation based on the attention maps (e.g., Figure 5.4) generated by the attention estimation module.

- How to further ensure the imperceptibility? Traditional adversarial perturbation methods try to minimize the strength of the noise. However, it is also important to ensure that the generated audio adversarial examples should be as stealthy as possible. I apply different audio steganography algorithms in the stegano-filter module to hide the audio perturbation in one pitch attack.

5.2.3 Design Overview

To address the two research questions, two main components: Attention Estimation and Stegano-filter, are designed in one pitch attack.

Attention Estimation. I estimate how a model focuses on certain frames by processing the input speech with multi-head attention estimation. As shown in Figure 5.3, the input speech will first be processed into mel-spectrogram form. While the baseline audio perturbation methods use traditional Time Domain Inversion (TDI) and Random Phase Generation (RPG). Second, the adversarial sample generation process is correlated with both attention estimation and Stegano-filter to decide where and how strong the perturbation should be. An attention map is built based on an 8-head attention estimation. The attention map used in this process contains a set of weights (for different frames) which indicate how vulnerable any frames are. The intuition is that if certain input areas are more focused by a model, the output of this model is more likely to be manipulated by changing these focused areas. However, the initial attention estimation can be inaccurate. This is because different SRS models can have different characteristics and tend to focus on different areas. To mitigate this problem, as a third step, attention map tuning is applied to assist the process of locating the attention of the models. In specific, a Differential Evolution (DE) strategy is used to help adjust the attention map with model queries. Fourth, the generated adversarial perturbation is

embedded with the original speech and feed to the targeted SRS. Similar to FakeBob, I leverage the l-infinite norm loss function to maximize the perturbation result while minimizing the perturbation area and strength. Moreover, Natural Evolution Strategy (NES-BIM) optimization solver [77] is used. In Section 5.3, I focus on illustrating detailed attention estimation process.

Stegano-filter. I design a stegano-filter to ensure that the generated adversarial perturbation is as imperceptible as possible. Specifically, I learn the lessons from both auditory masking and spread spectrum to decide what are pitches allowed in one pitch attack. First, I compute the log-magnitude power spectral density (PSD) of each frame to set up the intensity limit of the generated pitch. Second, I apply the philosophy of spread spectrum in audio steganography and set up the limit of pitch frequencies. As it is difficult to solve the optimization problem with multiple constraints (i.e., adversarial generation and perturbation minimization) at the same time [96], I apply a greedy approach and design the stegano-filter to filter out illegal candidate generated from the aforementioned attention-guided process. I introduce more details in Section 5.4.

Overlay Mode. In the default setting, one pitch attack limits the audio perturbation only at the most focused frames (i.e., at most 20% of the frames) and limits the model query number (i.e., at most 1000 queries). However, this could incur undesirable success rates. Hence, an overlay mode of one pitch attack is also used to boost the other traditional adversarial generation methods. After removing the cap of perturbation frames and model query, one pitch attack works as a booster to help reduce the model queries and noise addition.

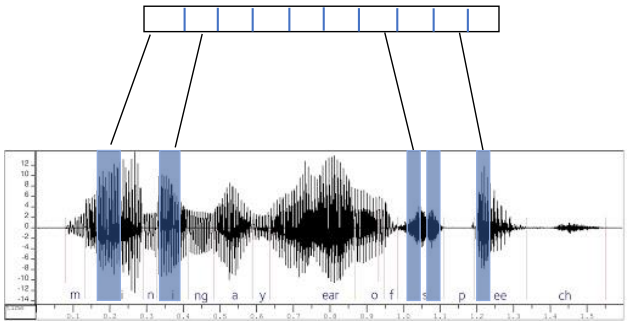


Figure 5.4: An example attention map.

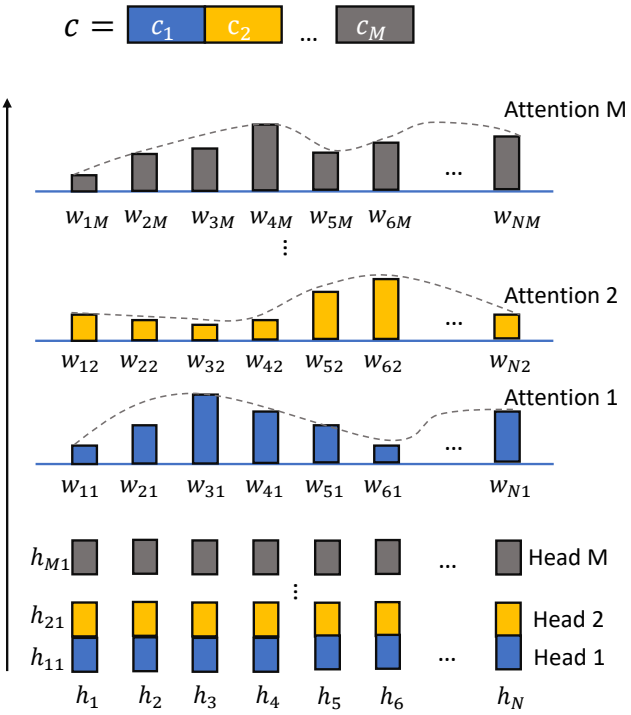


Figure 5.5: Illustration of the multi-head attention in utterance level representation.

5.3 Attention Estimation

In one pitch attack, the attention estimation module aims to find out the focused area of a targeted SRS without excessive model queries compared with existing work. A two-step design is provided in this section to illustrate how the Transformer-based multi-head attention mechanism helps find the "attention" of different SRS models. Specifically, the most critical frames that would affect the outcome most are decided in the form of an attention map. It is then used for guiding the adversarial perturbation.

Moreover, to ensure that this attention estimation is more scalable to different SRS models, I apply a differential evolution (DE) based optimization scheme to tune the attention maps further. The DE strategy with a model-query-based validation is applied in this optimization.

5.3.1 Leveraging Attention Mechanism

As mentioned above, the input of the attention estimation module is the spectrogram generated from input speech data. The output of the attention estimation is the corresponding attention map

Figure 5.6: Attention map generation with tuning.

for this speech. To formulate the attention estimation problem in one pitch attack, as shown in Figure 5.5, I show a sequence of hidden states $h = \{h_1, h_2, ..., h_N\}$, and training parameters u ($h_t, u \in \mathbb{R}^d$). In Equation 5.1, I define a softmax-based attention mapping which indicates a relevance scalar weight for each element of the sequence:

$$w_t = \frac{exp({h_t}^T u)}{\sum_{l=1}^{N} exp({h_l}^T u)} \tag{5.1}$$

w_t is then used to define the attention map in an utterance level representation c. As shown in Equation 5.2, the weights are aggregated to define the focus of this sequence:

$$c = \sum_{t=1}^{N} h_t^T w_t \tag{5.2}$$

If k heads are considered, $h_t = \{h_{t1}, h_{h2}, ..., h_t\}$. Note that $h_t j \in\in \mathbb{R}^{d/k}$ which means the head size is d/k. As shown in Equation 5.3, training parameters $u = \{u_1, u_2, ..., u_k\}$ (similarly, $u_j \in\in \mathbb{R}^{d/k}$) is used to define w_{tj} which indicates the attention map of the head j on the t sequence:

$$w_{tj} = \frac{exp({h_{tj}}^T u_j)}{\sum_{l=1}^{N} exp({h_l j}^T u_j)} \tag{5.3}$$

Similar to Equation 5.2, a multi-head attention map with utterance level representation is defined in Equation 5.4:

$$c_j = \sum_{t=1}^{N} h_t^T w_t \tag{5.4}$$

where $c_j \in \mathbb{R}^{d/k}$ which indicates the utterance level representation from head j. As shown in Figure 5.5, the attention map is acquired by a concatenation of utterance level representation $c =$

$\{c_1, c_2, ..., c_j\}$. The usage of multiple heads makes sure that the performance of attention estimated is not affected by the length of input speech. This attention map is then used to guide a ranked perturbation approach. The frames used for perturbation will be increased based on the weights. In one pitch attack design, I start with the top 1% frames and gradually increase the available frames. As mentioned above, in the default mode, I limit the maximum perturbed frame size to be 20% of the total frames.

5.3.2 Attention Map Tuning

As different SRS models can have different attentions, I further apply an attention estimation tuning using DE optimization (shown in Figure 5.6). The goal of this step is to adjust the attention map to different models in a black-box setting. DE's candidate evolutionary feature can help find the optimal attention map for SRS models. One of the reasons for using the greedy DE approach is to minimize the required model query needed for this step. Another reason is that attention estimation optimization is considered to be non-differentiable and non-continuous. This is because the distribution of weights is not predictable. The nature of a black-box setting makes traditional gradient descent unable to find a reasonable optimal area.

Hence, I apply (DE) [78, 79] to obtain optimal parameters because DE is known to be an efficient optimization method [103, 77] for practical problems which have objective functions that are non-differentiable, non-continuous, noisy, or have many local minima, constraints. DE is more suitable compared to other gradient-dependent methods because it can be used regardless of the underlying model or other settings.

5.4 Stegano-filter

The stegano-filter is designed to filter out adversarial examples based on different audio steganography algorithms. In this section, I introduce about what are the steganography algorithms that are applied.

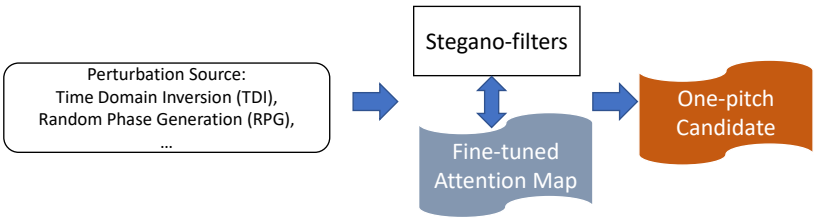


Figure 5.7: Stegano-filter workflow.

5.4.1 Stegano-filter Design.

Different from the original goal of audio steganography, which is to send hidden information secretly and reliably, the goal of stegano-filter is only to ensure that any embedded signal is as imperceptible as possible. As mentioned in Section 5.4, applying auditory masking in stegano-filter helps define what the valid audio perturbations are. The spread spectrum method helps define where these perturbations can be located if a frame is decoded to be focused in an attention map. As shown in Figure 5.7, the input of the stegano-filter is the generated audio signals from perturbation sources such as TDI and RPG. The stegano-filter works with the fine-tuned attention map to decide what perturbation result can be output as the one-pitch candidates for further testing.

5.4.2 Auditory Masking.

When applying auditory masking, there are two potential ways of setting up a threshold for filtering out invalid candidates: intensity-based or amplitude approaches. The latter is more complicated because the speech signal is usually not in a fixed form. As a result, I apply the intensity-based approach. In this way, I calculate the log-magnitude power spectral density. (PSD) to provide an estimation of the current frame (i.e., the host signal). Then, the allowed perturbation's PSD should strictly below the PSD of the host signal. Another threshold in my design is the temporal offset. This indicates how far a perturbation signal can be apart from the host signal. I set up an allowed offset to be two-frame length. Also, the allowed PSD is decayed when it has a longer offset.

I also learn the lesson from the spread spectrum and guide the perturbation to work certain

frequencies. First, I estimate the frequency using pitch estimation based on CREPE [104]. Then I disable any perturbation higher than the frequency that is estimated by the pitch estimation.

5.5 Evaluation

First, I show the implementation details of both SRSs and the proposed one pitch attack scheme. Second, to evaluate one pitch attack, I answer three questions:

- Can one pitch attack work? I evaluate the proposed scheme based on four different metrics: success rate, signal-to-noise ratio, model query number, and audio file similarity (between the original file and perturbed file)
- Can one pitch attack outperform existing approaches? I compare one pitch attack with existing work to showcase its efficiency and imperceptibility.
- How would one pitch attack strategy improve other adversarial audio example generation? Lastly, I demonstrate that the proposed perturbation can work with different audio adversarial example generation algorithms.

5.5.1 Implementation

I present the implementation details of one pitch attack and the testing SRSs.

One pitch attack. First, the baseline attention map generation is implemented based on s3prl's speech pre-trained models [105]. Second, other functions such as attention map tuning and stegano-filter are implemented with an in-house python program.

SRS. I tested the proposed adversarial perturbation with three different SRS implementations. First, a speech recognition based on ivector-PLDA [106] is used. It is the mainstream method for implementing SRSs in both academia [107] and industries [77] which provides the state-of-the-art performance for all the speaker recognition tasks. The second targeted system is Gaussian mixture model (GMM) based methods. It is known that GMM-UBM tends to provide better accuracy for shorter-length utterances in speech and speaker recognition systems. Third, I also leverage attention-based SRS implementations based on s3prl [108, 105] downstream tools.

Table 5.1: Performance of different SRSs.

Tasks	Metrics	ivector	GMM	Attention-based
Speaker Identification (close-set)	Accuracy	99.6%	99.2%	99.6%
Speaker Verification	FAR	1.1%	5.0%	4.1%
	FRR	11.0%	10.4%	9.6%

FAR: False Acceptance Rate; FRR: False Rejection Rate.

To train these systems and then test one pitch attack, I adopted two sets of datasets: VoxCeleb (both VoxCeleb1 [100] and VoxCleb2 [101]) and librispeech [109]. Specifically, for the comparison purpose, I use the same size of data as FakeBob's. There are 7,273 audio data for training the aforementioned SRSs. The testing dataset consists of 5 speakers from LibriSpeech: 3 female and 2 male. Also, there are 5 voices for each speaker and the audio length ranges from 3 to 4 seconds. For the imposter speaker dataset, there 4 speakers from LibriSpeech: 2 female and 2 male. Similarly, there are 5 voices for each speaker. The audio length ranges from 2 to 14 seconds.

5.5.2 One pitch attack performance

To test SRSs, I first evaluate their performance in terms of accuracy. As shown in Table 5.1, all three SRSs have high accuracy for speaker identification (close-set) tasks. A close-set speaker identification means that the classifier will always yield a result that is one of the enrolled speakers. These results show that the trained SRS classifiers are accurate and reliable for evaluating the proposed adversarial attack.

Attention Map Generation. Before the attack evaluation, I test whether attention map tuning is effective or not. To do that, I change the limit of tuning iterations mentioned in Algorithm 1. As shown in Figure 5.8, with more tuning iterations, the audio file similarity can be greatly reduced. Similarly, in Figure 5.9, I show that a better attention map tuning can reduce the SNR which means the injected adversarial example is more imperceptible. The reason for such reduction is that attention map tuning helps to locate more effective areas for adversarial perturbation. However, such a trend is not infinite. As shown in both Figure 5.8 and Figure 5.9, the reductions are negligible after

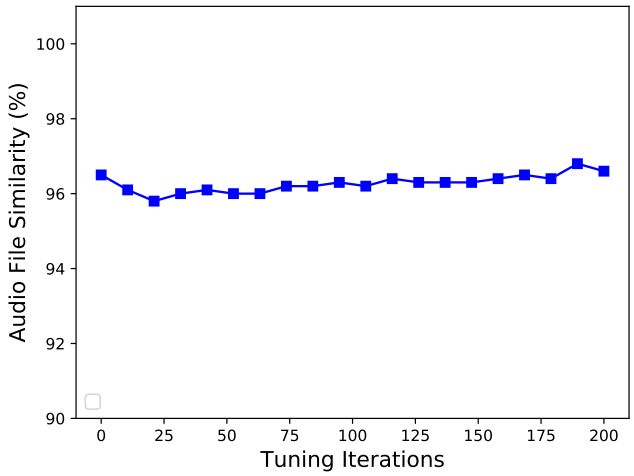


Figure 5.8: Audio file similarities for different attention map tuning.

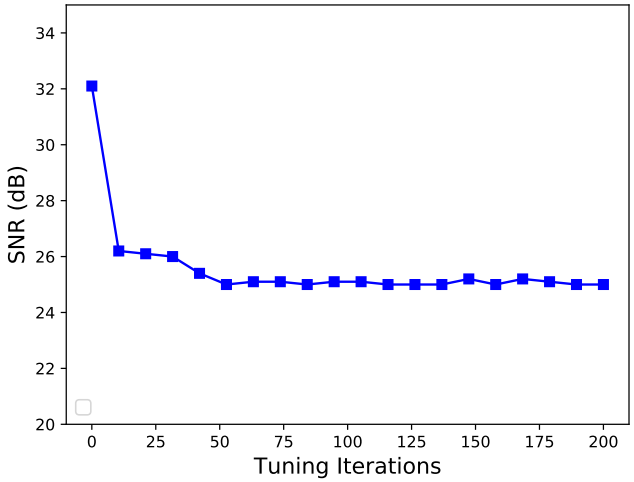


Figure 5.9: Signal-to-noise ratio results for different attention map tuning.

100 iterations.

In Figure 5.10, I show that attention map tuning can help reduce the model query cost for ivector and GMM SRS models. However, for attention-based the is no reduction observed. This is because the one pitch attack uses the same attention look-up scheme as the classifier. In other words, in a white-box approach, the tuning is not effective. A similar trend is also observed in Figure 5.11 that attention map tuning iteration setting does not affect the success rate of the one pitch attack attacks.

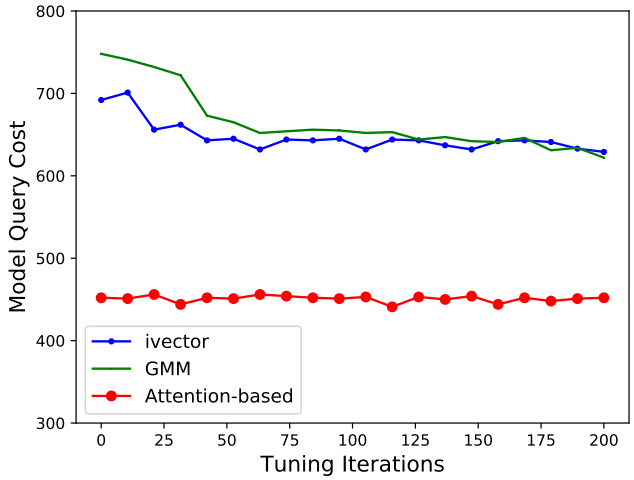


Figure 5.10: Model query results of three SRS models for different attention map tuning.

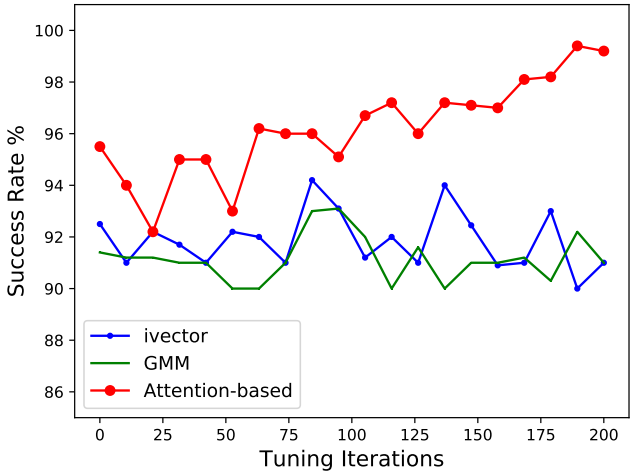


Figure 5.11: Success rates of three SRS models for different attention map tuning.

Attack Performance As shown in Table 5.2, one pitch attack can effectively (in hundred-level model queries) launch adversarial attacks with low SNR (i.e., below 30 dB). Also, the audio file similarity values are high which means the resulting attacking audio files sound similar to the original ones. I also find that the query costs are in the hundred levels which means the attack can be done in a few minutes if a standard computer is used.

Table 5.2: Performance of one pitch attack with targeted attack setting on different SRSs.

		Success Rate	Avg. Query Count	Signal to Noise Ratio (SNR)	Similarity
Speaker Verification	ivector	91%	672	26.1 dB	95.4%
	GMM	90%	850	27.3 dB	94.9%
	Attention-based	99%	470	28.1 dB	95.1%
	Azure	89%	750	29.6 dB	96.6%
Speaker Identification	ivector	91%	672	28.1 dB	95.4%
	GMM	89%	860	29.3 dB	95.2%
	Attention-based	99%	451	28.1 dB	96.5%
	Azure	90%	656	29.3 dB	95.1%

Table 5.3: Efficiency comparison between one pitch attack and FakeBob.

Attack Name	Model Query #	SNR
One pitch attack	672	27.2
FakeBob	2500	30.1

5.5.3 One pitch attack vs. FakeBob

I compare one pitch attack with the state-of-the-art FakeBob attack in terms of success rate with different SNR values. In Figure 5.12-5.15, I show that when reducing the SNR requirement, the attack success rates for FakeBob drop significantly. For one pitch attack, the success rates are more stable and can work with low SNR well. For FakeBob attack, it performs worse if the SNR restriction is tighter. This means FakeBob would work well with louder noise addition. This demonstrates the improved imperceptibility of the proposed one pitch attack. Moreover, in Table 5.3, I show that one pitch attack can outperform Fakebob in terms of efficiency with much lower model queries needed.

Table 5.4: One pitch attack with different audio adversarial generation algorithm.

