Preprint of paper to appear at 14th ACM ASIA Conference on Computer and Communications Security (ASIACCS 2019), Auckland, New Zealand, July 2019.

Alexa Lied to Me: Skill-based Man-in-the-Middle Attacks on Virtual Assistants

Richard Mitev
richard.mitev@trust.tu-darmstadt.de
Technische Universität Darmstadt
Germany

Markus Miettinen
markus.miettinen@trust.tu-darmstadt.de
Technische Universität Darmstadt
Germany

Ahmad-Reza Sadeghi
ahmad.sadeghi@trust.tu-darmstadt.de
Technische Universität Darmstadt
Germany

ABSTRACT

Voice-based virtual personal assistants such as Amazon's *Alexa* or *Google Assistant* have become highly popular and are used for diverse daily tasks ranging from querying on-line information, shopping, smart home control and a variety of enterprise application scenarios[1]. Capabilities of virtual assistants can be enhanced with so-called *Skills*, i.e., programmatic extensions that allow third-party providers to integrate their services with the respective voice assistant.

In this paper, we show that specially crafted *malicious Skills* can use the seemingly limited Skill interaction model to cause harm. We present novel man-in-the-middle attacks against benign Skills and Virtual Assistant functionalities. Our attack uses loopholes in the Skill interface to redirect a victim's voice input to a malicious Skill, thereby hijacking the conversation between Alexa and the victim. To the best of our knowledge this is the *first man-in-the-middle attack targeting the Skill ecosystem*. We present the design of our attack and demonstrate its feasibility based on a proof-of-concept implementation attacking the Alexa Skills of a smart lock as well as a home security system.

ACM Reference Format:
Richard Mitev, Markus Miettinen, and Ahmad-Reza Sadeghi. 2019. Alexa Lied to Me: Skill-based Man-in-the-Middle Attacks on Virtual Assistants. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS '19), July 9–12, 2019, Auckland, New Zealand.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3321705.3329842

1 INTRODUCTION

Personal voice-controlled virtual assistants are getting more and more popular and ubiquitous. There are numerous virtual assistants available on the market from different vendors like, e.g., *Amazon Alexa*, *Google Assistant*, *Apple Siri* and *Microsoft Cortana*. Virtual assistants can be integrated into other programs, built into OSs or into IoT devices like smart speakers, smart watches, appliances, cars or even clothing.

[1]https://aws.amazon.com/alexaforbusiness/

Recently, multiple new attacks against devices utilizing voice control-based UIs have emerged, mainly focused on issuing unauthorized commands without the legitimate user noticing this. The attacks typically utilize synthesized [1, 6], obfuscated [3, 21], embedded [16, 24] or ultrasound-modulated voice commands [13, 19, 25]. Besides sound, also other attack vectors exist, e.g., using electromagnetic interference [10] or attacks leveraging the headphone jack of a device [23]. There are also attacks exploiting the way we pronounce words to invoke a similar sounding Skill [12, 26].

What makes the voice-control interface of virtual assistants particularly interesting is that it can be augmented with various third-party service applications, called 'Skills', or 'Actions' that provide extended functionality beyond the basic functions of the virtual assistant, like, e.g., the possibility to control smart home appliances, or, access to specific information services. It is obvious that by misusing such 'Skills' an adversary could potentially cause much harm to the user. Therefore the interaction model of Skills is very limited: Skills can only be used to retrieve information from a third-party service or invoke actions explicitly exposed by the service.

Our goals and contributions. In this paper, we show that carefully crafted malicious back-end Skill functionality can be combined with known inaudible attack techniques to circumvent the seemingly limited Skill interaction model of a virtual assistant like Amazon Alexa to allow an adversary to *arbitrarily control and manipulate interactions* between the user and other benign Skills. We present a man-in-the-middle attack that *completely hijacks* a full conversation between a voice-controlled virtual assistant and the targeted user and is very hard to detect. The advantage of our approach using a malicious back-end Skill for the attack is that it works *in the context of an active user interaction* with the virtual assistant, completely *maintaining the interaction semantics* from the user's perspective. This allows the adversary to launch much more powerful attacks than simple command injection as presented in earlier work [3, 14, 16, 19, 21, 24, 25].

In our attack the adversary can arbitrarily modify the virtual assistant's responses, or, can entirely replace its functionality by her own, returning any responses of her choice to the victim that seem to be coming from genuine personal virtual assistant. Furthermore, our attack can make the responses seem plausible even to a suspicious and particularly vigilant user by maliciously *modifying genuine, plausible data* provided by benign Skills *on-the-fly*. With this Proof-of-Concept we show that designers of voice controlled applications in Skill ecosystems must take inaudible injection and jamming attacks into account.

To implement our attack we tackle a number of technical challenges: (1) How to capture user commands and redirect them to a malicious Skill so that the original intention of the user is overridden? (2) How to exfiltrate genuine information from legitimate Skills so that it can be used to form fabricated responses to the user that seem plausible? (3) Finally, how to orchestrate the technical components required by the attack seamlessly together to create the illusion that the user is conversing with the legitimate Alexa functionality or Skills, while it in reality is interacting with a malicious Skill.

Contributions Our main contributions are as follows:

- We present *Lyexa*, the (to the best of our knowledge) *first man-in-the-middle attack against personal virtual assistants* (Sect. 3).
- The implementation of a *malicious Skill framework* that can *convincingly impersonate* the behavior of virtual assistants and benign Skills associated with it, while simultaneously being able to modify this behavior as desired by the adversary (Sect. 4).
- A Proof-of-concept evaluation of the Lyexa attack framework in different rooms and environments, on two smart home security systems and the 10 top Skills of the U.S. Skill store controlled by an *Amazon Echo Dot* virtual assistant device (Sect. 5).

In this paper, we focus on the most popular virtual assistant, i.e., Amazon Alexa. However, we stress that our attack can be modified to cover also other similar virtual assistants from other vendors in a similar fashion. The implementation of the attack components is described in Sect. 4, and suggestions for countermeasures against this attack are given in Sect. 7.

2 PRELIMINARIES

The most popular voice-controlled virtual assistant utilising a smart speaker device placed in users' homes is *Alexa*[2], short for Alexa Voice Service (AVS), from Amazon, with a market share of ca. 70% in US households in 2018. [11]Typically Alexa is integrated into a smart speaker device like the *Amazon Echo* or its smaller variant *Amazon Echo Dot*. A similar set-up applies also to Google's comparable service, the *Google Assistant*. These smart speaker devices are equipped with microphones and loudspeakers in order to facilitate voice-based interaction with the user of the device. As shown in Fig. 1 users issue voice commands to their virtual assistants by uttering a so called *wake word* like "Alexa", or, "Ok, Google". When the corresponding smart speaker recognizes the wake word, it starts recording audio to capture any subsequent commands given by the user (1). The recorded audio is forwarded to the corresponding back-end service like Amazon Voice Service or Google Assistant for speech-to-text translation (2). Depending on the issued commands, the back-end service will look up information or initiate actions (3) and respond with a reply (4) that the smart speaker will play back to the user (5).

In addition to this basic usage scenario, virtual assistant functionality can also be built into dedicated third-party devices. Examples include *BMW cars*[3], where Alexa is able to control the air conditioning system of the car, or, *LG ThinQ Smart fridges and ovens*[4] where Alexa can for instance be used to preheat the oven.

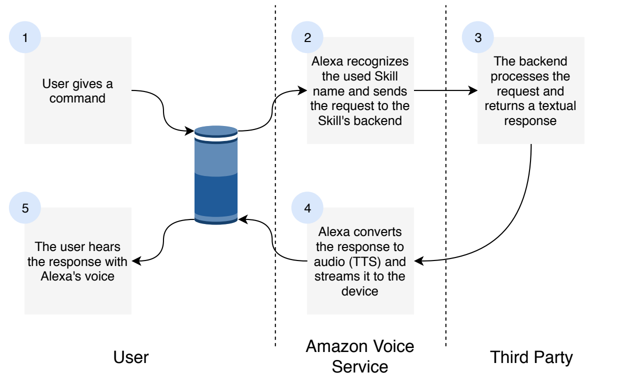

1
User gives a command
2
Alexa recognizes the used Skill name and sends the request to the Skill's backend
3
The backend processes the request and returns a textual response
5
The user hears the response with Alexa's voice
4
Alexa converts the response to audio (TTS) and streams it to the device
User
Amazon Voice Service
Third Party

Figure 1: User interaction flow while using a Skill

2.1 Skills and Actions

The functionality of Alexa can be extended by so-called *Skills*[5] (called *Actions*[6] in Google Assistant). These are third-party service plug-ins that extend the virtual assistant's functionality and allow it to interact with third-party services, generating an entire ecosystem around AVS. Any developer (private or organization) can develop Skills free of charge. Examples of third-party Skills include, e.g., ordering pizza through Domino's Pizza Skill or calling a Uber ride through Uber's Skill, to name a few. Utilizing appropriate Skills, Alexa can also be used for controlling user's smart home IoT devices given that the manufacturers of these devices provide the appropriate Skill for controlling them through Alexa.

In contrast to apps on popular smartphone platforms like Android or iOS, which are essentially pieces of software running on the host smartphone, Skills are different. They don't contain executable code that could be run on devices and they can't thus change the behavior of the Alexa-enabled device itself. Skills are merely third-party-provided service extensions to AVS and can only react to invocations, i.e. telling Alexa what to echo to the user in response to specific requests to the Skill. The Lyexa attack framework utilizes a specially-crafted malicious Skill to make Alexa speak to the user what the adversary wants, mimicking the behavior of genuine Skills or Alexa functionalities. The adversary can thereby exploit the Alexa ecosystem for different attacks without ever having to compromise the Alexa-enabled device itself.