Geneartion Algorithm	Model Query Decrease	SNR Decrease
FGSM	82%	15%
JSMA	31%	11%
DE	45%	12%

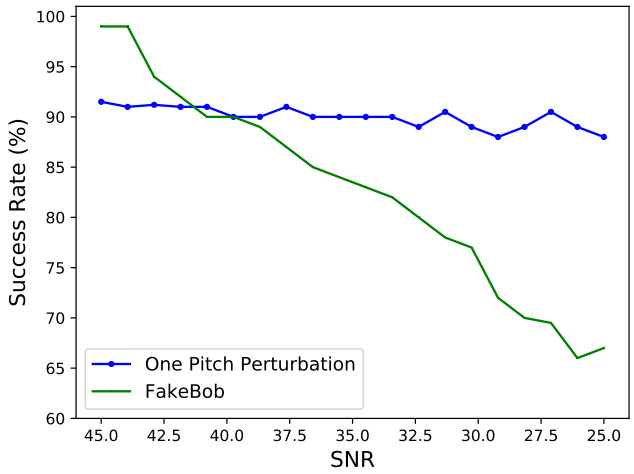


Figure 5.12: Success rate results for different SNR levels for speaker identification with ivector.

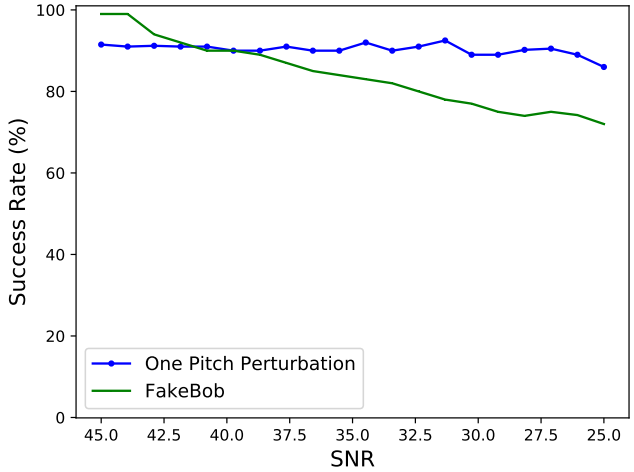


Figure 5.13: Success rate results for different SNR levels for speaker identification with GMM.

5.5.4 One pitch attack (overlay mode) with different adversarial attack algorithms

As mentioned in Section 6.2, one pitch attack can as an attack optimization scheme that works on top of other general adversarial perturbation algorithms. This means that one pitch attack is used to guide the perturbation and the perturbed areas are not restricted by be the attention map. I evaluate how one pitch attack can help existing perturbation schemes by implementing an overlay mode of the proposed attack. In Table 5.4, I show that the proposed scheme can greatly reduce the model query cost and improve the imperceptibility of these baseline attack algorithms. Note that,

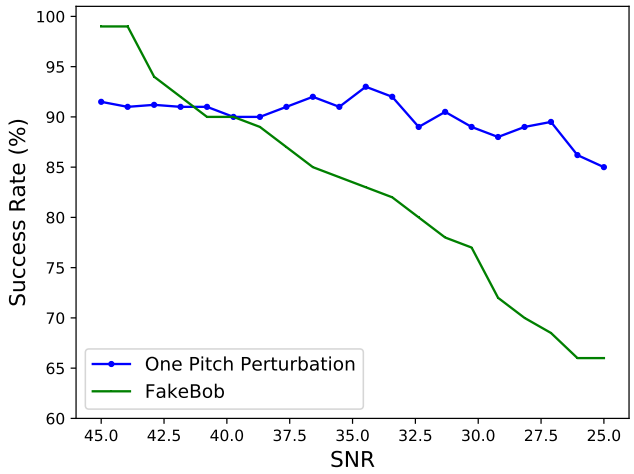


Figure 5.14: Success rate results for different SNR levels for speaker verification with ivector.

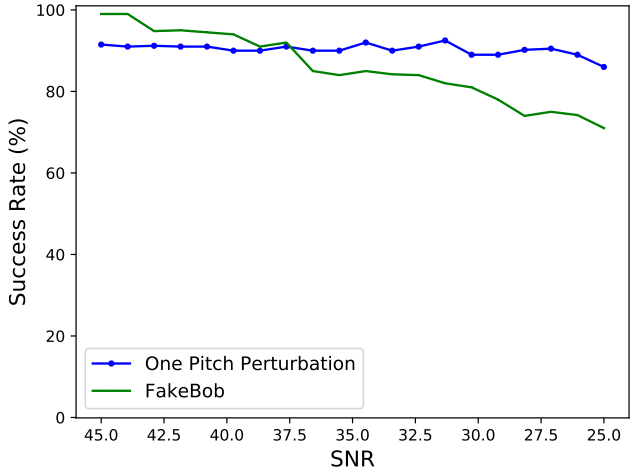


Figure 5.15: Success rate results for different SNR levels for speaker verification with GMM.

the attack success rates are reached 99%.

5.6 Discussion

One of the limitations of one pitch attack is that the success rate for a black-box setting is not always as high as 99%. The reason is that many models are not entirely relying on a part of the frames or features to make the final decision. However, I also show that if the one pitch attack works in an overlay mode, the attack success rate can be as high as other approaches. Moreover, the overlay-mode one pitch attack can greatly help to reduce the model query cost and improve the

attacks' imperceptibility.

5.7 Summary

In this chapter, the proposed one pitch attack solves the problem of inefficient and potentially noisy global perturbation which is commonly used in traditional adversarial attacks. Specifically, I adopt an attention-based strategy learned from state-of-art Transformer architecture. Also, with help of audio steganography, this scheme can effectively and imperceptibly attack SRSs with upto 99% success rate in a few hundred queries. Our proposed attack is also model-agnostic and can work as an overlay method over traditional adversarial attacking algorithms.

6. ANALYZING AND SECURING INTENT EXTRACTION OF VOICE ASSISTANT APPLICATION*

6.1 Introduction

In this chapter, I focus on the components beyond the speech processing step of the skill pipeline (shown in Figure 6.1). Several attacks have been reported to affect the integrity of existing Automatic Speech Recognition (ASR) component in skill processing. For example, acoustic-based attacks [10, 31, 32, 30, 28] leverage sounds that are unrecognizable or inaudible by the human. More recently, Kumar et al. [33] presented an empirical study of skill squatting attacks based on speech misinterpretation (for example, an Alexa Skill named "Test Your Luck" could be routed to a maliciously uploaded skill with a confusing name of "Test Your Lock"). This attack works, with proof-of-concept, in a remote manner that could potentially be more powerful than acoustic-based attacks.

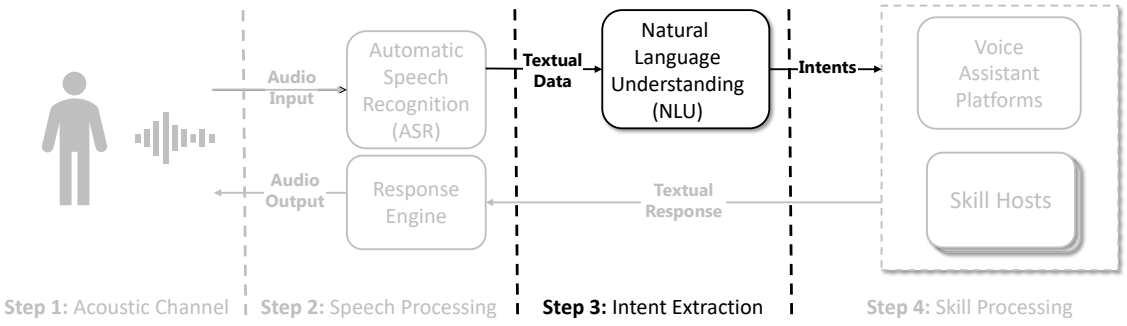


Figure 6.1: Skill Pipeline - Step 3

Despite recent evidence [33] of launching potential skill squatting attacks, little effort has been made to uncover the root cause of speech misinterpretation in skill processing. As mentioned in

* Reprinted with permission from "Life after Speech Recognition: Fuzzing Semantic Misinterpretation for Voice Assistant Applications" by Yangyong Zhang, Lei Xu, Abner Mendoza, Guangliang Yang, Phakpoom Chinprutthiwong, Guofei Gu, 2019. The 2019 The Network and Distributed System Security Symposium (NDSS '19), Copyright 2019 by Zhang et al., publication rights licensed to IEEE.

Chapter 1, in the NLU of skill pipeline, an Intent Classifier uses voice command templates (or templates) to match intents (similar to Intent Message in Android) with textual data obtained after ASR's text interpretation. Then, intents are used to reach skills with specific functionality. From the skill security perspective, the Intent Classifier plays a more important role when interpreting users' voice commands. The reason is that the Intent Classifier is the last step of the interpretation process. It has the capability of not only determining users' semantic intents, but also fixing the ASR's potential transcription errors. Second, while ASR is a default built-in service component, the construction of the Intent Classifier's semantic classification is contributed by both skill developers and service providers. Particularly, third-party developers can upload voice command templates to modify the unified intent classification tree used by all users. As a result, it creates the opportunity for misbehaving developers to maliciously modify the intent matching process of the NLU.

However, it is challenging to systematically study how NLU's Intent Classifier is penetrated to incur semantic inconsistency between users' intents and VA's machine understanding. First, mainstream VA platforms, such as Amazon Alexa, Google Assistant, and almost all skills are in closed development. Thus, it is difficult to conduct a white box analysis. Also, due to the strong privacy enforcement, it is impossible to get real users' speech input and the corresponding skill response output. Thus, conducting large-scale, real-world data-driven analysis is also very challenging.

My Approach. In this phase, I assess speech misinterpretation through the black-box mutation fuzzing of voice command templates, which are used as inputs of Intent Classifier for matching intents. My goal is to systematically evaluate how Intent Classifier behaves when inputting different forms of voice commands. However, it is not straight-forward to design such a fuzzing scheme. First, the computability of skill I/O is very limited as both input and output of skills are in the speech form, i.e., you can only speak with a VA device or listen to the audio response[1]. Thus, I have to determine mutation fields that I can work on in the context of skill processing.

[1]With a VUI setting, no other user interfaces such graphic user interface (or GUI) are being used.

Moreover, it is important to eliminate the effect of ASR's text interpretation so that error propagation is minimized. Second, in [33], the authors suggest that skill squatting attacks are caused by pronunciation-based interpretation errors of ASR. However, in reality, ambiguous natural languages incur many more different forms of confusing voice commands. For example, a user could be simply using regional vocabulary (i.e. words or expressions that are used in a dialect area but not commonly recognizable in other areas) rather than pronunciation issue alone. Moreover, it is unpredictable when and how a user would speak differently. All of these factors make the voice command fuzzing difficult because of the large searching space.

To overcome the aforementioned design challenges, I propose a novel linguistic-model-guided fuzzing tool, called LipFuzzer. My tool generates potential voice commands that are likely to incur a semantic inconsistency such that a user reaches an unintended skill/functionality (i.e. users think they use voice commands correctly but yield unwanted results). For convenience, I name any voice commands that can lead to such inconsistency as LAPSUS.[2] LipFuzzer addresses the two aforementioned challenges in two components. First, to decide mutation fields, I realize that the state-of-the-art Natural Language Processing (NLP) techniques can be used to extract computational linguistic data. Thus, I design LipFuzzer's basic template fuzzing by first mutating NLP pre-processed voice command templates. Then, I input mutated voice commands to VA devices by using machine speech synthesis to eliminate the human factor of producing ASR-related misinterpretation. Second, to reduce the search space of the basic fuzzing, in the linguistic modeling component, I train adversarial linguistic models named LAPSUS Models by adopting Bayesian Networks (BNs). BNs are constructed statistically with linguistic knowledge related to LAPSUS. The template fuzzing is then guided with linguistic model query results. As a result, LipFuzzer is able to perform effective mutation-based fuzzing on seed template inputs (gathered from existing skill user guidance available online).

In the evaluation, I first showcase that Intent Classifier is indeed the root cause of the misinterpretation. Then, I show that LipFuzzer can systematically pinpoint semantic inconsistencies with

[2]I use LAPSUS for its Latin meaning "slip".

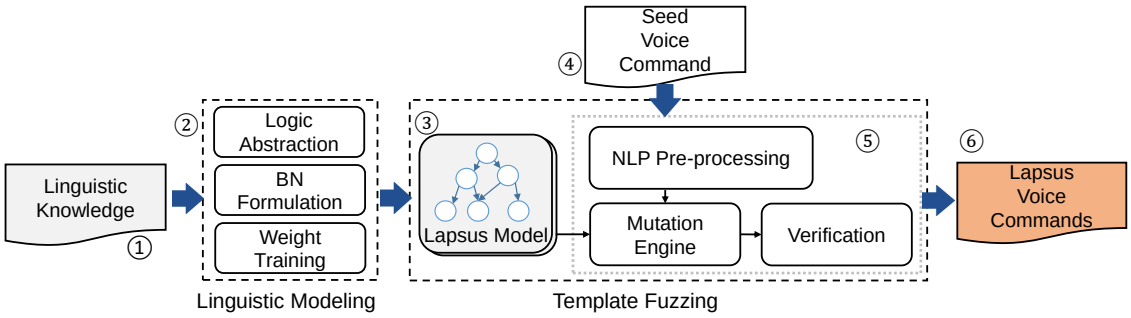


Figure 6.2: LipFuzzer architecture

the help of linguistic models generated from existing linguistic knowledge.

Furthermore, I scan Alexa Skill Store and Google Assistant Store to assess the security problem I found. My result shows that 26,790 (out of 32,892) Amazon Alexa Skills and 2,295 (out of 2,328) Google Assistant Actions are potentially vulnerable. A further sampling-based verification shows that around 71% of these Amazon Alexa skills and 29% of these Google Assistant skills are actually vulnerable. I provide detailed results and analysis in Section 6.5.3.

6.2 Fuzzing Speech Misinterpretation

In this section, I first present detailed challenges of fuzzing speech misinterpretation. Then, I briefly illustrate my LipFuzzer design which addresses the challenges accordingly.

6.2.1 Fuzzing Challenges

As mentioned in Section 6.1, I aim to design a mutation-based scheme to fuzz the problematic Intent Classifier. However, it is challenging because of VA's unique features:

(i) Mutation Field: Deciding the mutation field is not straight-forward since natural languages (e.g. voice commands) are complicated in terms of the linguistic structure. A voice command can be segmented into different levels of linguistic units: from low-level phonetic features to high-level phrases consisting of a few words. Therefore, I need to first decide what are the most LAPSUS-relevant linguistic units. Additionally, I need to select the proper tool to extract computational linguistic data for mutation.

(ii) Space Explosion: A naive fuzzing solution is to use random fuzzing to generate LAPSUS,

e.g., at the character level, I may mutate each character of a voice command. However, such random fuzzing is impractical as it would trigger a large number of resulting voice command variations. For example, for character-based mutation, a simple voice "open the truck" has 12 characters and each character of a word can have at least 25 possible ways of mutation (e.g. "a" to "b"). Moreover, a voice command can also be mutated with phonemes, words, etc., which makes the potential fuzzing space infinitely large. As a result, I need to find a strategy to reduce the search space of finding effective LAPSUS.

6.2.2 My solution: LipFuzzer

To solve the fuzzing challenges, I design LipFuzzer shown in Figure 6.2.

Linguistic Modeling. To overcome the difficulty of locating mutation fields, I convert vague voice commands (textual data) into computational fine-grained linguistic data using NLP. Specifically, the textual data are processed into a three-level linguistic structure that is commonly used in studying linguistic speech errors [110]: pronunciation, vocabulary, and grammar. Thus, I design the Linguistic Modeling component of LipFuzzer. As the input of linguistic modeling, I collect relevant linguistic knowledge from both academic materials and English teaching websites. Second, my tool abstracts logic representation (i.e., predicate logic) of these pieces of knowledge by manual parsing, or automatic logic generation (if examples are provided). Third, I construct linguistic models, namely LAPSUS Models, by formulating Bayesian Networks (BNs) from predicate logic entities I collected. Lastly, I train BNs with statistical weights which measure how likely it is that a state transition (i.e., mutating a field to result in another state) would take place. The LAPSUS Models can be used to answer both how a LAPSUS will be generated, based on seed voice commands (i.e., collected templates used for matching skill intents), as well as the weights of these mutations. Lastly, The LAPSUS Models are loaded with Template Fuzzing to guide the Mutation Engine for generating LAPSUS voice commands.

Template Fuzzing. To address the space explosion problem when fuzzing voice commands, I leverage LAPSUS-related linguistic knowledge to reduce the search space to find effective LAPSUS. Thus, I design my template fuzzing component which takes the input of seed voice com-

mands and generates LAPSUS voice commands. Three basic modules are shown in Figure 6.2: NLP Pre-processing, Mutation Engine, and Verification. First, the NLP Pre-processing module extracts linguistic information for mutating. Second, the Mutation Engine mutates the fields such as phonemes, words, or grammar entities based on LAPSUS Models. Third, the Verification module synthesizes speech voice command to verify whether a LAPSUS is effective.

6.3 Linguistic-Model-Guided Fuzzing

In this section, I detail LipFuzzer's design. First, I define the input and output of LipFuzzer. Second, I present the linguistic modeling component for producing LAPSUS Models. Lastly, I illustrate Template Fuzzing which is guided with LAPSUS Models.

6.3.1 Fuzzing Input & Output

Linguistic Knowledge. The input of linguistic modeling is linguistic knowledge data shown in ① of Figure 6.2. I choose LAPSUS-related linguistic knowledge from multiple sources [111, 112, 113, 114, 115]. They are part of many linguistic concepts (examples shown in Table 6.1) related to LAPSUS. With various descriptions and discrete features, I have to manually select them. As a result, I have a total of 498 pieces of knowledge collected (which are expected to expand over time), including 53, 264, 181 for sound-level, word-level, and grammar-level, respectively. In addition, 223 of them come with examples (e.g., target and LAPSUS in Table 6.1).

LAPSUS Models. The output of the linguistic modeling component is shown in ③. Multiple linguistic models are generated after the mathematical modeling. In this phase, I use graph-based data structures to represent linguistic knowledge statistically. Moreover, a LAPSUS can be queried with proper query inputs (more details in Section 6.3.3).

Seed Input. The input of Template Fuzzing is seed voice command templates (or seed templates) shown in ④. I collect seed templates from Alexa Skill Store through web crawling. These seed templates are example voice commands posted by skill developers. Although only a few developers would display all voice commands in the store, most of these example voice commands are essential ones. Thus, I believe these voice commands are enough to demonstrate the design

flaw in Intent Classifier.

Fuzzing Output. In ⑥, LipFuzzer generates modified voice commands based on seed inputs. A modification is done through a linguistic knowledge guided mutation.

Table 6.1: Example LAPSUS with logic abstraction

Lapsus	Description	Examples	Example Logic Abstraction
Blends†	Two intended items fuse together when being considered.	Target: person/people LAPSUS: perple	∀ x,y,phoneme(END,"S-N",x), phoneme(END,"P-L",y) → phoneme_exch("S-N","P-L",-)
Morpheme∗ -Exchange	Morphemes changes places.	Target: He packed two trunks. LAPSUS: He packs two trunked.	∀ x,y,suffix("ed",x), suffix("s",y) → suffix_exch("ed","s",-)
Regional Vocabulary ‡	Everyday words and expressions used in different dialect areas	Target: Entree Lapsus: Hotdish (esp. Minnesota)	∀ x,word("entree",x),→ word_exch("entree","hotdish",-)
Category Approximation ‡	Word substitution due to the lack of vocabulary knowledge.	Target: Show my yard camera. Lapsus: Turn on my yard camera.	∀ x,word("show",x), → word_exch("show","turn on",-)
Portmanteaux‡	Combined words that are used.	Target: Eat the (late) brekfast Lapsus: Eat the brunch	∀ x,word("late breakfast",x), → word_exch("late breakfast", "brunch",-)

†: Pronunciation, ‡: Vocabulary, ∗: Grammar.

6.3.2 Linguistic Modeling

I build LAPSUS Models based on existing LAPSUS knowledge. As shown in ② of Figure 6.2, LAPSUS Models are generated with three modules: Logic Abstraction, BN Formulation, and Weight Training.

Logic Abstraction. As the first step of linguistic modeling, I use predicate logic to represent collected LAPSUS knowledge. The reason is that predicate logic is widely used in traditional linguistic studies with extensible representation. It is capable of using quantified variables over non-logic LAPSUS, which then allows succeeding modeling and computation of LAPSUS knowledge. As my fuzzing scheme works with pronunciation, vocabulary, and grammar level of linguistic units, my predicate logic representations are also defined with these three types.

However, as linguistic knowledge is typically defined in a discrete manner, many are vague and difficult to translate. Thus, I transform collected linguistic knowledge into predicate logic repre-

sentation in two ways: manual abstraction and automated example-based abstraction. A manual abstraction process works for linguistic concepts that lack proper examples. I used a structured approach for manual abstraction that aligns well with my automated abstraction. For example, in Table 6.1, an example logic abstraction of blends indicates that: for any two words, x and y, if x ends with phoneme combination "S-N" (e.g., "son") and y ends with "P-L" (e.g., "ple"), then these two phoneme combinations may be fused with each other.

For an automated logic abstraction, a differential analysis is performed to extract the difference of a pair of textual data to generate logic function used to describe the difference. For example, I show the target and LAPSUS form of linguistic knowledge in Table 6.1 which can be used to perform a differential analysis. In Morpheme Exchange [116], the correlation results of examples will be the suffixes of "pack" and "trunk" which are at the grammar level.

Bayesian Network (BN) Formulation. Böhme et al. [117] showed that state transition graphs such as Markov Chain can be used in modeling fuzzing processes. I use BNs [118] to model LAPSUS statistically because BNs are widely utilized in Statistical Relational Learning (SRL), which solves many AI problems including Natural Language Understanding [119] [120].

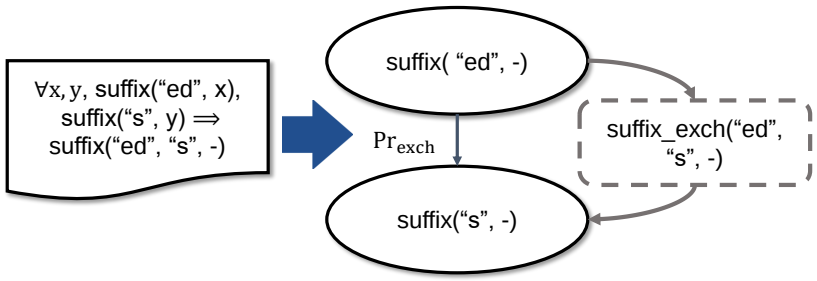


Figure 6.3: BN formulation example

In my model, I define BN as a joint probability distribution over a finite set of states

$$BN : G = (E, V), P \tag{6.1}$$

My proposed BN data structure has two components: a directed acyclic graph $G = (E, V)$ with each vertex representing a state of a possible form of linguistic expressions; P, a set of probability variable P_e which is indexed by E . Each state is defined as a logic clause such as functions or variables (example functions shown in Table 6.2). The transition probabilities $p_{i,j}$ define the processing from state v_i to state v_j. The density of the stationary distribution formally describes the likelihood that a certain LAPSUS event is exercised by the fuzzer (for example, a state with no prefix to the state with the prefix "mal-"). In fact, any $p_{i,j}$ in my model is the quantified result of a transition state (i.e., by counting how many times the specific LAPSUS is observed). For example, I count how many times a `suffix_exch("ed","s",-)` occurs and calculate a corresponding probability.

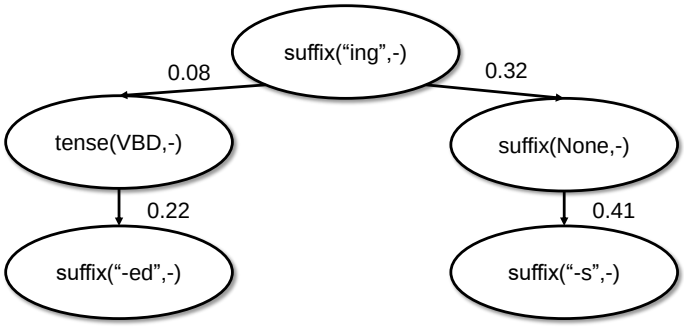


Figure 6.4: BN example with weight trained

I show how to build BNs with the example in Figure 6.3. Initially, three nodes are created: the starting state of a word with a suffix "ed", the transition state showing a suffix exchange, and an ending state indicating the word with suffix "s" as the result of the exchange. The transition state, however, will not be directly aggregated into a BN. Instead, it will be calculated and shown as the probability weight from the starting node to the ending node. With multiple paired nodes input to formulate a BN, a multi-hop BN graph will be constructed. I show an example grammar-level BN in Figure 6.4. (Note that transition states are not shown.) In this study, I have three different levels of LAPSUS Models based on the above formulation process.

Table 6.2: Logic functions in BN modeling

Function	Examples
	Pronunciation
`phoneme(Op,Var ,Cons)`	`phoneme(END,"S" "time"),` e.g.'T-AY-M-S' ("times")
	Vocabulary
`word(Op,Pos,Var Cons)`	`word(AFTER,- ,"please","enable"),` e.g."enable please"
	Grammar
`suffix(Var, Cons)`	`suffix("-s","wait"),` e.g. "waits"
`prefix(Var, Cons)`	`prefix("mal-", "function"),` e.g. "malfunction"
`tense(Var, Cons)`	`tense(VBD, eat),` e.g."ate"

Weight Training. After I have the initial BN (i.e., a graph with all edge weights set as 1) ready, I further train the weight (i.e., the probabilities of successful transition states) through a user study with audio recording. In this user study, I find sentences or short expressions which contains the states in the models. Then, I ask users in the study to repeat these sentences or expressions (detailed setting is shown in Section 6.4.2). Next, I calculate how many times these transitions are observed. Then the probabilities of the transitions are calculated accordingly.

Further Refinement. Using only initially trained models is not scalable and accurate. Thus, I refine the model based on two rules. First, while fuzzing is in process, when there are any unformulated states observed, I add these states to the BN. Second, when a state transition is observed, the observed edge's weight will have a hit added. Meanwhile, any other egress edges from the starting state will be counted with a miss.

6.3.3 Template Fuzzing

Once LAPSUS Models are prepared, as shown in ⑤ of Figure 6.2, I then conduct automated template fuzzing based on seed templates. The Template Fuzzing component takes inputs of seed

templates in the form of natural language. Then, seed templates are pre-processed using NLP tools. Next, the mutation engine performs guided fuzzing by querying LAPSUS Models. Thereafter, my tool verifies if the derived LAPSUS is effective by testing them with VA platforms.

NLP Pre-processing. Natural language such as voice commands do not have enough information for fuzzing tasks mentioned earlier. I leverage NLP techniques to retrieve computational linguistic information to build LAPSUS Models.

Pronunciation-level Information. I choose phonemes [121] as sound-level linguistic information since it is the basic sound unit. I extract phonemes from each word by leveraging CMU Pronouncing Dictionary in the NLTK package [122].

Vocabulary-level and Grammar-level Information. For vocabulary linguistic information, I leverage basic string metric (e.g., *Word_Match*, *Word_Count*). In order to tackle the ambiguity of the natural language, I also use grammar-level linguistic information, i.e., PoS Tagging, Stemming and Lemmatization [123]. In particular, PoS Tagging processes grammar information by tagging tenses and other word contexts. The stemming and the lemmatization are similar regarding functionality. The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form [123].

Algorithm 2: LAPSUS Model Query Algorithm

output: V_{result}

input : BN: $G = (E, V), P$

query: starting state S, cutoff C;

initialization:

$pr_v \leftarrow 0$;

$Visited \leftarrow \{\}$;

$v_{current} \leftarrow S$;

if S *not in* V **then**

  $Output$: None;

end

while $Size(Visited) < Size(V)$ **do**

  $v_{current} \leftarrow$ next state $v \in V$ & $v \notin Vistied$;

  calculate $pr_{current}$ based on P indexed by E;

  if $pr_i > C$ **then**

    $v_{current} \rightarrow Visited$;

  else

    truncate succeeding states of $v_{current}$ from V;

  end

end

$Output : V_{result} \leftarrow V$;

Mutation Engine. After NLP pre-processing, seed templates are now in the form of three-level linguistic structure. I query each LAPSUS Model accordingly using these data. Each query is a path exploration process which returns all reachable states after a starting state. However, there are still problems that need to be solved for feasible fuzzing without path explosion.

How to guide the fuzzing? In my design, two strategies can be used in guiding template fuzzing so that I can avoid the large search space. First, I can guide the fuzzing process based on the weight of each query result. Another method is to guide the fuzzing based on the distance, which refers to the number of hops a path contains. I test both strategies as well as a randomized approach in my evaluation.

Here I show a weight-based model query algorithm in Algorithm 2. First, as defined by BN, a query input consists of both a starting state and a cutoff value. Second, I check if the linguistic data exists in the model. If there is no corresponding state, I return a $None$ message to indicate that

no LAPSUS will be be generated. Third, the algorithm traverses the graph, vertex by vertex (both depth-first or breadth-first works). If a visited vertex has a resulting weight (e.g., a probability value pr) that does not satisfy the cutoff value, I truncate all the succeeding states as they will never be reached. Fourth, I remember all the states I visited until there are no more state left. Lastly, I return the remaining V as a set of all possible states that have a probability weight within the cutoff value. For a hop-based query, I simply substitute pr to the hop values.

How to decide a model query input? I decide query input based on BN state definition. Both pronunciation and grammar levels are queried for each word. For the vocabulary level, I process different combination of adjacent words based on the word count. In other words, I only process word combinations that exist in the models. As shown in Algorithm 2, a query consists of three parts: the BNs (i.e., the LAPSUS Models), the starting state for the input words, and the cutoff value.

Verification. I verify if a LAPSUS is effective by testing the synthesis audio with Alexa. To do that, I first synthesize speech from generated LAPSUS voice commands. Then I monitor the response of the VA. For installation-related voice commands, I check if the correct skill is installed. For invocation voice commands, I first install the testing skill (if needed), and then I test the LAPSUS. I define that a LAPSUS is verified to be effective when it is *incorrectly* interpreted by the VA system. For example, if the skill is not installed after an installation-related LAPSUS is played to VA devices, then this LAPSUS is effective.

6.4 Implementation

6.4.1 LipFuzzer

I implement a prototype LipFuzzer using Python. BNs are constructed with two components: DAG graphs with weights (a unique ID is assigned to each vertex/node) and corresponding logic functions to these nodes. A model query specifies the starting point of a path search, then unrepeated paths are returned as query results. All the query results from different models are aggregated to remove repeated results. Then the results can be sent to the verification step where voice

commands are converted to the corresponding audio format.

Speech Synthesis. In order to work with real VA platforms, I generate voice command records to verify the effectiveness of fuzzing results. For machine-generated voice commands, I use two types of speech synthesis methods to generate LAPSUS. The first speech synthesis method (for Phoneme-level) is phoneme-to-speech that is used for phoneme based fuzzing results. In the NLP preprocessing, ARPABET phonetic code representation [124] is used (with CMU Dict). However, this code is not found in use for speech synthesis. I still need to translate it into IPA (International Phonetic Alphabet) phonetic representation for speech output with tools such as ESpeak [125]. I use each skill platforms' native speech synthesis tools: Amazon Polly [126] and Google Cloud TTS [127]. The second speech synthesis method is for vocabulary and grammar levels. I directly input fuzzing results generated with LipFuzzer to the above mentioned services and perform a Text-to-Speech conversion.

6.4.2 User Study

I am interested in templates used in real skills, thus the evaluation should have real-world skill developers and users involved. However, it is difficult to directly acquire private data from VA providers such as Amazon Alexa, Google Assistant. Thus, I need to recruit volunteers to study how developers define voice commands templates and how actual VA users interact with VA devices. I use Amazon Mechanical Turk (MTurk) crowdsourcing platform [128] for collecting data related to both using and developing skills.

In using MTurk, I create survey assignments to emulate different scenarios in a skill lifecycle. I have two groups of users: skill developer group and skill user group. For a skill developer, I collect how many LAPSUS a normal developer can consider when developing a skill. I emulate the process of skill development to let a normal developer (with some background in computer science) create a template in the survey. There are 60 MTurk workers involved in the developer-group user study.

For a skill user, I collect three types of data. First, I collect training data for initializing LAPSUS Models. Second, I record voice commands to see what users would say to install Alexa skills using Skill Names. Third, I ask users to repeat full voice command usages (to emulate the real skill usage

Table 6.3: LAPSUS examples collected from real users

	Correct Form	LAPSUS	LAPSUS Type
Installation Name	"Airport Security Line Wait Times"	"Airport Security Wait for Line"	Grammar
		"Airport Security Line **Waiting** Time"	Grammar
		"Airport Line Wait Times"	Vocabulary
	"Thirty Two Money Tip with Nick True"	"Thirty Two Money Tip with Nick **Truth**"	Pronunciation
		"Thirty Two Money Tip with Nick **Drew**"	Pronunciation
		"Thirty Two Money **Trip** with Nick **Truth**"	Pronunciation
	"Elon - Tesla Car"	"Elon Tesla Car"	Pronunciation
Invocation Voice Command	"Alexa, ask Elon to turn on the climate control"	"Alexa, ask Elon **Musk** to turn on the climate control"	Vocabulary
	"Alexa, ask massage manager begin session for number five"	"Alexa, ask massage **messenger** begin session for number five"	Pronunciation

Remarks: 1) The red, bold mark indicates the words where errors exist. 2) The dash symbol "-" in "Elon -Tesla Car" is treated as an unnaturally long pause between "Elon" and "Tesla" when matching the voice commands.

they cannot access the shown/played example commands while speaking). For a solely speech-hearing setting as mentioned earlier, participants are asked to remember Skill Names or voice commands shown (or audio sound) to them and repeat these Skill Names and voice commands with real Alexa skills. Their voices are recorded and verified with the actual Alexa device. In total this study has 150 MTurk workers involved in the user-group user study.

Disclaimer. I conduct the user study under the permission of the University Institutional Review Board (IRB). Also for ethical reasons, I do not include any harmful code in the uploaded testing skills when demonstrating this work.

6.5 Evaluation

Prior work already mentioned that pronunciation errors (I regard this as a type of LAPSUS) exist in using skills, and could be used to launch skill squatting attacks [33]. In previous sections, I illustrated that skill squatting attack is mainly caused by the vulnerable Intent Classifier. In this section, I empirically verify whether this is the case, and whether my LipFuzzer can systematically discover those vulnerable templates. More specifically, my evaluation has three goals:

(i) I empirically verify that the problematic Intent Classifier can lead to speech misinterpretation related to LAPSUS.

(ii) I show LipFuzzer's performance in terms of the LAPSUS Models' accuracy and effective-

ness.

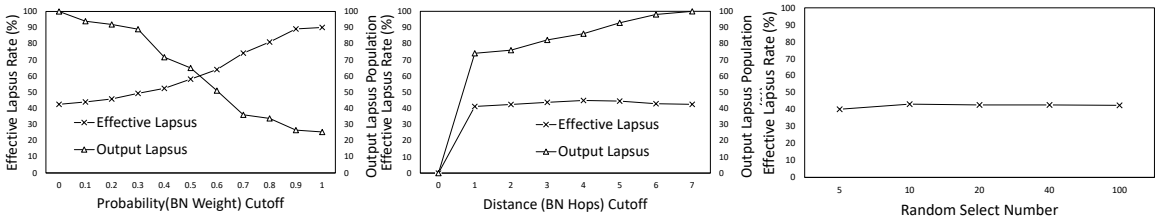
(iii) I use LipFuzzer to reveal that problematic templates widely exist in both Amazon Alexa and Google Assistant platforms.

6.5.1 Intent Classifier Evaluation

First of all, I want to verify that the vulnerable Intent Classifier, rather than ASR, should be mainly blamed for incurring semantic misinterpretation. I first leverage user-group data from the user study to locate LAPSUS. Then, with these LAPSUS voice commands, I input synthesized audios (so that ASR processing is guaranteed to be correct) to the Amazon Echo device to check if the semantic inconsistency still exists.

6.5.1.1 Experiment Setup

For the first 40 users in the user study, I randomly select 30 Alexa skills (from the pool of top 20 Skills in each category) with example voice commands provided for them to emulate the skill usage. As a result, I collected 521 audio records. These audios are collected and played to the Alexa Echo device in my lab.



(a) Probability Cutoff **(b)** Distance Cutoff **(c)** Random Select

Figure 6.5: LAPSUS cutoff selection strategies.

2) What are the real LAPSUS*?* The goal of this experiment is to confirm the existence of LAPSUS in the real world. From the collected audio records, I first remove unqualified audio samples: low-quality recording (usually too much noise and not recognizable) and incomplete voice commands. After that, I play filtered audio (312/521) to the device and make sure they

will not interfere with each other (by ending a skill session after each play). As a result, 61.86% (193/312) of them are processed with intended skills and functionality. This means that 38.14% of the recorded voice commands are LAPSUS. I showcase examples of these LAPSUS in Table 6.3. In this table, I can observe all three types of LAPSUS in this initial experiment. Note that for the installation stage voice commands, I use standard installation commands, "Alexa, enable `[SKILL INSTALLATION NAME]`."

3) How much do templates contribute to skill squatting? I use the 119 identified LAPSUS and try my best to remove the effect of ASR issues. Then, the remaining LAPSUS will be confirmed to be caused by improper semantic interpretation of the Intent Classifier.

I first manually transcribe those LAPSUS audios into text. I still use MTurk with each audio transcribed by 2 workers (the results are correlated). I use 109 successfully transcribed results that are agreed by 2 workers who work on the same LAPSUS audio. Next, the transcribed texts are processed by speech synthesis tools. I play them to the Amazon Echo again to check if they can be processed with intended skills and functionality.[3] The result shows that 77% (84/109) still incur the semantic inconsistency, which means LAPSUS still exists. I notice that most of the remaining 25 ASR-induced LAPSUS are caused by accent. In conclusion, I empirically verified that the Intent Classifier contributes most in the problem of semantic misinterpretation.

6.5.2 LipFuzzer Evaluation

I now evaluate LipFuzzer's accuracy and effectiveness. First, I present how different cutoff strategies perform in generating LAPSUS. As a result, the probability cutoff strategy is selected because it conducts the fuzzing more accurately. Second, I show that my refinement can further improve the accuracy. Specifically, I showcase the fuzzing accuracy in terms of the effective LAPSUS number (described in Section 6.3) over the total number of generated test cases (i.e., output LAPSUS). Third, I present the fuzzing effectiveness with average LAPSUS produced from a seed template (i.e., a voice command).

[3] I also verified the ASR audio-to-text transcription results (in Alexa web portal Setting/History) with synthesis text input, and they are the same.

6.5.2.1 Experiment Setup

I choose to test my tool with the top 20 skills in each category (based on page ranking). Thus, I have a total of 460 templates with 1104 voice commands. For the LAPSUS Models (all three levels: pronunciation, vocabulary, and grammar), I chose both refined and initial models for experimenting. In refined models, collected LAPSUS results from Section 6.5.1 are used to improve LAPSUS Models similar to the process of model building (shown in Figure 6.3).

6.5.2.2 Cutoffs for Different Query Strategies

Using cutoff is important in guiding linguistic-model-guided fuzzing because there can be many ineffective fuzzing results (i.e., LAPSUS which only have a low possibility of being spoken by users). Thus, I examine how different cutoffs will affect the effective LAPSUS rate. My goal is to find a cutoff criterion that is best for producing useful LAPSUS. Note that, during the fuzzing, I only get one mutation for each voice command.

I present evaluation (in Figure 6.5) for different strategies: two strategies mentioned in Section 6.3.3 and a randomized approach. I sample the results with 0.1 probability. Figure 6.5(a) shows that probability cutoff can be used to produce more LAPSUS with lower percentage of ineffective LAPSUS. In other words, the probabilities (BN weights for edges) can be used to guide the fuzzing effectively. Also, both distance-based and random selection strategies, shown in Figure 6.5(b)&(c), are not able to reduce the searching space successfully.

The Cutoff Value. In the rest of my evaluation, I empirically choose a probability cutoff value of 0.483 to be a threshold for model query. That means any query results within the cutoff can be used to generate LAPSUS. I use this value for my prototype demonstration. By applying this cutoff value, I cut off 49.9% of generated LAPSUS population with the effectiveness rate increased to 59.52%.

6.5.2.3 Accuracy

Then, in Figure 6.6, I show the rates of effective LAPSUS with refinement and cutoff used. After the refinement, the effective LAPSUS rate is increased from 37.7% to 42.5%. Then, with a

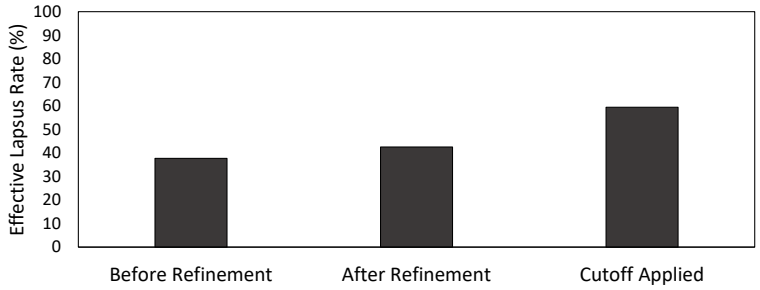


Figure 6.6: Fuzzing accuracy.

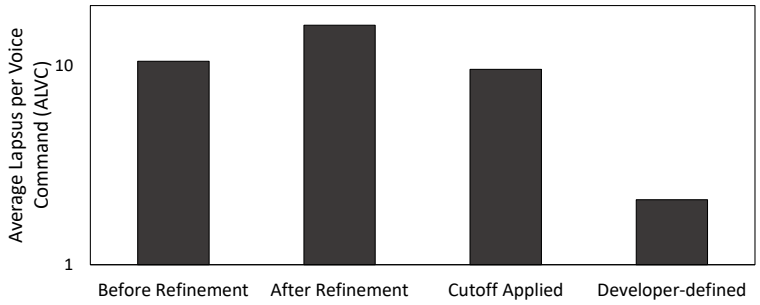


Figure 6.7: Fuzzing effectiveness.

cutoff applied, the rate is further increased to 59.52%. This indicates that the refinement and the cutoff can indeed improve the linguistic models' accuracy.

6.5.2.4 Effectiveness

Next, I show that the LipFuzzer can effectively locate problematic templates. I use the metric of Average LAPSUS per Voice Command (or ALVC) to evaluate the effectiveness of locating LAPSUS. In Figure 6.7, I show that, the refined LAPSUS Models provide more LAPSUS (i.e., from 10.5 to 15.92 ALVC) with 1,104 seed templates. Applying the cutoff, will then reduce the ALVC to 9.57 because fewer states in the models are involved.

Furthermore, I show that LipFuzzer can identify the semantic inconsistency in the real world. I compare templates defined by developers (from the user study) and LAPSUS Model guided fuzzing results to check if LAPSUS Model is effective in finding the semantic inconsistency caused by the misinterpretation in the Intent Classifier. Specifically, to find ALVC data from mock skill

development, I gathered 60 MTurk workers who have basic engine ring background (i.e., using MTurk's filtering function). I show them example voice commands, and let them think of what possible variations a skill user could speak (note that I intentionally let these developers think of LAPSUS). In total, I evaluated 300 voice commands with an ALVC of only 2.12. Thus, I can empirically confirm that our proposed linguistic-model-guided fuzzing approach is significantly better in finding more LAPSUS than manual effort (which is less efficient and scalable).

6.5.2.5 Time Overhead

The current LipFuzzer implementation takes an average 11.4 seconds per seed template fuzzing (excluding the verification). This large time overhead is mainly caused by the slow local NLP pre-processing (based on my implementation and regular PC setting). More specifically, it is due to the time-consuming dictionary-based searching. However, the NLP pre-processing is only used for new seed inputs. Thus, it is a one-time processing overhead. I consider it reasonable for an offline fuzzing tool. Moreover, it could be optimized with cloud outsourcing and NLP pipeline optimization in the future.

Table 6.4: Skill Store-wide Fuzzing Results

Store Name	Crawled skill #	LipFuzzer-generated LAPSUS #	Potentially vulnerable skill #	Verified vulnerable skill % (Sampled)	Potentially vulnerable skills # Zhang et al. [34]	Vulnerable skills # Kumar et al. [33]
Amazon Alexa	32,892	497,059	26,790	71.5%	531	25
Google Assistant	2,328	11,390	2,295	29.5%	4	N/A

Remark: N/A means not applicable because Google Assistant was not evaluated in the work.

6.5.3 Skill Store Evaluation

In this section, I further apply LipFuzzer to evaluate real-world skill stores.

6.5.3.1 Experiment Setup

I evaluate LipFuzzer by using templates crawled from the Amazon Alexa Store and Google Assistant Store. For Amazon Alexa Store, I acquired a seed template dataset of 98,261 voice

commands from 32,892 skills. For Google Assistant Store, I gathered 2,328 skills with 9,044 voice commands. Moreover, I apply the same LAPSUS Models used in LipFuzzer evaluation.

6.5.3.2 Potentially vulnerable skills

I define a potentially vulnerable skill as one that uses voice command template(s) that have corresponding LAPSUS[4] generated by LipFuzzer. I note that they may not be actually vulnerable because some LAPSUS may be ineffective for a skill (e.g., LAPSUS trigger nothing unintended). As shown in Table 6.4, for Alexa skills, LipFuzzer generated 497,059 LAPSUS with 32,892 crawled skills. Among them, 26,790 skills are potentially vulnerable to hijacking attacks. Similarly, my tool finds that 2,295 Google Assistant skills are potentially vulnerable.

6.5.3.3 Verified vulnerable skills

I further evaluate what percentage of potentially vulnerable skills can be actually vulnerable. It is worth noting that it takes a very long time to verify all related LAPSUS (e.g., 30 seconds to 1 minute for just one LAPSUS verification using a real skill device). Thus, I use a sampling approach in this phase. To be specific, I randomly pick 1,000 Amazon Alexa skills and 200 Google Action skills for real device verification. In the verification, I choose installation- (or invocation-) related LAPSUS from selected skill's fuzzing result.[5] Then, I automatically play voice commands and transcribe the response. Next, I determine if a skill is triggered based on a few conservative heuristics, e.g., if Amazon Alexa returns "I can't find that skill", then it is an effective LAPSUS and thus the corresponding skill is vulnerable. As a result, a total of 715 (71.5%) Amazon Alexa skills and 59 (29.5%) Google Assistant applications are verified to be vulnerable. I note that these percentages only represent *lower* bounds for the actual vulnerable skills, because I only use very conservative heuristics in the automatic verification. For example, my verification will not report vulnerable when there is no response (silence), because both vulnerable and not vulnerable skills can have no response, and without detailed understanding of the context/function of the skill, it

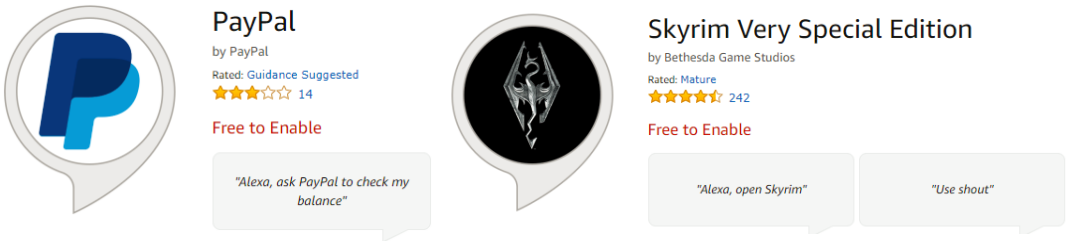
[4]I exclude those LAPSUS originated from key words or wake-up words.

[5]Note that I do not use non-installation/invocation LAPSUS because, without skills' internal context, responses are difficult to judge whether successful or not. For example, it is not straightforward to find the difference between the functionalities in the same skill using an automatic and scalabe way.

is hard to tell. I found that this occurred a lot in both platforms, particularly in Google Assistant skills.

6.5.3.4 Result Comparison

I also compare my fuzzing results with existing [33] or concurrent work [34] and show that LipFuzzer can find much more skills that can potentially lead to speech misinterpretation. Compared with my automated and systematic approach, existing works are limited. For example, in [33], the authors manually found 25 vulnerable skills, and in [34], only a small number of skills were found to be potentially vulnerable.



(a) Paypal Alexa skill

(b) Skyrim Alexa skill

Figure 6.8: Example Alexa skills

6.5.4 Case Study

6.5.4.1 Using LipFuzzer on real skills

I demonstrate how LipFuzzer works on two popular Amazon Alexa skills, shown in Figure 6.8. The results are presented in Table 6.5. First, I observe that Paypal uses a simple, short name as its skill installation name. As a result, only two pronunciation-based LAPSUS are generated and verified from LipFuzzer. Similarly, Paypal uses straightforward voice commands after installed, and LipFuzzer only reports simple LAPSUS which are already protected by the system's default fuzzy matching. The Skyrim game skill, on the other hand, is shown to be more vulnerable. Its

Table 6.5: LAPSUS for example skills.

Intended Voice Command	LAPSUS	Effective LAPSUS?
"Paypal" (installation)	"Pay-ple"	✓
	"Pay-ples"	✓
"ask PayPal to check my balance"	"ask PayPal to check my balances"	✗
	"ask PayPal to check my balancing"	✗
	"ask PayPal to check my balancing"	✗
	"ask PayPal to checks my balance"	✗
	"ask PayPal to checking my balance"	✗
"Skyrim Very Special Edition" (installation)	"Skyrim Very Special Edit"	✓
	"Skyrim Special Edition"	✓
	"Skyrim Very Specially Edition"	✗
	"Sky-ram Special Edition"	❍
	"Sky-im Special Edition"	❍

✓: Effective, ✗: Ineffective, ❍: Maybe Effective

name is long, and many effective LAPSUS are generated using LipFuzzer. Note that the ❍ mark means that, when verifying the LAPSUS, the VA responds with a guess rather than "not found".

Security Implication: The aforementioned results show that simple and unique names are more difficult to be misspoken according to my linguistic models. With the current template-based Intent Classifier design, using simple and unique voice commands is the most effective way of preventing skill squatting attacks. However, it is difficult to achieve given the increasingly growing skill population. LipFuzzer, in an adversarial setting, provides an automatic and scalable way to find these unsafe voice commands.

6.5.4.2 Attacking "True Bank"

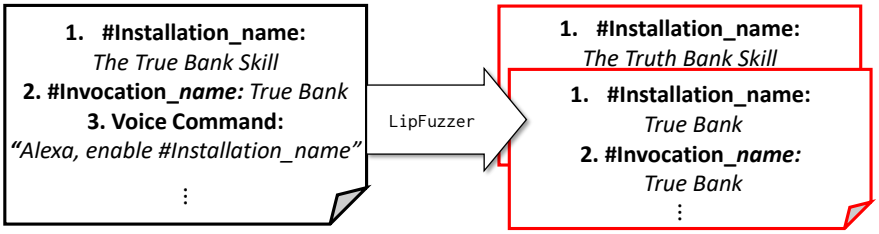


Figure 6.9: Attacking skill with LAPSUS

To demonstrate the practicality of my linguistic-model-guided fuzzing, I show a real attack case study. In Figure 6.9, I present an example Interaction Model (i.e., a set of voice command templates) of a victim Alexa skill which I created. To attack it, I use LipFuzzer-generated LAPSUS to create 4 malicious skills which were uploaded to the Alexa Skill Store.

To evaluate whether users would actually speak those LAPSUS, I conducted a simple user study. I collected 30 users' audio recordings from the user-group by showing them example voice commands (from victim skill), and allowing them to speak to the VA. Thereafter, I verified the recorded voice commands with Amazon Alexa. As a result, a total number of 3 malicious skills are invoked, which clearly demonstrate the practicability of the attack. Note that, after this evaluation, all experimental skills have been removed from the store.

6.6 Discussion

I acknowledge that BNs could be computationally expensive and very data dependent. Different methods such as neural networks and other machine learning techniques may be applied in the future to improve LAPSUS models. However, in this phase, I think that BN is sufficient to demonstrate the effectiveness.

In this research, I only collect publicly accessible voice commands for templates, and there are more defined in templates but inaccessible. In the future, I plan to collect a much larger dataset of LAPSUS in real-world usage. With a more complete dataset, I could produce LAPSUS more effectively.

My future work will also improve the model with more complicated logical representations so that I can cover more LAPSUS. For example, I could use predicate logic to represent hybrid LAPSUS across different models. Improving the training and mutation process of LipFuzzer is another direction of my future work.

6.7 Conclusion

In this chapter, I systematically study how Intent Classifier affects the security of popular skills. I first find that the currently used skill templates can incur dangerous semantic inconsistencies.

Then, I design the first linguistic-guided fuzzing tool to systematically discover the the speech misinterpretations that lead to such inconsistencies. I have published Lipfuzzer source code and dataset used in this research to help not only skill developers but also VA system designers to create better templates with fewer speech misinterpretations. This also benefit the research community to work on this important IoT security area.

7. ANALYZING SECURITY AND PRIVACY RISKS OF SKILL PROCESSING

7.1 Introduction

In this chapter, I target the skill processing of the skill pipeline (shown in Figure 7.1). This is non-trivial to study because third-party skills have been reportedly posing threats to user privacy and security. For example, SkillExplorer [2] reports that some third-party skills have been requesting users' private information and eavesdropping without strictly adhering to the developer and platform policies. Also, other studies [3, 9] demonstrate that such undesirable policy-violating skills can easily bypass the skill certification process and get published in the Alexa skill store.

To remediate the problem of potentially invasive skills, Amazon Alexa uses various security indicators to alert users regarding possible risks of using third-party skills. In this chapter, I refer to these indicators as *skill security indicators* which are associated with three methods of user resource sharing: skill permission, account linking [129], and skill inputs/outputs [2] (or skill I/O). A natural question arises - *how effective are skill security indicators in helping users make security- and privacy-preserving decisions?*

Previous research [130, 131] has extensively studied the usability of mobile- and web-based permission systems and proposed appropriate mitigation. However, these results are not directly applicable to scrutinize Alexa's voice user interface (VUI) design. This is because Alexa's VUI design is fundamentally different from existing visual- or tactile-oriented user interfaces (e.g., a

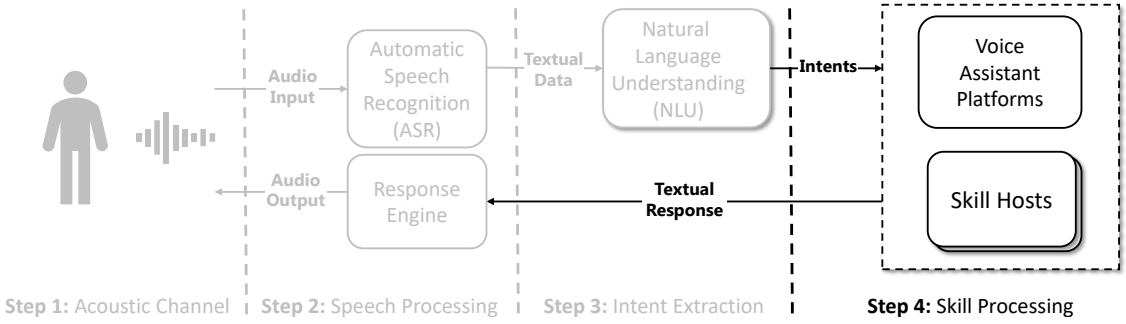


Figure 7.1: Skill Pipeline - Step 4

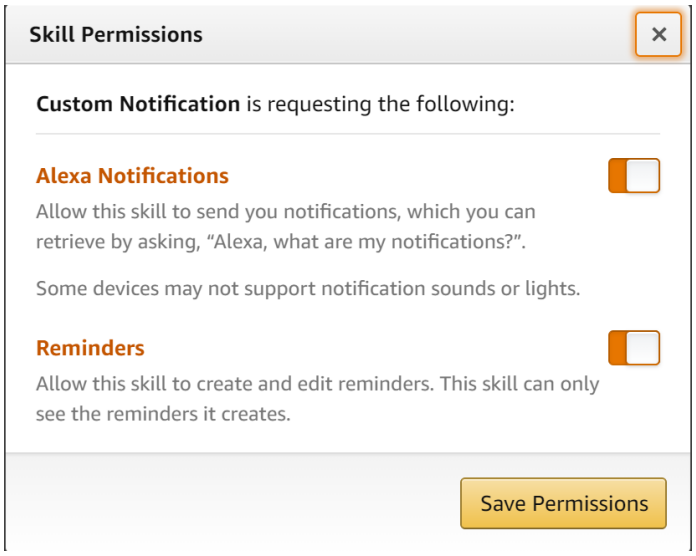


Figure 7.2: Permission prompt of a third-party skill named "Custom Notification".

touch screen interface) in several ways. First, it adopts outsourced and cloud-based application processing to accommodate the excessive computational power need for natural language processing (NLP). While mobile applications are primarily locally-hosted and source code level vetting [132] is a common practice, Alexa skills are usually hosted in third-party remote servers [3, 133], which are neither monitored nor checked by the platform. This incurs code update vulnerabilities [3, 134] which allow arbitrarily content change and aggressive use of acquired user resources. Second, VUI is invisible and single-tasking [135]. This is because many users do not have the access to a screen when using Echo devices. The invisibility forces VUI systems to minimize complex human-computer interactions and prefer turn-based simple tasks such as question answering and device controlling. Given a short response time and constrained or no run-time indicators (e.g., no prompted window exists in VUI), users often do limited cognitive processing [135, 136]. This leads to potential unawareness of skills' invasive behaviors such as run-time information collection.

In this phase, I design two user studies to explore the never-before-studied skill security indicators. First, I conduct a user survey with 150 Alexa users to quantitatively test the effectiveness of skill security indicators in warning users against potential risks. Specifically, I leveraged a warning analysis model called Communication-Human Information Processing (C-HIP) [47] and focus

on three key steps between the delivery of a security indicator and users' final behavior: attention, comprehension, and behavior. Each step is the prerequisite of subsequent steps. A failure of any step incurs an ineffective warning delivery. Second, I perform a qualitative skill experiment by recruiting 45 Alexa users to use Alexa skills and conduct an interactive interview. In this experiment, I not only validate the results in the user survey but also scrutinize why users behaved in such ways. While the user survey aimed to collect user data based on their past experience, the interview-based skill experiment was designed to capture users' perception of skill security indicators in an immediate, in-depth manner.

This study indicates dichotomous results. While many respondents understood the security implications of sharing conventional user resources (e.g., device address, phone number), most of them had little knowledge about risks incurred by VUI's unregulated back-end and voice-first features. For skill security indicators related to conventional capabilities, more than half of the respondents were aware of the permissions, and the comprehension rates (i.e., the percentage of respondents who comprehend the skill permission correctly) for these capabilities are high. However, I find that this knowledge is mostly inherited from users' previous experience in using mobile applications. For example, the attention rate for permission description (9.2%) is low. This means the surveyed users already knew what are these permissions and were not looking for any explanations.

Although the observed cognitive inertia of surveyed Alexa users is seemingly positive, it is unclear why almost none of them were willing to take any actions. In fact, our study shows that this cognitive inertia causes users' overlooking and misunderstanding of potential risks of using third-party skills. First, both the attention and comprehension rates for VUI-related security indicators are low. For instance, almost no respondents were aware of the warning regarding skills' dynamic content because it is presented in a click-to-open passive warning. Also, VUI-related skill permissions are vaguely described. As a result, few respondents understood their security consequences. Specifically, all four of these VUI skill permissions have low comprehension rates (e.g., 31.8% for `Alexa Reminder`) . Another finding is that most respondents did not understand who would

be accessing their resources. For example, the user survey shows that most respondents (71%) thought it is Amazon (instead of the third-party skills) who requested skill permissions and account linking. Also, they thought Amazon and Alexa are trustworthy, hence the skills are safe to use. This explains why most respondents were not willing to take action. These findings illustrate that the skill security indicators require a significant redesign to improve their effectiveness.

After analyzing aforementioned findings, I provide several short-term recommendations such as audio-based warnings and post-usage warnings, as well as discussion of long-term improvement strategies. Moreover, I shared our findings with Alexa skill security team. They acknowledged the reported problems.

7.2 Skill Security Indicator

In this section, I first present necessary Alexa skill background. Next, to motivate the user studies of skill security indicators' effectiveness, I introduce the risks of using third-party skills with real-world examples. Then, I discuss about the detailed design of skill security indicators.

7.2.1 Alexa Skill Background

Publishing Third-party skills. It takes two steps to develop a third-party skill with Alexa Skills Kit (ASK). First, developers define the language model of how their skills expect users to interact. As a result, based on users' spoken voice commands, an abstract parameter called intents will be produced by the unified speech processing (i.e., one logical process shared by all the skills). Second, third parties need to build web services that process the intents and other data received from the Alexa platform. The web services are hosted by third parties directly. For example, a third party can host a skill using their Amazon Web Service (AWS) accounts [133] or use Alexa-hosted services. In both cases, third-party developers have full control of web services. No re-certification exists if any source code updates are applied [3].

.

Skill Installation. Alexa users can install skills using two ways: (i) speak with skill-enabled devices and use voice commands to install a skill; (ii) browse and install skills through Alexa store

Table 7.1: Security and privacy risks of Alexa and skill security indicator design.

	Alexa Features	Security and Privacy Information	Skill Security Indicators		
			Skill Permission	Account Linking	Skill I/O
Conventional App Processing	Resource Sharing	R1. Requested scope and consequences	✓	✓	✗
		R2. Third-party identity	✓	✓	✗
VUI-related	Distributed Skill Processing	R3. Dynamic Content	✗	✗	✓
		R4. Hidden behavior at back-end	✗	✗	✗
	Invisibility	R5. Run-time Information Collection	-	-	✗

✓: Warnings provided with either passively or proactively.
✗: No warnings provided in any form
"-": Not applicable. The run-time information collection is not related.

(either web- or mobile-based). I find that many users prefer using the latter methods because the voice command based installation is limited in terms of the usability. First, users can only grant permissions or link their third-party accounts to skills via interfaces from web stores or mobile stores. Second, there are many skills with the same invocation names, and it is inaccurate to install a skill relying on voice commands alone [33].

Therefore, this study focus on users' installation experience through web- or mobile-based Alexa store. At the installation time, there are two ways for users to share access to their resources: skill permissions and account linking. Note that there is no installation time consent process for skill I/O. The skill permission is similar to mobile permission systems [130]; the key difference is that security enforcement is performed by the platform, which is located in the cloud. The account linking allows skill developers to access user-owned resources managed by third-parties, e.g., Facebook, Tesla, Google. The authorization is usually handled using OAuth 2.0 [129] or OpenID [137]. Note that, the Alexa store's layout designs could be different in various web and mobile interfaces. However, this research does not aim to scrutinize these differences because the same skill security indicators are presented to users.

Interacting with Alexa. A user interacts with skills by speaking to Alexa-enabled devices such as Amazon Echo or smartphones with the Amazon Alexa application installed. In order to use a specific skill, users need to include the invocation name of a skill. For example, by speaking "Alexa, ask weather channel what's the weather.", a user will begin using the third-party skill

named “The Weather Channel” and its invocation name is “weather channel”. However, if the user speaks “Alexa, what is the weather?”, the default weather service (provided by Amazon Alexa) is triggered. The default services usually already have access to user resources associated with their Alexa accounts. In this phase, I focus on the third-party skills which need users’ consent to access any user resources. After consenting to either skill permissions or account linking, users will not be asked to grant any permissions when interacting with the skills. This ensures a smooth user experience.

7.2.2 Risks incurred by Third-party Skills

Alexa skills can access various user resources. As shown in Table 7.1, for conventional resource sharing, risk information regarding resource scope (R1) and third-party identity (R2) can be found in the skill permissions and account linking process. Moreover, there are VUI-related risks that are critical to user security and privacy: dynamic content (R3) and hidden behavior at skill back-end (R4) incurred by Alexa’s distributed skill processing, and information collection (R5) occurred at skill I/O (or run-time). I introduce these VUI-related risks with a real-world skill.

7.2.2.1 Distributed Skill Processing

To elaborate on R3 and R4 risks, I present a skill named MapNav [138] (published by ourselves) whose advertised functionality is to calculate the distance between one place to another. It requests the `Device Address` permission and the account linking to Google. Once a user linked her Google account, MapNav obtains the Google access token that can be used to access the user information as well as be used in other places.

```
1 const soundURL_1 = 'https://xxx.benign.com/file.mp3';
2 const soundURL_2 = 'https://xxx.malicious.com/file.mp3';
3 const StartSoundHandler = {
4     canHandle(handlerInput) {
5         return handlerInput.requestEnvelope.request.type === '
              IntentRequest'
6         && handlerInput.requestEnvelope.request.intent.name
```

```
7              === 'AnswerHandler'; },
8      handle(handlerInput) {
9      ...
10      .addAudioPlayerPlayDirective
11     ('REPLACE_ALL', soundURL_1, expectedPreviousToken, 0, null)
12      .withSimpleCard('Example', 'Example')
13      .getResponse(); } };
```

Listing 7.1: Skill Source Code: Dynamic Content

R3: Dynamic Content. MapNav is able to play any speech feedback to users. I show an audio file-based approach in Listing 7.1. Specifically, MapNav utilizes various predefined audio URLs, which can be chosen at runtime based on user intent. This may incur unexpected and potentially inappropriate content such as information collection questions, abusive words, etc.

R4: Hidden Behavior. This skill can also perform different hidden actions such as unauthorized resource sharing, gaining more resources than needed, etc. In Listing 7.2, I demonstrate how to share the gained resource and even gain more user data than needed. In detail, MapNav acquires the access tokens for Amazon and Google after interacting with the Alexa platform. Then it obtains the user data using these tokens. At the same time, the tokens can be sent out to other parties without any restriction. Also, this skill can not only acquire the location information on Google account profile but also unnecessary information such as a user's full name and email address. To do that, MapNav simply defines more fields in the request. I find that Google People API by default allows the requester to gain profile information that is not requested. Moreover, other conditional code executions such as behavior changes based on execution time are also feasible [3].

```
1 ...
2 const userFields = 'birthdays,addresses' ;
3 const APIkey = 'AIzaSyBaZjRAlgSb4B0FFYbQxxxxxxxx';
4 const permissions = ['read::alexa:device:all:address'];
5 const consentToken = requestEnvelope.context.System.user.permissions
6 //get access token from Amazon for platform permission
7 && requestEnvelope.context.System.user.permissions.consentToken;
```

```
8  const deviceAddressServiceClient =
9  serviceClientFactory.getDeviceAddressServiceClient();
10 // get device address from Amazon
11 const address = await deviceAddressServiceClient.getFullAddress(deviceId);
12 var accessToken = handlerInput.requestEnvelope.context.System.user.
       accessToken;
13 //get access token for Google from Amazon Alexa
14 const urlUser =  'https://www.googleapis.com/oauth2/v1/userinfo?alt=json&
       access_token='+accessToken ;
15 // getting user's Google user ID
16 const userInfo = await (async () => {
17 const fetchResult = await fetch(urlUser);
18 const data = await fetchResult.json();
19 console.log(data);
20 return data;
21 const urlPeople = 'https://content-people.googleapis.com/v1/people/'+
       userOpenID+'?personFields='+userFields+'&key='+APIkey+'&access_token='+
        accessToken;
22 const people = await (async () => {
23 const fetchPeopleRes = await fetch(urlPeople);
24 const peopleJson = await fetchPeopleRes.json();
25 console.log(peopleJson);
26 return peopleJson; //get data from Google
```

Listing 7.2: Skill Source Code: Access users' resource in their Google accounts

7.2.2.2 Invisible VUI

R5: Run-time Information Collection. The sensitive information collection at skill run-time can happen in any skills [2, 3]. A skill may ask questions such as "What is your name?" or "What is your age". Similar to the results reported by existing work [2], I find many skills do ask these questions and no skill security indicators are designed to warn users against such unwanted information sharing. For example, when discussing this research with Amazon, I reported a skill

named "Symptom Checker" (developed by “Infermedica”) to have unjustified information collection. While this skill requested no permission or account linking, it would ask for unnecessary and unclaimed user information such as users’ full names. Although this skill was soon removed from the Alexa store, it shows that there is still no effective protection mechanisms exist to protect users from the run-time information collection threat.

7.2.3 Indicator Design

Next, I describe the indicator design and what are the conveyed risks. I show that current skill security indicators are only designed to convey risks: R1, R2, and R3. While the rest of this chapter focuses on these 3 types of risks, I also discuss (Section 7.5) potential strategies to alert users to R4 and R5.

7.2.3.1 Skill Permissions

User resources associated with their Alexa accounts (e.g., device address, Amazon Pay) can be acquired via skill permissions. If one or more skill permissions are declared in a skill, two skill security indicators are shown to users. First, a *passive warning* (① in Figure 7.3) lists the skill permission names on the homepage. Second, a permission prompt window with all requested permissions (shown in Figure 7.2) is used to request users’ consent. For each permission, a description is provided to help users understand the risks of consenting to the request. For example, the permission prompt in Figure 7.2 (in Section 7.1) explains what is `Alexa Notification` and `Alexa Reminder`. A list of detailed descriptions provided by Amazon Alexa is provided in Table 7.2. Users can toggle the permission buttons to select what permissions to grant [139].

Similar to mobile permission systems [49], Alexa leverages skill permission prompts [139], together with the passive warning, to alert users regarding potential privacy- or security-invasive skills. Hence, skill permissions are designed to cover both R1 and R2.

Table 7.2: The descriptions provided in permission prompt.

Skill Permission	Description shown in the permission prompt
`Device Address`	Allow this skill to access the full postal address configured for your Alexa devices
`Country and Zipcode`	Allow this skill to access the country and postal code configured for your Alexa devices.
`Alexa Notification`	Allow this skill to send you notifications, which you can retrieve by asking, "Alexa, what are my notifications?"
`Read List`	Access to your Alexa lists
`Write List`	Permission to modify information on your Alexa lists
`Email`	Allow this skill to access the email address associated with your account
`Full Name`	Allow this skill to access the full name associated with your account
`Alexa Reminder`	Allow this skill to create and edit reminders. This skill can only see the reminders it creates
`First Name`	Allow this skill to access the first name associated with your account
`Amazon Pay`	Allow this skill to use Amazon Pay to make your payments. Amazon Pay will share your name, email and shipping address, but not your financial information, with the skill developer
`Mobile Number`	Allow this skill to access the mobile number associated with your account
`Location Service`	Allow this skill to access your location while the skill is in use

7.2.3.2 Account Linking

User resources managed by non-Alexa entities[1] can be accessed by account linking. For example, a skill can be linked to a user's Google account to access the associated user data, such as contact information. There are two skill security indicators designed for account linking. First, a passive warning (i.e., an "account linking" icon shown as ② in Figure 7.3) will be shown on the skill homepage if the skill developer declares account linking. However, similar to the passive warning for skill permissions, no detailed information regarding the risks of using this function is displayed. Second, after the users clicked the "Enable" button, an account linking process will/-can be triggered to redirect the users to an external log-in page. [2], and users will be redirected to a log-in page for configuring the account linking. During this process, a log-in page warning (recommended by Amazon [129]) should exist to inform users of how the third-party skill access users' resources.

According to the Alexa developer documentation, " 'link accounts' means 'to get the user's

[1]Amazon is considered to be a separate entity. In other words, Amazon and Alexa use different accounts. For example, as shown in Fig. 7.6, a skill still need to gain access to resources associated with users' Amazon account.

[2]Amazon Alexa allows a skill to conduct automatic redirect log-in page redirecting. In this case, the passive warning is shown as "Account Linking Required" [140]. However, there is also a passive mode, "Account Linking Available", in which the account linking redirecting will only be triggered if users click the "Link Account" button after enabling the skill. With no security implication mentioned in the Alexa documentation, I do not further discuss their difference in this chapter.

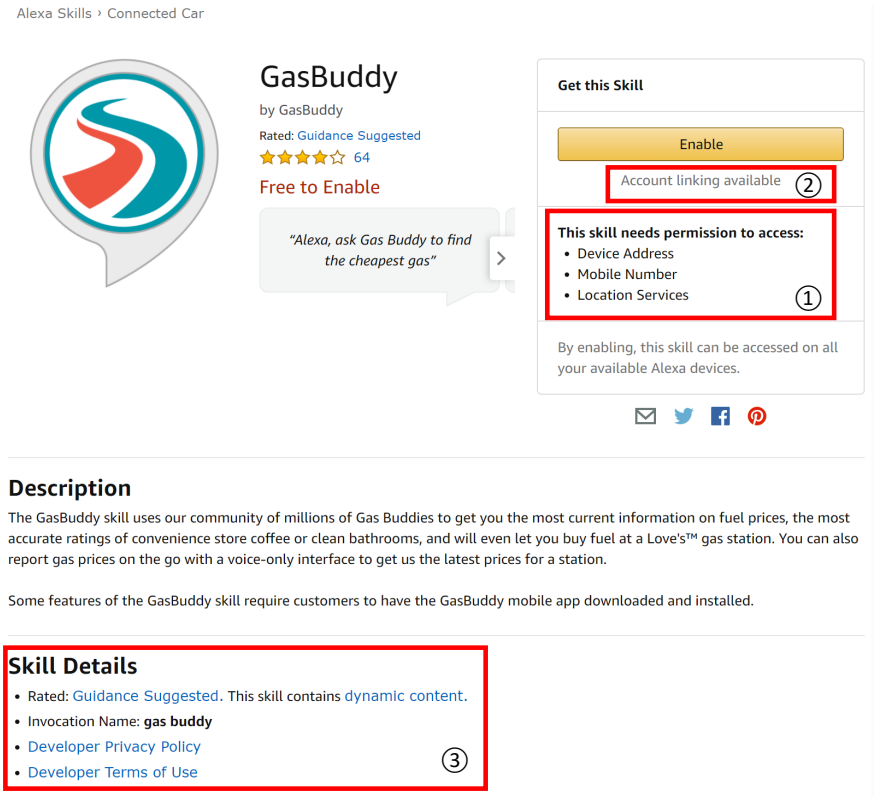


Figure 7.3: Skill homepage for "GasBuddy" skill.

permission to obtain ... the user data' " [141]. The underlying risk (i.e., R1 and R2) is that the third-party skill would gain access to users' sensitive resources associated with the linked account. To warn users against potentially invasive third-party skills, Amazon recommends that any implemented account linking process should let the users "view and accept any terms and conditions".

7.2.3.3 Skill I/O

Skills can also acquire user information through skill I/O [2]. When interacting with a user, skills can output a question and collect the information with the speech input. For example, a skill named Wiffy [142], which is designed to be a Wi-Fi password management skill, asks for users' phone numbers. Security indicators related to skill I/O are limited to a passive warning (i.e., the "Dynamic Content" shown in ③ in Figure 7.3). Specifically, in this warning (Figure 7.4), potential dynamic content is mentioned to warn users of the risks while interacting with third-party skills.

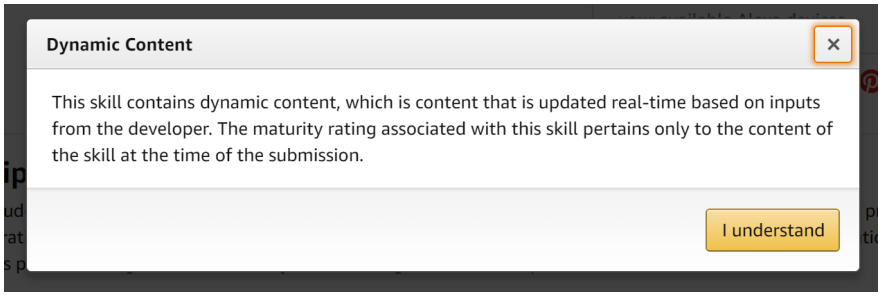


Figure 7.4: Passive warning for skill I/O.

This passive warning is clickable, but no prompt window or other proactive security indicator design is used. Also, this dynamic content warning exists in all skill homepages.

As shown in Figure 7.4, a skill's output can be dynamic (i.e., R3). Also, Amazon does not guarantee the effectiveness of the certification process provided by the Alexa market. It is admitted that a skill can change their behavior at any time. For example, a trivia skill [143] may change its output questions every time it is triggered. However, the change in skill output can be policy-violating or even malicious. For example, an originally normal question may contain inappropriate sexual comments [3] at a later time. Also, a neutral question could become a question aiming to collect private information.

7.3 User Survey

The user survey aims to quantitatively test the skill security indicators' effectiveness with the aforementioned risks considered. I start presenting the user survey by introducing the applied methodology. Next, I describe user study details and the findings.

7.3.1 Methodology and Recruitment

When designing the user survey, I adhered to two design principles. First, following the best practice in studying security indicators [49], both user survey and skill experiment were used to correlate and justify the study results. Second, I leveraged C-HIP model [47, 144] as a guideline to design user studies for both studies. Specifically, three key steps in the C-HIP model is used in this phase:

- Attention: Do users pay attention to a skill security indicator? A user needs to switch focus from the primary task (i.e., installing a skill) to the security indicator, and she needs to focus on the security indicator for long enough to read and evaluate them.

- Comprehension: Do users understand the risk of granting permissions (for allowing resource access via skill permissions, account linking, or skill I/O) to third-party skills? Users need to understand the scope and implications of the permission.

- Behavior: Do skill security indicators influence users' installation decisions? Do users ever cancel installation because of the security indicators? Users should not install skills whose permissions exceed their comfort thresholds.

I did the recruitment and payment distribution of the user survey using Amazon Mechanical Turk (MTurk)[3]. I used different MTurk recruitment filters to ensure the quality of the data collection. For example, I recruited MTurk workers who had high job approval rates (greater than 90%) and a sufficient number of approved jobs (greater than 500). I used MTurk as my recruitment platform because research has shown that MTurk workers are more attentive than subject pool participants [145]. With Mturk's large pool of diverse workers, for academic research, it was by far the dominant platform and has been extensively validated [146].

I recruited 150 respondents and paid $10 for each finished task. To ensure the quality of such online data collection, two attention check questions (e.g., "Can you open the survey link properly?") in the survey to help filter out bots and inattentive respondents. I also filtered out respondents who were inconsistent in answering the questions. For example, I asked a question at the beginning of the survey: if you own a smartphone? I do not include the answers from those who provided a negative answer. This was because I only recruit respondents who have used Alexa, and one needs a mobile app to initiate an Echo device or use the Alexa mobile application. Alexa users may use different portals. I asked the respondents to select their primary way of exploring

[3]This study's data collection was done between May 2020 to September 2020. The study was approved by the university Institutional Review Board (IRB). No personal identifiable information was collected or stored. To comply with the university COVID-19 guideline, no in-person user experiment was conducted.

and installing skills (either mobile, web, or voice command). I do not consider the results from those who only use voice commands to install skills. This was because skills with account linking or skill permissions cannot be installed via voice commands alone. As a result, 137 respondents' answers were collected. I assigned 18 minutes for a user to finish the survey. The average finish time was 14 minutes 17 seconds.

Among the 137 respondents, 65 were male, and 72 were female, with the remainder declining to identify their gender. Most of the respondents' age was between 29 and 39 (42%). Others' age distribution was: 22% between the ages of 18 and 28, 26% between the ages of 40 and 50, 8% between the ages of 51 and 61, and 2% over the age of 62. Most of the participants were frequent Alexa users (54%). Moreover, 58% of the participants used web Alexa store as their primary Alexa portal. More than 65% of them own a bachelor's degree or higher.

7.3.2 Survey Questions

The survey questions are designed to focus on collecting users' perceptions of the skill security indicators based on their experience of using Alexa. In this study, I did not require the respondents to interact with Alexa. The user survey has 30 questions. The first part of it consists of 9 general questions asking about respondents' backgrounds, including their gender, education history, and familiarity with Alexa. The second part of the user survey consists of 21 questions regarding user experience related to skill security indicators. Six of the questions were randomly-picked skill permission quiz questions (see the full question list and results in Table 7.3). These quiz questions were designed to study users' comprehension of skill permissions. To do that, I showed the respondents a skill permission prompt (with one permission request). Among the options, other than "I don't know" and "None of these", one or two correct options were provided. The correct options were reasonable inferences of the capabilities implied by the skill permissions.

Table 7.3: User comprehension quiz questions for skill permissions.

Question	n	Options	Response
`Device Address`	69	✗ Amazon will send me advertisements to my physical address. ✓ The skill will know my full address associated with the device. ✗ The skill will know how I go to work. ✗ None of these. I don't know.	22 (31.9%) 41 (59.4%) 4 (5.8%) 0 (0%) 2 (2.9%)
`Country and Zipcode`	68	✗ The skill can locate my phone location. ✓ The skill will know my zipcode associated with the device. ✗ My home address. ✗ None of these. I don't know.	13 (19%) 52 (76.4%) 2 (3%) 1 (1.5%) 0 (0%)
`First Name`	69	✗ The skill will ask me whether I want to play a voice message. ✓ The skill can read the first name configured for my Alexa account. ✗ The skill may call me. ✗ None of these. I don't know.	2 (2.9%) 48 (69.5%) 16 (23.2%) 3 (4.3%) 0 (0%)
`Full Name`	68	✗ Amazon can make purchases using my name. ✗ Amazon will know the full name of the skill I am using. ✓ The skill can read the full name configured for my Alexa account ✗ None of these. I don't know.	1 (1.5%) 5 (7.4%) 61 (89.7%) 0 (0%) 1 (1.5%)
`Mobile Number`	69	✗ The skill will send me a email. ✓ The skill will know my phone number configured for my Alexa account ✗ The skill may access my phone calling history. ✗ None of these. I don't know.	13 (18.9%) 47 (68%) 8 (11.5%) 1 (1.5%) 0 (0%)
`Location Service`	68	✗ The skill will ask me whether I want to play a voice message. ✓ A skill can know my dynamic location information. ✗ Allow a skill to access my device address. ✗ None of these. I don't know.	6 (8.8%) 47 (68.1%) 11 (16.1%) 3 (4.4%) 2 (2.9%)
`Email`	69	✗ The skill will know my home address. ✓ A skill can read the email configured for my Alexa account ✓ The skill might send me a email. ✗ None of these. I don't know.	14 (20%) 48 (69.5%) 7 (10%) 0 (0%) 0 (0%)
`Amazon Pay`	68	✓ Amazon Pay will share my name, email and shipping address ✓ Allow a skill to use Amazon Pay to make my payments. ✗ Someone will call me. ✗ None of these. I don't know.	22 (32.3%) 40 (58.8%) 0 (0%) 3 (4.4%) 1 (1.5%)
`Alexa Notification`	68	✗The skill will call me. ✗ Alexa will play a message from the skill without asking me. ✓ The skill will ask me whether I want to play a voice message. ✗ None of these. I don't know.	1 (1.5%) 20 (29.4%) 36 (53%) 7 (10.3%) 3 (4.4%)
`Alexa Reminder`	69	✗ The skill will ask me whether I want to play a voice message. ✓ Alexa will play a message from the skill without asking me. ✗ The skill can read my browser history. ✗ None of these. I don't know.	32 (46.3%) 22 (31.8%) 13 (18.9%) 0 (0%) 2 (2.9%)
`Read List`	68	✗ The skill may play music automatically. ✗ Allow a skill to modify my shopping list. ✓ The skill can read my shopping list. ✗ None of these. I don't know.	20 (29.4%) 16 (23.5%) 25 (36.7%) 5 (7.3%) 2 (2.9%)
`Write List`	69	✗ The skill will notify me about my bills. ✗ Alexa will play a message from the skill without asking me. ✓ The skill may change my shopping list. ✗ None of these. I don't know.	16 (23.2%) 15 (21.7%) 30 (43.4%) 4 (5.8%) 4 (5.8%)

Table 7.4: User survey results for skill permission attention rates.

Attention to Passive Warning		95% CI	Attention to Permission Prompt		95% CI
Looked at permission	39%	23% to 54%	Looked at permission	42.3%	20% to 59.2%
Did not look, but aware	51.7%	28% to 63%	Did not look, but aware	42%	24% to 61%
Checked Description	N/A	N/A	Checked Description	9.2%	7% to 15%
Was unaware of permission*	9.3%	4% to 13.5%	Was unaware of permission	6.5%	3% o 12%

7.3.3 Skill permissions

7.3.3.1 Attention

Do users pay attention to skill permission security indicators? Attention is the prerequisite of other steps in a C-HIP model. It is crucial to understand if Alexa users are paying attention to the skill indications, such as skill permissions in the first place.

First, I asked the respondents if they ever used skills with permissions. For respondents who answered yes, I then asked them, "For the last time when you installed an Alexa skill, which of the following properties of the skill were you aware of before you decided to enable it?" Next, for the respondents who chose the option of "skill permission", I further checked what did they look at? The answer could be either the passive warning or permission prompt. Lastly, if the respondents answered that they looked at the permission prompt, I further questioned them if they checked the permission descriptions (e.g., the descriptions in Figure 7.2).

I find that 13 respondents had never used skills with permissions or account linking. The reason could be that many skills published in the Alexa store only perform simple tasks like question answering. Thus, these skills usually do not request skill permissions or account linking. As shown in Table 7.4, even though a fair amount of Alexa users were aware of or looked at the permissions (either passive warning or permission prompt), they did not further check the detailed permission descriptions. The skill permission attention rates were low because only a few of them (9.2%) checked the details.

Finding 1: *Most respondents did not pay attention to skill permission details.* The results from the user survey show that some respondents did notice the skill permissions. However, only a small

portion of them checked skill permission descriptions at least once when they were installing the skills. As suggested by the C-HIP model, this could prevent users from understanding the risks conveyed in the skill security indicators.

7.3.3.2 Comprehension

Do users understand the risks of granting skill permissions? It is essential to assess how Alexa users perceive the scope and implication of skill permissions.

I asked each respondent 6 randomly-selected permission quiz questions. In each question, both the permission name and descriptions will be displayed. The results indicate that most respondents understood conventional permissions such as `Device Address`. Also, I consider Alexa-specific skill permissions which are capabilities directly related to VUI functions such as Alexa list read/write, Alexa audio playing (e.g., `Alexa Reminders`). The results show that the comprehension rates for Alexa-specific permissions were low. For instance, `Alexa Reminders` has a comprehension rate of 31.8%. Also, users were confused at the difference between `Alexa Reminders` and `Alexa Notification`. For example, for `Alexa Reminders` quiz question, 46.3% of the respondents chose the wrong option, which is the correct answer for `Alexa Notification`.

Also, for conventional permission quiz questions, there was no statistical correlation between age and the number of correct answers. However, for Alexa-specific permissions, there was a negative correlation ($r = -0.557$, $p < 0.001$); younger people were more likely to understand Alexa capabilities.

Finding 2: *The respondents did not have a good understanding of Alexa-specific skill permissions.* Comparing to the conventional permissions, much fewer Alexa users understand Alexa-specific skill permissions (75.5% vs. 44%, for overall correctness). Users are more familiar with these conventional capabilities, which are often used in other popular platforms such as mobile phones. However, I am not able to further check the statistical significance for respondent groups who owned or not owned a phone. This is I already excluded respondents who reported to have not used mobile applications. As an alternative way, I explore such behavior in the skill experiment

(in Section 7.4).

Another question I asked was, "Who requested and used the skill permission(s)?". This question was used to understand whether the respondents realized who could be causing the risks. Surprisingly, a large portion of Alexa users did not understand how third parties were involved in the Alexa ecosystem. Most respondents (71%) in the user survey chose that it is Amazon or Alexa who requested the skill permissions. No respondent answered the correct answer, which is "the third-party skill".

Finding 3: *Many of the respondents thought that it is Amazon who requested the skill permission.* Some users did not consider the permissions seriously because they thought Amazon created the skills so that these skills can be trusted. This incurs a confused deputy problem. For example, if an evil skill can leverage users' misplaced trust to acquire private information or other resources. Specifically, because user-owned resources are often protected well by either Amazon or other platforms, it could be difficult for an attacker to gain these resources without becoming a skill developer. However, once the users' trust in Amazon is falsely transferred to a skill controlled by a third party, the third-party could easily access users' resources.

7.3.4 Account Linking

7.3.4.1 Attention

Do users pay attention to the security indicators for account linking? There are two skill security indicators used in the account linking process: the passive warning before installing and the log-in page warning during the account linking process. I aim to examine if the respondents ever checked these two indicators.

I first asked the respondents, "Have you ever used any skills with account linking?" I exclude respondents who never used account linking in skills. As a result, 107 valid results were collected. Next, I raised the question, "Did you notice any warning messages when using the account linking?" I find that most of the respondents (81.3%) were aware of warnings regarding account linking in general. Fewer respondents (64.4%) specifically noticed the log-in page warning[4]. I used

[4]I provided a note to help respondents understand what log-in page warnings are.

the skill experiment illustrated below to understand further why respondents paid more attention to account linking than skill permissions.

7.3.4.2 Comprehension

Do users understand the risks of the account linking? I asked respondents' understanding of account linking.

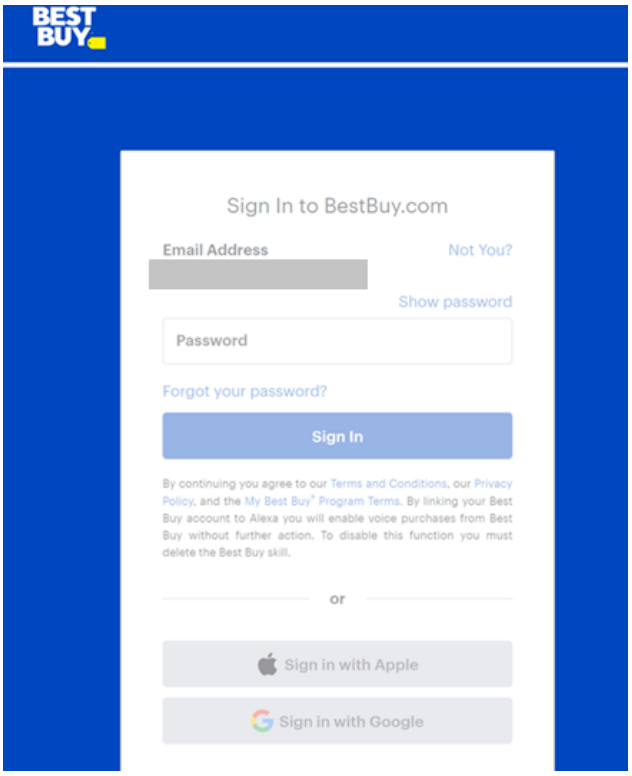


Figure 7.5: Example of automatic login during account linking.

In the survey, for the 107 respondents who have used skills with account linking, I asked them regarding the passive warning of account linking, "what does 'account linking available' mean?" The result shows that 72% of the web-based Alexa respondents and 62% of the mobile-based respondents understood the conveyed risk of proceeding with the skill with account linking. I also showed the respondents three different log-in page warning examples (i.e., screenshots from Amazon, Google, and Best Buy account linking). There were 88.9% (32/36), and 72.7% (24/33) of the

respondents who perceived the warning messages from Amazon and Google correctly. However, there were only 29% (11/38) respondents who checked Best Buy account linking warning (shown in Figure 7.7) understood it correctly. Another question I asked was about who requested account linking. There are 100 out of 107 respondents (93.5%) who answered either Amazon or Alexa. I further studied the reasons behind the issue in the skill experiment.

Finding 4: *The respondents understood the log-in page warnings if it is well presented.* I find that most of the respondents understood the warnings provided in the account linking processes. This indicates that, with proper indicator design, it is possible to inform the users regarding potential risks in using the skill account linking. However, I find that the problem of recognizing who the requester was (for skill permissions) also existed in the account linking case.

7.3.5 Skill I/O

7.3.5.1 Attention and Comprehension

Do users pay attention to and understand the security indicator for skill I/O? As discussed in Section 7.2.3, only a passive warning is associated with skill I/O. Moreover, the passive warning was not shown to the user directly. Instead, a user has to click the "Dynamic Content" link in the Skill Detail section of the skill homepage (shown as ③ in Figure 7.3) to see skill I/O related warning message. In both the user survey and skill experiment, I checked if any respondent ever clicked and checked the passive warning. The result shows that no respondents noticed or clicked it. I further want to check that assuming a respondent looked at the warning message, will she understand the risks conveyed? Thus, I presented them the"Dynamic Content" message and asked their understanding of that. As a result, 21 out of 124 respondents of the user survey and 4 out of 28 respondents from the skill experiment perceived it correctly.

Finding 5: *Alexa users did not check or understand skill I/O security indicators.* I find that the respondents are unfamiliar with skill I/O in terms of its potential risks or how it works. While VUI is new to any users, more effective and educational indicators should be provided to help users understand the risks in such a new user interface.

7.3.6 Behavior

I asked the respondents if they have ever decided not to install a skill? Only 6 respondents answered that they had declined the skill installation process. Four respondents claimed that they did not remember if they ever declined to install a skill. The others (114/124) provided negative answers. Next, for respondents who had ever decided not to install a skill, I asked them the reasons. The result shows that only 3 respondents were influenced by skill permission. No one declined to install or use a skill due to the security indicator of account linking. The potential reason could be that there are no actionable recommendations provided in the skill security indicator (i.e., the icon "Account Linking Available"). Thus, the respondents might not know what they are supposed to do even if they think the account linking can be dangerous. Also, no respondent reported that she was affected by the skill I/O security indicators.

7.4 Skill Experiment

In the user survey, I find that risks regarding VUI are not well presented to users. To further verify these findings and explore users' mental models behind these findings, I recruited 45 respondents to conduct a skill experiment.

7.4.1 Experiment Design

Different from the user survey's questionnaire design, the skill experiment asked the respondents to install real-world Alexa skills and answer interview questions. There are two reasons for conducting the skill experiment. First, by asking users to use the skills, I collect user data based on their fresh memory. This can be useful to confirm and explore the findings from the user survey. Second, the skill experiments were designed to gather nuanced data. With a longer experiment time assigned, I design various open-ended questions to explore the mental models behind users' behaviors.

The first part is general questions including education backgrounds, gender, etc. Second, I asked the respondents to use three different Alexa skills. After using each skill, they were asked to answer interview questions regarding their experience in using the skills. Specifically, the first

skill (created by us) is used as a check-in skill to validate if the respondent can use Alexa properly and help the respondent to warm up. The next two skills in each skill experiment were picked from a pool of 9 skills covering different observed skill security indicator use cases. As a result, there are 30 respondents assigned for each set of skill security indicators. I excluded 4 respondents who did not pass the check-in skill. This results in 28 valid answers for each set of the indicators.

In the skill experiment, respondents are instructed to stop using the skills once they are installed (except for the first check-in skill) to prevent unwanted information sharing. For each skill installation instruction, I designed a three-step procedure to mimic the real-world usage of an Alexa skill:

S.1 Participants will be asked to use a skill with the given real-world scenario. For the mobile Alexa users, I asked the respondents to search the skill on their own. Also, *account credentials* are provided if an account linking is needed. I illustrate one example instruction of using the Strava in Alexa:

Pretend that you are a frequent runner and you want to use Alexa to access your Strava account. You use the following skill after searching for different news skills. Please click the link using your phone or computer. www.amazon.com/Running-history-check-Strava-unofficial/dp/B06XG4Z1N6/

S.2 Now please try to use it. Note I do not evaluate how you use the Alexa skill. The following interview questions were designed to collect your feedback for using Alexa skills.

S.3 After you used the skill for the first time, please 1) disable the skill, 2) close the skill homepage, and 3) finish the following survey questions. Please feel free to stop using it if you decide not to proceed with the skill.

7.4.2 Results

Skill Permission Attention. To validate Finding 1, I asked them the same questions as the user survey to check what they looked at when installing the skill. I collected 28 valid results from

the skill experiment[5], and 17 (60.7%) of them claimed (at least for one time) that they noticed the skill permission. Five of them checked the detailed permission warnings. For those who did not check the detailed descriptions, I asked for the reasons. I find that many respondents tend to get rid of permission prompts as soon as possible. For example, one respondent's feedback is listed as follows,

I just keep clicking next and use the skill. The skill should be safe I think.(P-02, G2[6])

The result consolidates Finding 1. Also, the mental model of ignoring the skill permission prompts explains why many users were not checking skill permission details. However, it is still unclear about the reasons behind this mental model. Hence, I asked the respondents to answer a skill permission quiz with (similar to the user survey) 3 Alexa-specific and 3 conventional skill permissions. There are 21 (75%) respondents who answered all three conventional permission questions correctly. Among these 21 respondents, 9 of them reported that they never checked the skill permission details. I then asked them why they still knew the answer. I find that most of them (8/9) said the knowledge was inherited from their experience of using mobile applications. Thus, I conclude that there exists cognitive inertia that some Alexa users tend to only use what they know and resist changes.

Alexa-specific Permissions. In the quiz mentioned above, I find that only 1 (3.6%) of the respondents answered all three Alexa-specific permission correctly. I also asked the respondents the difference between `Alexa Reminders` and `Alexa Notification`. This question is an open-ended question to prevent any inference and collect users' perceptions on these two skill permissions. After manually checking the answers, I find that most of the respondents (24/28) did not answer them correctly. Note there are 4 of these 24 respondents failed to provide a meaningful answer (blank or one-word answer). These results approve Finding 2.

Who uses my data? For respondents who used skill with skill permissions, I asked them who requested the skill permission; the result confirms Finding 3 that most users (20/28) did not realize

[5]The excluded results are from respondents who could not pass the first check-in skill.

[6]G1 is the participant group for user survey, and G2 is the participant group for skill experiment.

that skills are not managed or owned by Amazon/Alexa. For respondents who experienced the account linking process, 9 out of 28 respondents (32.1%) answered that it was the third-party developer/skill who gained access to their resources. These results are consistent with Finding 3.

For respondents who think it is Amazon or Alexa who requested and used the permission, I further asked them why they thought so. As a result, I find many of them thought that the Alexa and the skills are conceptually one entity. For example, one of the respondents answered:

> *I am talking to the Echo device and it is sold by Amazon. Also, I enabled the skill on Amazon website. Then it should be Amazon who did that. (P-033, G2)*

Log-in Page Warnings. I first asked the respondent if they noticed any security warnings during the account linking process. The result shows that 22 (out of 28) of the respondents were aware of the passive warning. Also, there were 17 respondents noticed the log-in page warning and 13 of these 17 respondents claimed they checked the details of the log-in page warnings. There were 2 out 10 respondents who tested "Running history check for Strava" reported that they were aware of log-in page warnings. I find that, compared with respondents who used the account linking process provided by Strava, the respondents who used skills with account linking to Amazon or Twitter performed significantly better in checking log-in page warnings (20% vs 83.3%; Mann-Whitney U Test, $U=33$, $p< 0.05$). I checked the log-in page warning design of Strava and noticed that Twitter and Amazon's account linking processes are better presented with proper scope declaration and consent windows. I discuss more details in Section 7.5.

Skill I/O. After the respondents used skills without account linking and skill permissions, I asked them if they decided not to install the skill? As a result, only two out of 28 respondents have declined the skill installation. Both of them were assigned to test "Wiffy" skill.

However, I find that they become cautious not because of the skill security indicators but other factors. For example, one respondent who declined to use Wiffy skill said that,

> *I don't feel comfortable giving out my Wi-Fi information to an app/company that I know little about. Looking at the developer privacy policy, there isn't a concrete description of how the*

Wi-Fi password was stored, and if they can or cannot share it with others. (P-29, G2)

I chose "Wiffy" based on the example policy-violating skill mentioned in SkillExplorer [2]. Guo et al. [2] showed that type of skill could be dangerous as it would collect users' private information covertly with skill I/O. The aforementioned result shows that the skill I/O security indicator failed to alert users regarding potential risks.

7.5 Discussion

I first discuss about potential design issues of skills security indicators. Then, I provide short-term recommendations to help improve the skill security indicator design. Then, I illustrate several open problems regarding the skill permission system and skill security based on my findings. Lastly, I discuss the limitations and ethical considerations for this research.

7.5.1 Design Issues

Skill Permissions. The first problem is the prompt window's permission-manager design [147]. This approach ensures a smooth user experience; however, it is less effective as a security indicator to convey R1 and R2 (i.e., requested scope and consequences & third-party identity). For example, unlike mobile's accept/reject option for each requested resource, skill permissions are preset to be true (shown in Figure 7.2), and users would click "save permission" to proceed with the skill. The problem is that users may not pay attention to these remindful skill permission prompts and take it for granted that these skill permission requests should be safe to consent. The second problem is that, as shown in Table 7.2, the descriptions provided by permission prompts could be vague to users. As a result, a user might proceed with the skill without fully understanding the scope and security implications of granting skill permissions. For example, the description for `Alexa Reminder` capability [148] does not depict that a third-party skill can play audio messages automatically. This capability can potentially become invasive with the extensive control of when and how to play audio to users' private space[7]. Another example is that the Alexa list mentioned in `Read List` and `Write List` is unexplained. Users could underestimate the

[7]The skill's backend code can be modified freely by the developers [3]

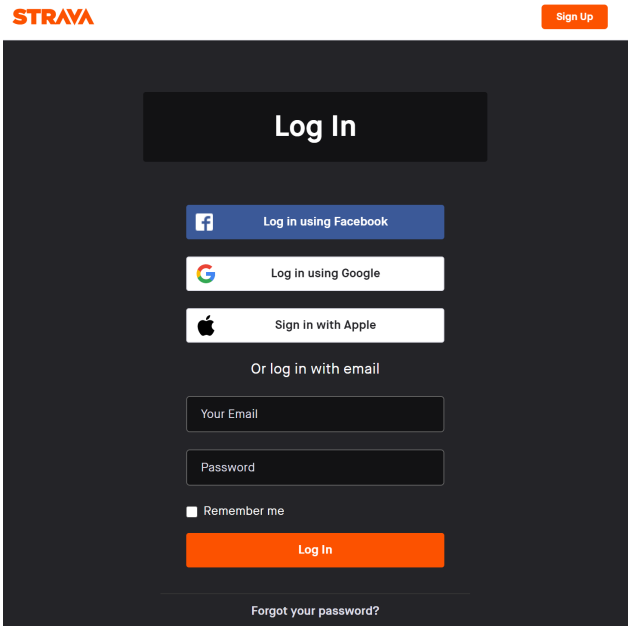


Figure 7.6: Account linking to Strava.

risks because skills can access both Shopping and To-Do Lists associated with users' Amazon accounts [149] using these two skill permissions.

Account Linking. Skills' heterogeneous linking processes are implemented by different third-party authorization servers. This heterogeneity results in the question of whether log-in page warnings would always be effective in alerting users to R1 and R2. For example, as shown in Figure 7.6, no scope declaration or consent process is implemented by Strava. The missing log-in page warnings make it unclear what resources would be given away to the third-party skill. As a result, users' private information associated with their Strava account, such as home address, GPS location, daily activities, could be exposed unexpectedly. This is not an isolated example because I find some other companies also did not have a well-implemented log-in page warning [150]. As another example, Best Buy (Shown in Figure 7.7) integrates the warning messages with its regular log-in notice, and no subsequent consent process is implemented. It could be difficult for users to notice such a security indicator. With limited account linking related indicator provided on skill home pages (i.e., the passive warning mentioned above), the responsibility of warning users against potential risks of link accounts to third-party skills solely relies on third-party authorization

servers [129].

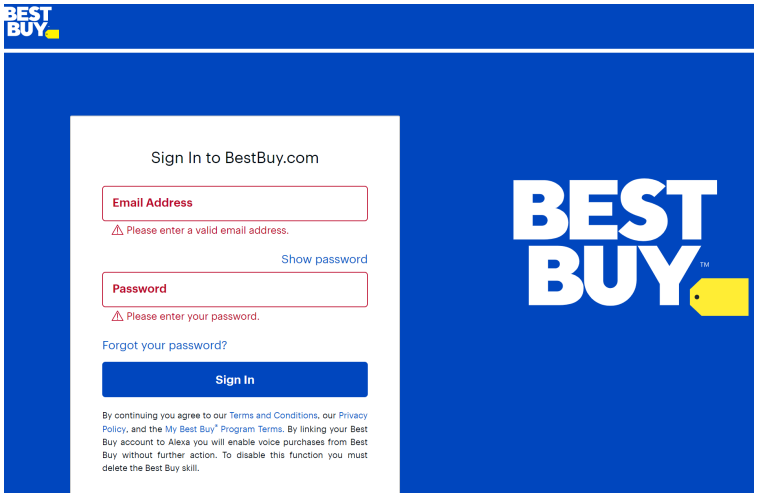


Figure 7.7: Best Buy account linking.

Skill I/O. The problem is that passive warnings, such as the click-to-open warning design used in this case, usually have very low attention rates [50, 131]. As a result, users may ignore this skill security indicator (shown in Figure 7.4) which is related to R3. Moreover, compared with visual interfaces, VUI is relatively new to users. Thus, if this indicator cannot convey all the risks properly (i.e., R3, R4, and R5 mentioned in Section 7.2.3), users may not notice any potential dangers when interacting with skills. For example, the aforementioned Wiffy skill has a potentially policy-violating question to collect users' phone numbers. A user might not be aware of the potential problems of collecting their phone number (R5: run-time information collection) and potential unwanted information sharing (R4: hidden behavior at back-end). These problems lead to a potential ineffective skill security indicator design for skill I/O.

7.5.2 Short-term Recommendations

7.5.2.1 Attention

Based on our findings related to user attention, I provide the following recommendations.

Skill Permission prompt. The permission prompt should not be designed like a permission manager [147]. The current design implies a false impression that the skills have been installed and users are asked to remove any preset permissions if they want. I notice that users tend to skip the prompt window as soon as they can by clicking the "Save Permission" buttons. First, the skill permission should not be preset to be true. Second, the prompt window design should encourage the users to check the skill permission descriptions carefully. For example, similar to the mobile permission prompt design, the skill permissions' prompt window should ask users to choose either "Allow" or "Reject" for each permission. This can potentially increase the users' willingness of checking permission details.

Log-in page warning. Alexa relies on the third-parties to implement the log-in page warnings. However, I find that not all third-party authorization servers provide well-defined and easy-to-read log-in page warnings. I first recommend that the Amazon Alexa store enforce a better log-in page design by including a stricter certification process for the account linking process. It should use a prompt window for each permission requested. Moreover, the scope should be clearly defined as well. Second, the passive warning of account linking should be more warning-like with detailed risk information included. Third, I recommend that the scope of account linking should be added to the skill permission prompt window. This is a feasible approach because the scope information could be acquired easily with the OAuth protocol. The benefit is that users can then check the warnings for account linking consistently rather than viewing them in different formats implemented by different third parties.

Skill I/O. It is not sufficient to only use a passive warning to alert the users to the risks of skill I/O. How a skill could leverage skill I/O should be clearly described in a more obvious manner. First, skills' I/O-related warning can be considered as a type of skill permission to be displayed in the permission prompt window. This would help increase the attention rate and help users understand what the risks could be. Another way to alert the users is to play audio warnings when users are interacting with the skills. Unfortunately, this is a challenging task because it will significantly decrease usability by reducing the effective interaction time. Audio-based warning messages could

still be useful. For example, Alexa may use a different voice or accent when playing speech from a third-party skill. I also believe that post-usage warnings can be helpful. For example, in the current iOS permission system [151], a post-usage warning for background resource usage will warn users of how many times an app (such as Google Maps) has used the user's resource (e.g., location) in the background. Alexa may learn the lesson of leveraging post-usage warnings to help users understand implicit permission usage.

Skill identity and reviews. When analyzing the user study result, I find that users often paid more attention to the other elements (instead of the skill security indicators) such as skill names and user reviews. I recommend that more indicator-related designs for these elements should be added to the skill homepages. For example, Twitter uses a blue verified badge [152] besides the account public profile name to let users know that "an account of public interest is authentic." Also, the platform could highlight useful reviews to help users understand the skills' problems. This would potentially help users pay more attention to third-party skills' security- and privacy-related issues.

7.5.2.2 Comprehension and Behavior.

I also present two recommendations to help improve user comprehension and motivate Alexa users to carefully consider skill security indicators.

Educational UI. Many users have limited knowledge about Alexa-specific permissions. An educational user interface design should be applied in the Alexa installation process. In the current permission prompt, only vague descriptions are provided for Alexa-specific permissions, such as `Alexa Reminders`'s unclear definition shown in Table 7.2. It is not sufficient to help the user understand what could be the potential risks. First, any deceptions shown in skill security indicators should be accurate and easy to understand. Second, more contextual information (e.g., how third parties would use users' data) can be included in the skill homepage.

Actionable recommendation. I find that no recommended actions were provided in any of the skill security indicators. It would be helpful if the skill security indicators can recommend the users reject a permission request if they do not want a third party to access it. By adding more

actionable recommendations, users could be more motivated to consider permissions and take actions accordingly. This is also used by many other security indicators. For example, Android permission would display a recommended action at the bottom of the screen when a permission request is presented.

7.5.3 Limitations

I used a remote semi-moderated design in the skill experiment. In the future, to explore more about the usable skill security indicators, another way could be applying a moderated physical experiment. This could potentially help collect more details regarding users' thinking process. However, I argue that the methodology used in this research is also working well. By studying the methodologies from previous work [134, 130], I used various ways to ensure the user study is close to a real-world one. For example, I first recruited people who own voice assistant devices. Second, I used many screenshots and detailed scenario descriptions to mimic how users used Alexa skills in the real world.

7.5.4 Ethics and Safety.

My research was approved by the university IRB. Both the user studies and developed skills did not collect any personal identifiable information or any other sensitive user data. Moreover, I followed university guidelines over campus safety during the COVID-19 period, and no in-person experiment is conducted.

7.6 Conclusion

In this chapter, I present the effectiveness study of security indicators in the popular Amazon Alexa platform. To understand how users perceive the skill security indicators, I performed two user studies: a user survey and a skill experiment. My findings show that Alexa users pay little to no attention to some critical skill security indicators and often misinterpret the risks associated with skill's permissions. Additionally, many users have misplaced trust in third-party skills wherein they may expect all skills to be safe because they are backed or owned by Amazon/Alexa. As this can lead to users falling victim to undesirable or even malicious skills, I discussed recommended

changes to skill security indicators to improve their effectiveness. I hope that my findings can stimulate future research efforts in this emerging direction.

8. SUMMARY AND LESSONS LEARNED

In this dissertation, I introduce three systems, i.e., AEOLUS, one pitch attack, and LipFuzzer, and a user study targeting skill users. In this chapter, I first review these works and summarize the research contributions in three aspects, research questions, research challenges, and research outcome. Next, I discuss lessons learned from these studies to further illustrate the insights I gained from this research.

8.1 Summary

In this section, as shown in Table 8.1, I describe how the studies in this dissertation are interconnected and what unique contributions they intended to make.

8.1.1 Research Questions

My overall research question for this dissertation is how to understand and secure voice assistant applications systematically? Specifically, four phases of studies are conducted to address this question. These phases are different from each other mainly because they are targeting four different steps in the skill pipeline. Hence, when designing their detailed research questions, I follow a similar thinking process and aim to solve the unique problems in different skill pipeline steps. I describe more details in the following:

- I find that the first step, the acoustic channel, in VA systems are not well protected. For example, re-use and replay attacks still widely exist. This is a non-trivial problem because the acoustic channel is the first step of the skill pipeline, and users interact with VA or skills through this potentially dangerous channel. To start tackling this problem, in Phase 1 of this dissertation, I aim to solve the speech re-use issues and secure the VA speech input captured from the open acoustic channel. Specifically, although many research works have attempted to solve the speech re-use problem, VA's acoustic channel remains vulnerable due to the limited performance of these prior works. The research question is how to tackle the speech re-use issue without limited working distance as well as data and device dependencies.

Table 8.1: Dissertation Summary

	Phase 1	Phase 2	Phase 3	Phase 4
Skill pipeline	Acoustic channel	Speech processing	Intent extraction	Skill processing
Target problem	Speech re-use	Adversarial attacks	Misinterpretation	Malicious skills
New Attack	✗	✓	✗	✗
Defense Strategies	✓	✗	✓	✓
System/Tool	✓	✓	✓	✗
User Study	✗	✗	✗	✓
Usability	✓	✓	✓	✓
Imperceptibility	✓	✓	✗	✗

- The speech processing is the next step after the acoustic channel. Different from the previous step's focus on audio capturing and playback, different signal processing and machine learning techniques are involved in this step. One of the important research areas for the speech processing step is to attack the speech recognition and speaker recognition systems. In Phase 2, I focus on improving the state-of-the-art adversarial attacks targeting SRS of the skill speech processing. The problem I find in the existing adversarial attacks is that they are not practical in terms of efficiency and imperceptibility. Thus, my research question in Phase 2 is how to perturb a speech efficiently and imperceptibly?

- The third step of the skill pipeline processed text data generated from the speech processing step. This intent extraction step aims to precisely produce the users' intent based on the text data. However, I find that existing work lacks understanding beyond the speech processing step of VA processing. Hence, the research question in Phase 3 is that what is beyond speech processing? Can they be problematic and cause potential misinterpretation of users' voice commands?

- After processing users' speech input in the previous three steps, the last step of the skill pipeline is to process the users' request. The problem is that given a limited understanding of third-party skills' back-end processing, it is difficult to pinpoint and fix the problems in this step directly. In Phase 4, I question whether these third-party skills can conduct malicious behaviors? Moreover, do users understand the risks of using potentially dangerous skills?

8.1.2 Research Challenges

I conclude that the main challenge of this dissertation is to tackle the research questions with practicability and usability considered. While different technical challenges are posed by the research questions mentioned above and the unique designs of each step in the skill pipeline, I present a summary of how they are related to keeping VA security practical and usable.

- In Phase 1, I present a security overlay named AEOLUS to embed acoustic nonce in the open acoustic channel. The design goal is the use AEOLUS to ensure the freshness of any speech input. The challenge is that acoustic nonce propagation in the acoustic channel can be unstable. Hence, to make the acoustic nonce embedding practical, it is important to ensure that the acoustic nonce transmission is reliable. Moreover, for usability purposes, the generated acoustic nonce should be as imperceptible as possible.

- In Phase 2, the design goal of one pitch attack is to find more effective areas in audio data to perturb. One of the challenging parts is to locate such areas accurately. Moreover, similar to Phase 1, another challenge is to keep the perturbed audio imperceptible. This is critical to launch such attacks practically and use them for other purposes.

- In Phase 3, I first discuss the problematic intent extraction processing in the NLU of VA systems. The design goal of LipFuzzer is to find out potentially vulnerable voice commands that may cause semantic misinterpretations. The challenge is how to define the fuzzing rules to tackle the problem of a large searching space when mutating a voice command. The reason is that mutating any possible phonemes and letters can lead to impractically large fuzzing results. A usable security tool should be designed to solve the problem in a reasonable time and cost.

- In Phase 4, I design a user study to understand the user perception of using potentially dangerous third-party skills. The problem is how to capture the participants' thoughts regarding using skills faithfully. This is challenging because launch a useful user study needs to con-

sider many practical issues. For example, how to make sure there are no leading questions?

8.1.3 Research Outcome

This dissertation provides a thorough security assessment for different steps in the skill pipeline. In Phase 1, I evaluate AEOLUS in three different environments and show that it works reliably up to a range of 4 m. Additionally, I conduct a user study involving 120 subjects (approved by the institutional human study review board) to evaluate the imperceptibility of the embedded nonce. The study results show that most users find the embedded acoustic nonce to be either imperceptible or non-disruptive. In Phase 2, I evaluate one pitch attack with the VoxCeleb dataset [100, 101] and four different SRSs systems from both open-source or commercial tools. The result shows that one pitch attack achieves a 99% success rate on attention-based SRS and up to 99% success rate for other systems (in overlay mode). In Phase 3, I first show that intent extraction is the root cause of the misinterpretation. Then, I demonstrate that LipFuzzer can systematically pinpoint semantic inconsistencies with the help of linguistic models generated from existing linguistic knowledge. I scanned Alexa Skill Store and Google Assistant Store to assess the security problem I found. My result shows that 26,790 (out of 32,892) Amazon Alexa Skills and 2,295 (out of 2,328) Google Assistant Actions are potentially vulnerable. In Phase 4, I highlight three findings. First, both the attention and comprehension rates for VUI-related security indicators are low. Second, VUI-related skill permissions are vaguely described. As a result, few respondents understood their security consequences. Third, most respondents (71%) did not understand who would be accessing their resources.

As a result of this research, I not only provided a systematic understanding of the current skill pipeline but also pinpointed the key security and privacy concerns in each step of this pipeline. Moreover, I designed security tools and assessment studies to evaluate and start to mitigate the problems I observed.

8.2 Lessons Learned

The four phases mentioned above illustrate how skill systems can be exploited. In addition, I present security countermeasures to start to mitigate the problems. Furthermore, I introduce other takeaways to shed light on future innovations to understand and secure VA applications.

8.2.1 Using Acoustic Nonce as a Generalized Tool

In Phase 1, the acoustic communication techniques are leveraged to embed acoustic nonce in the acoustic channel. I think that acoustic nonce can be further improved. First, different acoustic communication techniques should be tested. For example, chirp spread spectrum [153] could be used to allow more reliable nonce transmission. Second, instead of the digitally modulated acoustic nonce, an analog nonce could improve imperceptibility. For example, different natural sounds can be used to embed a nonce. Moreover, the concept of using acoustic nonce can be applied in different scenarios, such as cellular networks. For instance, AEOLUS can be used in phone banking services to ensure the freshness of customer calls.

However, tackling speech re-use can only solve a part of the issues that exist in the acoustic channel and physical space where VA devices are located. The are many remaining problems, such as device attestation and other hardware security tasks.

8.2.2 Using Adversarial Examples with a Benign Setting.

In Phase 2, the efficiency property provided by one pitch attack can be used in many different scenarios where fast and scalable adversarial perturbations are needed. For example, in the cloaking tasks for preserving data privacy [154], it is important to make sure the adversarial example generation method can be scaled to fit extensive data input. They can potentially solve the security and privacy concerns of publishing videos and audios on widespread media sharing platforms such as YouTube, Twitter. The reason is that attackers may leverage these data to train speech synthesis models and launch spoofing attacks.

The open question is how to leverage the adversarial examples and make sure itself is not vulnerable? My observation is that this could be challenging because current adversarial machine

learning techniques are still very data-dependent, and the subsequent bias in a model would cause further attack opportunities.

8.2.3 Building Robust Intent Extraction in Skill Development Process

As mentioned in Phase 3, the current intent extraction is based on a classifier trained on developers' template inputs. The key issue is that this allows malicious developers to register malicious templates. The proposed Lipfuzzer can potentially be used to help build a more robust intent classifier by generating corner cases and potentially adversarial templates. Specifically, the mutated fuzzing results can be used to test how different classifier designs perform in the presence of malicious developers.

However, Lipfuzzer can only provide recommendations of how a voice command could be misspoken. Thus, future innovations of intent classification need a less developer-dependent and template-driven intent extraction process. This is an open problem that may involve different state-of-the-art machine learning techniques such transformers [20] in natural language understanding.

8.2.4 Interacting with Third-party Skills

For many years, users have been educated in using different mobile and web applications. In Phase 4, I show that many users' behaviors are affected by the knowledge inherited from their past experience using these mobile apps or other applications. For example, users tend to trust applications downloaded from Google Play Store. However, there is a significant difference between the traditional app store and the Alexa skill store. This is because skills are not vetted based on their source code [3]. Therefore, it could be dangerous to trust a skill even it is after the store vetting because the output from the skills can be malicious or policy-violating [3, 2]. The remaining question is how to educate users and prevent harmful cognitive inertia? Also, in the context of VUI, what is the harmful inertia, and what could be helpful?

Moreover, as many services are migrating to the cloud, it is becoming an increasingly challenging task to study security and privacy-related resource usage in modern applications. This is also a big problem for Alexa skills and other voice assistant platforms. The reason is that almost

all these platforms let the third-party developers hosted skills on *any* cloud-based web services. As a result, it is difficult for the platforms to moderate the skill behaviors if the skill is not hosted in the same domain. Thus, the question is, how can a cloud-hosted skill be checked or monitored by the platforms or security researchers? This also leads to another question of how to enforce cloud-based permission systems?

9. CONCLUSION AND FUTURE WORK

In this dissertation, I present four chapters to cover the security assessment and defense strategies for four skill processing stages. Also, I use one chapter to describe the lessons learned from these studies.

In Chapter 4, I introduce the unsolved problem of speech re-use in the acoustic channel of skill processing. Then, to ensure the freshness of user speech input, a skill security overlay called AEOLUS is proposed to perform acoustic nonce embedding. It can be integrated with any VA to prevent speech re-use. AEOLUS reliably embeds a dynamic acoustic nonce that is non-disruptive to a VA user and detects the embedded nonce from the recorded user speech. Experiments conducted in three different environments show that AEOLUS can be used in operational VA scenarios with minimal overhead.

In Chapter 5, I proposed one pitch attack to solve the problem of inefficient and potentially noisy global perturbation which is commonly used in traditional adversarial attacks. Specifically, I adopted an attention-based strategy learned from state-of-art Transformer architecture. Also, with help of audio steganography, this scheme can effectively and imperceptibly attack SRSs with upto 99% success rate in a few hundred queries. One pitch attack is also model-agnostic and can work as an overlay method over traditional adversarial attacking algorithms.

In Chapter 6, I showcase how to systematically study the skill intent extraction process which affects the security of popular skills. I first found that the currently used skill templates may cause dangerous semantic inconsistencies. To systematically discover the speech misinterpretations, I designed the first linguistic-guided fuzzing tool based on natural language processing techniques. I have published Lipfuzzer source code and dataset used in this research to help not only skill developers but also VA system designers to create better templates with fewer speech misinterpretations. This also benefits the research community to work on this important IoT security area.

In Chapter 7, I present the effectiveness study of security indicators in the popular Amazon Alexa platform. To understand how users perceive the skill security indicators, I performed two

user studies: a user survey and a skill experiment. Our findings show that Alexa users pay little to no attention to some critical skill security indicators and often misinterpret the risks associated with skill permissions. Additionally, many users have misplaced trust in third-party skills wherein they may expect all skills to be safe because they are backed or owned by Amazon/Alexa. As this can lead to users falling victim to undesirable or even malicious skills, I discussed recommended changes to skill security indicators to improve their effectiveness.

In Chapter 8, I discuss a summarized overview of this dissertation and lessons learned from studying different stages in current skill design. The summary illustrates how four studies are related to each other and existing research. Also, the lessons learned section poses open questions and discussions based on my observations while conducting the research.

In the future, I plan to further explore more security and privacy problems of different stages in the skill pipeline. For the acoustic channel, I hope to conduct an in-depth investigation of methods for generating and encoding the acoustic nonce, e.g., using variations of music or other forms of sound and different modulation schemes. I also plan to conduct large-scale live subject testing to evaluate reliability and imperceptibility further. Finally, for speech processing, more studies regarding speech data protection are expected. My idea is to leverage adversarial examples to de-identify speech data with practical considerations. For intent extraction, I first want to improve the model with more complicated logical representations to cover more LAPSUS. Also, improving the training and mutation process of LipFuzzer is another direction to improve LipFuzzer. For skill processing, I plan to explore more about skill behaviors. Specifically, the future work will be automatically checking skill misbehavior with the help of chatbot techniques.

Also, I plan to study other IoT platforms with novel user interface designs such as augmented reality and virtual reality. My goal is to find out the key difference between these platforms and conventional ones. These differences could incur security and privacy risks to both the users and service providers. Also, the usage of multiple user interfaces in these platforms can be interesting to study. This is because the inconsistencies among different user interfaces can lead to unexpected consequences.

REFERENCES

[1] "Voicebot.AI: Amazon Alexa Has 100k Skills." https://voicebot.ai/2019/10/01/amazon-alexa-has-100k-skills-but-momentum\-slows-globally-here-is-the-breakdown-by-\country/.

[2] Z. Guo, Z. Lin, P. Li, and K. Chen, "Skillexplorer: Understanding the behavior of skills in large scale," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pp. 2649–2666, 2020.

[3] L. Cheng, C. Wilson, S. Liao, J. Young, D. Dong, and H. Hu, "Dangerous skills got certified: Measuring the trustworthiness of skill certification in voice personal assistant platforms," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1699–1716, 2020.

[4] "Voice Assistant User Population Survey." https://www.emarketer.com/Article/Alexa-Say-What-Voice-Enabled-Speaker-Usage\-Grow-Nearly-130-This-Year/1015812.

[5] "Smart Speaker Market Share in 2019." https://www.androidcentral.com/amazon-dominates-smart-speaker-market\-nearly-70-share-2019.

[6] "Amazon prize: The socialbot challenge." https://developer.amazon.com/alexaprize/2017-alexa-prize.

[7] "Natural language understanding – google ai." https://ai.google/research/teams/nlu/.

[8] "Stanford corenlp." https://stanfordnlp.github.io/CoreNLP/index.html.

[9] C. Lentzsch, S. J. Shah, B. Andow, M. Degeling, A. Das, and W. Enck, "Hey Alexa, is this skill safe?: Taking a closer look at the Alexa skill ecosystem," in *Proceedings of the 28th*

ISOC Annual Network and Distributed Systems Symposium (NDSS), 2021.

[10] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 513–530, 2016.

[11] "Just say it: The future of search is voice and personal digital assistants." https://www.campaignlive.co.uk/article/just-say-it-future-search-voice-personal-digital-assistants/1392459.

[12] "Alexa skill interaction model." https://developer.amazon.com/docs/alexa-voice-service/interaction-model.html.

[13] "Voice Payment Survey." https://bit.ly/3p6k97z.

[14] "Amazon Alexa Pay in Gas Station.." https://bit.ly/37v25xN.

[15] "Paysafe Study." https://bit.ly/2WtgjJl.

[16] N. Scaife, C. Peeters, and P. Traynor, "Fear the reaper: Characterization and fast detection of card skimmers," in *27th USENIX Security Symposium*, 2018.

[17] "Alexa for Business." https://aws.amazon.com/alexaforbusiness/.

[18] "Will voice assistant change workplace." https://bit.ly/3mxRDKm.

[19] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[21] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[22] M. India, P. Safari, and J. Hernando, "Self multi-head attention for speaker recognition," *arXiv preprint arXiv:1906.09890*, 2019.

[23] W. Cai, J. Chen, and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," *arXiv preprint arXiv:1804.05160*, 2018.

[24] R. Wegel and C. Lane, "The auditory masking of one pure tone by another and its probable relation to the dynamics of the inner ear," *Physical review*, vol. 23, no. 2, p. 266, 1924.

[25] H. Matsuoka, "Spread spectrum audio steganography using sub-band phase shifting," in *2006 International Conference on Intelligent Information Hiding and Multimedia*, pp. 3–6, IEEE, 2006.

[26] R. Scholtz, "The origins of spread-spectrum communications," *IEEE Transactions on communications*, vol. 30, no. 5, pp. 822–854, 1982.

[27] D. Kahn, "Cryptology and the origins of spread spectrum: Engineers during world war ii developed an unbreakable scrambler to guarantee secure communications between allied leaders; actress hedy lamarr played a role in the technology," *IEEE spectrum*, vol. 21, no. 9, pp. 70–80, 1984.

[28] 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 *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, pp. 63–74, ACM, 2014.

[29] T. Vaidya, Y. Zhang, M. Sherr, and C. Shields, "Cocaine noodles: exploiting the gap between human and machine speech recognition," *WOOT*, vol. 15, pp. 10–11, 2015.

[30] X. Yuan, Y. Chen, Y. Zhao, Y. Long, X. Liu, K. Chen, S. Zhang, H. Huang, X. Wang, and C. A. Gunter, "Commandersong: A systematic approach for practical adversarial voice recognition," in *27th USENIX Security Symposium*, 2018.

[31] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "Dolphinattack: Inaudible voice commands," in *ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[32] N. Roy, H. Hassanieh, and R. Roy Choudhury, "Backdoor: Making microphones hear inaudible sounds," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 2–14, ACM, 2017.

[33] D. Kumar, R. Paccagnella, P. Murley, E. Hennenfent, J. Mason, A. Bates, and M. Bailey, "Skill squatting attacks on amazon alexa," in *27th USENIX Security Symposium*, pp. 33–47, 2018.

[34] N. Zhang, X. Mi, X. Feng, X. Wang, Y. Tian, and F. Qian, "Understanding and mitigating the security risks of voice-controlled third-party skills on amazon alexa and google home," *arXiv preprint 1805.01525*, 2018.

[35] S. Chen, K. Ren, S. Piao, C. Wang, Q. Wang, J. Weng, L. Su, and A. Mohaisen, "You can hear but you cannot steal: Defending against voice impersonation attacks on smartphones," in *International Conference on Distributed Computing Systems*, 2017.

[36] M. E. Ahmed, I.-Y. Kwak, J. H. Huh, I. Kim, T. Oh, and H. Kim, "Void: A fast and light voice liveness detection system," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pp. 2685–2702, 2020.

[37] T. Kinnunen, M. Sahidullah, H. Delgado, M. Todisco, N. Evans, J. Yamagishi, and K. A. Lee, "The ASVspoof 2017 challenge: Assessing the limits of replay spoofing attack detection," 2017.

[38] H. Feng, K. Fawaz, and K. G. Shin, "Continuous authentication for voice assistants," in *International Conference on Mobile Computing and Networking*, 2017.

[39] S. Pradhan, W. Sun, G. Baig, and L. Qiu, "Combating replay attacks against voice assistants," *ACM Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 3, 2019.

[40] M. D. Swanson, B. Zhu, A. H. Tewfik, and L. Boney, "Robust audio watermarking using perceptual masking," *Signal processing*, vol. 66, no. 3, pp. 337–355, 1998.

[41] J. W. Seok and J. W. Hong, "Audio watermarking for copyright protection of digital audio data," *Electronics Letters*, vol. 37, no. 1, pp. 60–61, 2001.

[42] P. Bassia, I. Pitas, and N. Nikolaidis, "Robust audio watermarking in the time domain," *IEEE Transactions on multimedia*, vol. 3, no. 2, pp. 232–241, 2001.

[43] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon, "A secure, robust watermark for multimedia," in *International Workshop on Information Hiding*, 1996.

[44] C. Neubauer and J. Herre, "Digital watermarking and its influence on audio quality," in *Audio Engineering Society Convention 105*, Audio Engineering Society, 1998.

[45] S. Wu, J. Huang, D. Huang, and Y. Q. Shi, "Efficiently self-synchronized audio watermarking for assured audio data transmission," *IEEE Transactions on Broadcasting*, vol. 51, no. 1, pp. 69–76, 2005.

[46] D. Kirovski and H. S. Malvar, "Spread-spectrum watermarking of audio signals," *IEEE transactions on signal processing*, vol. 51, no. 4, pp. 1020–1033, 2003.

[47] M. S. Wogalter, "Communication-human information processing (c-hip) model," *Handbook of warnings*, pp. 51–61, 2006.

[48] S. Egelman, L. F. Cranor, and J. Hong, "You've been warned: an empirical study of the effectiveness of web browser phishing warnings," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1065–1074, 2008.

[49] A. P. Felt, H. J. Wang, A. Moshchuk, S. Hanna, and E. Chin, "Permission re-delegation: Attacks and defenses.," in *USENIX Security Symposium*, vol. 30, p. 88, 2011.

[50] K. R. Laughery and M. S. Wogalter, "A three-stage model summarizes product warning and environmental sign research," *Safety science*, vol. 61, pp. 3–10, 2014.

[51] "Voice Password in Samsung Bixby." `https://bit.ly/34rlPAN`.

[52] "Voice Password in Wechat." `https://zd.net/3myAGQ2`.

[53] "Voice Password Applications in Google Play." `https://bit.ly/38dq1oG`.

[54] Y. Gong and C. Poellabauer, "Protecting voice controlled systems using sound source identification based on acoustic cues," in *International Conference on Computer Communication and Networks*, 2018.

[55] M. Leng, S. S. Arora, and K. Wagner, "Replay spoofing detection for automatic speaker verification system," 2020. US Patent App. 16/535,836.

[56] "Black Hat 2018: Voice Authentication is Broken." `https://bit.ly/3mxQIto`.

[57] S. Wang, J. Cao, X. He, K. Sun, and Q. Li, "When the differences in frequency domain are compensated: Understanding and defeating modulated replay attacks on automatic speech recognition," in *ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[58] "Face ID Security." `https://apple.co/34mmO59`.

[59] N. Cvejic, *Algorithms for audio watermarking and steganography*. 2004.

[60] N. Cvejic and T. Seppänen, "Spread spectrum audio watermarking using frequency hopping and attack characterization," *Signal processing*, vol. 84, no. 1, pp. 207–213, 2004.

[61] T. Chen, L. Shangguan, Z. Li, and K. Jamieson, "Metamorph: Injecting inaudible commands into over-the-air voice controlled systems," *Proceedings of NDSS*, 2020.

[62] "Paypal Voice Assistant Application with Pins." `https://bit.ly/3h8QkR6`.

[63] J. L. Tsai, "Efficient nonce-based authentication scheme for session initiation protocol.," *IJ Network Security*, vol. 9, no. 1, pp. 12–16, 2009.

[64] M. Bellare and P. Rogaway, "Provably secure session key distribution: the three party case," in *ACM symposium on Theory of computing*, 1995.

[65] "Alexa skill: 300,000 daily users." `https://vux.world/google-assistant-500m-active-users-popular-alexa-skill-300000-daily-us`

[66] J. Park, K. Park, and J. R. Yoon, "Underwater acoustic communication channel simulator for flat fading," *Japanese Journal of Applied Physics*, vol. 49, no. 7S, 2010.

[67] B. Truax, *Acoustic communication*. Greenwood Publishing Group, 2001.

[68] Y. Gong, J. Yang, J. Huber, M. MacKnight, and C. Poellabauer, "Remasc: Realistic replay attack corpus for voice controlled systems," *arXiv preprint 1904.03365*, 2019.

[69] P. Kumar, M. Jayakumar, and A. Vidyapeetham, "Comparison of bit error rate for propagation mechanisms of millimeter waves in a practical communication systems employing psk and fsk," *PIERS Proceedings*, 2010.

[70] D. H. Johnson, "Signal-to-noise ratio," *Scholarpedia*, vol. 1, no. 12, p. 2088, 2006.

[71] "Personal Music Players & Hearing." `https://bit.ly/34tyWkN`.

[72] A. Wabnitz, N. Epain, C. Jin, and A. Van Schaik, "Room acoustics simulation for multichannel microphone arrays," in *International Symposium on Room Acoustics*, 2010.

[73] H. Abdullah, W. Garcia, C. Peeters, P. Traynor, K. Butler, and J. Wilson, "Practical hidden voice attacks against speech and speaker recognition systems," *arXiv preprint 1904.05734*, 2019.

[74] Y. Suzuki and H. Takeshima, "Equal-loudness-level contours for pure tones," *The Journal of the Acoustical Society of America*, vol. 116, no. 2, pp. 918–933, 2004.

[75] "Equal-loudness contour.." `https://bit.ly/2LF1ukH`.

[76] "IoSR Matlab Toolbox: Equal-loudness Contour Mapping." `https://bit.ly/2Wtd1Wz`.

[77] G. Chen, S. Chen, L. Fan, X. Du, Z. Zhao, F. Song, and Y. Liu, "Who is real bob? adversarial attacks on speaker recognition systems," *IEEE Symposium on Security and Privacy*, 2021.

[78] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE transactions on evolutionary computation*, vol. 15, no. 1, pp. 4–31, 2010.

[79] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[80] E. A. Lee and D. G. Messerschmitt, *Digital communication*. Springer Science & Business Media, 2012.

[81] "Microsoft Cognitive Services." `https://bit.ly/38haWCH`.

[82] "Microsoft Azure Speaker Recognition - Verification." `https://bit.ly/3h82Sbh`.

[83] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke, "The Microsoft 2017 conversational speech recognition system," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018.

[84] "MP3 Wikipedia." `https://en.wikipedia.org/wiki/MP3`.

[85] F. Wilcoxon, S. K. Katti, and R. A. Wilcox, "Critical values and probability levels for the wilcoxon rank sum test and the wilcoxon signed rank test," *Selected tables in mathematical statistics*, vol. 1, pp. 171–259, 1970.

[86] M. S. Kang, V. D. Gligor, V. Sekar, *et al.*, "Spiffy: Inducing cost-detectability tradeoffs for persistent link-flooding attacks.," in *NDSS*, vol. 1, pp. 53–55, 2016.

[87] S. B. Lee, M. S. Kang, and V. D. Gligor, "Codef: Collaborative defense against large-scale link-flooding attacks," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pp. 417–428, 2013.

[88] Y. Soupionis and D. Gritzalis, "Audio captcha: Existing solutions assessment and a new implementation for voip telephony," *Computers & Security*, vol. 29, no. 5, pp. 603–618, 2010.

[89] P. Poveda-Martinez, R. Peral-Orts, N. Campillo-Davo, J. Nescolarde-Selva, M. Lloret-Climent, and J. Ramis-Soriano, "Study of the effectiveness of electric vehicle warning sounds depending on the urban environment," *Applied Acoustics*, vol. 116, pp. 317–328, 2017.

[90] C. Geeng, "Egregor: An eldritch privacy mental model for smart assistants," in *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–9, 2020.

[91] "Echo Device Light Indicator." `https://www.amazon.com/gp/help/customer/display.html?nodeId=GKLDRFT7FP4FZE56`.

[92] "Google Cloud Multiple Voices Speaker Diarization." `https://cloud.google.com/speech-to-text/docs/multiple-voices#speaker_diarization`.

[93] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*, pp. 387–402, Springer, 2013.

[94] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[95] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 1–7, IEEE, 2018.

[96] Y. Qin, N. Carlini, G. Cottrell, I. Goodfellow, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," in *International conference on machine learning*, pp. 5231–5240, PMLR, 2019.

[97] R. Taori, A. Kamsetty, B. Chu, and N. Vemuri, "Targeted adversarial examples for black box audio systems," in *2019 IEEE Security and Privacy Workshops (SPW)*, pp. 15–20, IEEE, 2019.

[98] Y. Chen, X. Yuan, J. Zhang, Y. Zhao, S. Zhang, K. Chen, and X. Wang, "Devil's whisper: A general approach for physical adversarial attacks against commercial black-box speech recognition devices," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pp. 2667–2684, 2020.

[99] F. Kreuk, Y. Adi, M. Cisse, and J. Keshet, "Fooling end-to-end speaker verification with adversarial examples," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1962–1966, IEEE, 2018.

[100] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," *arXiv preprint arXiv:1706.08612*, 2017.

[101] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," *arXiv preprint arXiv:1806.05622*, 2018.

[102] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[103] K. Fleetwood, "An introduction to differential evolution," in *Mathematics and Statistics of Complex Systems Symposium*, 2004.

[104] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 161–165, IEEE, 2018.

[105] "s3prl." `https://github.com/s3prl/s3prl`.

[106] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.

[107] "Kaldi." `https://github.com/kaldi-asr/kaldi`.

[108] S. wen Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin, T.-H. Huang, W.-C. Tseng, K. tik Lee, D.-R. Liu, Z. Huang, S. Dong, S.-W. Li, S. Watanabe, A. Mohamed, and H. yi Lee, "Superb: Speech processing universal performance benchmark," 2021.

[109] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5206–5210, IEEE, 2015.

[110] G. S. Dell, C. Juliano, and A. Govindjee, "Structure and content in language production: A theory of frame constraints in phonological speech errors," *Cognitive Science*, vol. 17, no. 2, pp. 149–195, 1993.

[111] "Oxford living dictionary - confusable words." `https://www.merriam-webster.com/dictionary/confusable`.

[112] J. H. Hulstijn and W. Hulstijn, "Grammatical errors as a function of processing constraints and explicit knowledge," *Language learning*, vol. 34, no. 1, pp. 23–43, 1984.

[113] "30 common grammar mistakes to check for in your writing." `https://blog.hubspot.com/marketing/common-grammar-mistakes-list`.

[114] G. S. Dell, "Representation of serial order in speech: Evidence from the repeated phoneme effect in speech errors.," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 10, no. 2, p. 222, 1984.

[115] A. W. Ellis, "Errors in speech and short-term memory: The effects of phonemic similarity and syllable position," *Journal of Verbal Learning and Verbal Behavior*, vol. 19, no. 5, pp. 624–634, 1980.

[116] "Wikipedia speech error." `https://en.wikipedia.org/wiki/Speech_error`.

[117] M. Böhme, V.-T. Pham, M.-D. Nguyen, and A. Roychoudhury, "Directed greybox fuzzing," in *Proc. of the 2017 ACM CCS*, 2017.

[118] K. Kersting, L. De Raedt, and S. Kramer, "Interpreting bayesian logic programs," in *Proceedings of the AAAI-2000 workshop on learning statistical models from relational data*, pp. 29–35, 2000.

[119] L. Getoor and B. Taskar, *Introduction to statistical relational learning*, vol. 1. MIT press Cambridge, 2007.

[120] M. Richardson and P. Domingos, "Markov logic networks," *Machine learning*, vol. 62, no. 1-2, pp. 107–136, 2006.

[121] C. Boitet and M. Seligman, "The whiteboard architecture: A way to integrate heterogeneous components of nlp systems," in *Proceedings of the 15th conference on Computational linguistics-Volume 1*, 1994.

[122] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *Proc. of the ACL Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics*, 2002.

[123] "Introduction to information retrieval: Stemming and lemmatization." https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html.

[124] "Arpabet phonetic transcription codes." https://en.wikipedia.org/wiki/ARPABET.

[125] "Text-to-speech tools." https://help.ubuntu.com/community/TextToSpeech.

[126] "Amazon polly: Turn text into lifelike speech." https://www.mturk.com/.

[127] "Google cloud text-to-speech." https://cloud.google.com/text-to-speech/.

[128] "Amazon mechanical turk crowdsourcing platform." https://www.mturk.com/.

[129] "Amazon Alexa Account Linking Documentation." https://developer.amazon.com/en-US/docs/alexa/account-linking/account-linking-concepts.html.

[130] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 627–638, 2011.

[131] C. Thompson, M. Shelton, E. Stark, M. Walker, E. Schechter, and A. P. Felt, "The web's identity crisis: understanding the effectiveness of website identity indicators," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pp. 1715–1732, 2019.

[132] "Google Play Protect." `https://www.android.com/play-protect/`,.

[133] "Where to host an Alexa Skill." `https://developer.amazon.com/en-US/docs/alexa/custom-skills/understanding-custom-skills.html`. Accessed April 30, 2020.

[134] 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 *IEEE Symposium on Security and Privacy*, 2019.

[135] K. M. Farley, J. O'Reilly, L. Squire, and R. Beasley, *Voice application development with VoiceXML*. Pearson Education, 2001.

[136] C. Murad, C. Munteanu, B. R. Cowan, and L. Clark, "Revolution or evolution? speech interaction and hci design guidelines," *IEEE Pervasive Computing*, vol. 18, no. 2, pp. 33–45, 2019.

[137] D. Recordon and D. Reed, "Openid 2.0: a platform for user-centric identity management," in *Proceedings of the second ACM workshop on Digital identity management*, pp. 11–16, ACM, 2006.

[138] "Alexa Skill: Mapnav." `https://www.amazon.com/none-mapNav/dp/B0849QD5F4/`.

[139] "Alexa Permission in Custom Skill." `https://developer.amazon.com/en-US/docs/alexa/custom-skills/`

configure-permissions-for-customer-\information-in-your-skill.html.

[140] "Alexa Account Linking: Available or Required." https://developer.amazon.com/en-US/docs/alexa/account-linking/configure-optional-account-linking.html.

[141] "Alexa Developer Documentation: Account Linking Concept." https://developer.amazon.com/en-US/docs/alexa/account-linking/account-linking-concepts.html.

[142] "Alexa Skill: Wiffy." https://www.amazon.com/hartman-Wiffy/dp/B06WVB8WM7/.

[143] "Alexa Skill: Twenty Questions ." https://www.amazon.com/Amazon-Twenty-Questions/dp/B01C3CO48G.

[144] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android permissions: User attention, comprehension, and behavior," in *Proceedings of the eighth symposium on usable privacy and security*, pp. 1–14, 2012.

[145] D. J. Hauser and N. Schwarz, "Attentive turkers: Mturk participants perform better on online attention checks than do subject pool participants," *Behavior research methods*, vol. 48, no. 1, pp. 400–407, 2016.

[146] N. Stewart, J. Chandler, and G. Paolacci, "Crowdsourcing samples in cognitive science," *Trends in cognitive sciences*, vol. 21, no. 10, pp. 736–748, 2017.

[147] H. Almuhimedi, F. Schaub, N. Sadeh, I. Adjerid, A. Acquisti, J. Gluck, L. F. Cranor, and Y. Agarwal, "Your location has been shared 5,398 times! a field study on mobile app privacy nudging," in *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pp. 787–796, 2015.

[148] "Alexa Reminder in Alexa Skill Kit." https://developer.amazon.com/en-US/docs/alexa/smapi/alexa-reminders-overview.html.

[149] "Alexa List in Alexa Skill Kit." https://developer.amazon.com/en-US/docs/alexa/custom-skills/access-the-alexa-shopping-and-to-do-lists.html.

[150] "Alexa Skill: debugMining." https://www.amazon.com/%E8%B0%A2%E5%8B%87-debugMining/dp/B07FFKLPY2.

[151] "What to do when your iPhone says an app has been using your location in the background." https://qz.com/1713581/what-to-do-when-an-iphone-app-is-using-\your-location-in-the-\background/.

[152] "About Twitter Verified Account." https://help.twitter.com/en/managing-your-account/about-twitter-verified-accounts.

[153] B. Reynders and S. Pollin, "Chirp spread spectrum as a modulation technique for long range communication," in *2016 Symposium on Communications and Vehicular Technologies (SCVT)*, pp. 1–5, IEEE, 2016.

[154] S. Shan, E. Wenger, J. Zhang, H. Li, H. Zheng, and B. Y. Zhao, "Fawkes: Protecting privacy against unauthorized deep learning models," *arXiv preprint arXiv:2002.08327*, 2020.