Skills need to be registered in Amazon's Skill repository, from where users can activate them to be used. The activation of Skills can happen through the Alexa app, webpage, or, using a voice command. To be listed in the repository each Skill has to pass a certification process in which Amazon tests the Skill to verify that it works as specified. However, as the Skill is running on a remote

[2]https://www.amazon.com/Amazon-Echo-And-Alexa-Devices/b?node=9818047011

[3]https://www.gearbrain.com/which-cars-have-amazon-alexa-2525958778.html

[4]https://www.cnet.com/news/lg-instaview-thinq-alexa-fridge-clever-kitchen-tricks-ces-2018/

[5]https://developer.amazon.com/alexa-skills-kit

[6]https://developers.google.com/actions/

server, AVS can't control whether a Skill is modified after it has passed the certification.

2.2 Inaudible Speech Injection

Recent attacks against virtual assistants by Song *et al.* [19] and Zhang *et al.* [25] utilize the non-linearity of microphones beyond the audible frequency spectrum to issue commands to a voice-controlled virtual assistant that can't be heard by the user. This is achieved by exploiting physical characteristics of the microphones typically used in such devices. By carefully injecting selected signals at high frequencies that lie beyond the hearing capacity of humans, it is possible to create 'shadow' signals in the microphone within the audible frequency spectrum [14] and use this for issuing inaudible commands. These works assume a malicious device in proximity to the targeted Voice Assistant injecting the inaudible audio signal. We think this to be a realistic assumption and adopt the same adversarial scenario, assuming that the attacker is able to inject signals into the victim's audio environment. To show that Skills can be used for attacks we realise the Lyexa attack where we will take use of this inaudible command injection technique together with inaudible jamming of user commands to redirect the user interaction from the Skill the user intended to the adversary's malicious Skill as described below.

3 OUR ATTACK LYEXA

Our goal with the Lyexa attack is to show that even though the functionality and interaction model of Skills is very limited, it is possible to craft Skills with malicious functionality that can be used to construct harmful attacks against users by utilizing compromised IoT devices in the vicinity of the Alexa device for realizing the attack. The Lyexa attack has been designed under the assumption that the Alexa devices and the AVS ecosystem have no exploitable vulnerabilities and that the traffic is properly secured (e.g., TLS, no ARP spoofing). We think this to be a realistic assumption since Amazon spends considerable resources in designing and testing their devices and would also have the necessary means for quickly updating or patching their devices in case security vulnerabilities affecting them were to be found. With this attack we want to show how the AVS ecosystem can be exploited using Skills for unintended purposes *without* compromising individual components as such.

However, as recent numerous reports about IoT devices with security vulnerabilities suggest, many IoT devices in the smart homes of users can be exploited relatively easily by automated security attacks like those performed by IoT malware like *Mirai* [2], *Hajime* [7] or *Persirai* [22]. Especially devices like IP cameras seem to be often affected by such attacks, as many of them can be easily exploited due to insecure security configurations like easy-to-guess administrator passwords.

3.1 Attack Overview

We developed and verified the Lyexa attack on Amazon's AVS ecosystem, as it is currently the most popular voice-controlled virtual assistant platform for smart speakers. [11] However, as virtual assistants are conceptually similar, our attack can likely be extended also to other virtual assistants. A conceptual overview of the Lyexa attack scenario is shown in Fig. 2. It involves a smart home user U, a voice-controlled virtual assistant device E like the Amazon Echo connected to the Alexa Voice Service (AVS), a malicious IoT device D and a malicious Skill S in the AVS ecosystem. The adversary is a remote attacker controlling both malicious components S and D and able to take use of a speech-to-text service or library STT, many of which are readily available.

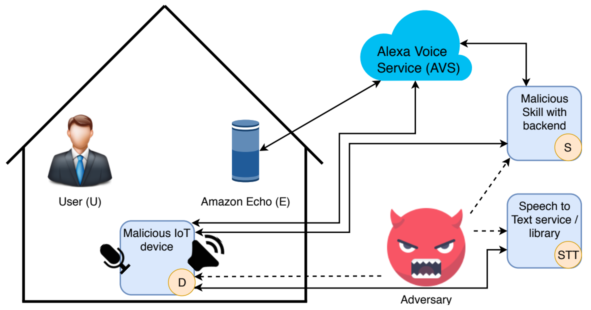

Alexa Voice Service (AVS)
Malicious Skill with backend
S
User (U)
Amazon Echo (E)
Speech to Text service / library
STT
Malicious IoT device
D
Adversary

Figure 2: Overview of the Lyexa attack, dotted lines represent a "can control" or "has access to" relation, solid lines show that objects have a connection

3.1.1 *Assumptions.*
The malicious IoT device D is located in the vicinity of E (e.g., on the same living room table) and is equipped with a microphone and a loudspeaker capable of emitting ultrasound signals. We will show in Sect. 5.2.1 that numerous different kinds of entry-level IoT devices like IP cameras are equipped with hardware that are likely to satisfy these requirements.

Device D is accompanied by a malicious Skill S under the control of the adversary. This Skill has to be activated on the victim's AVS account. Note that since it is possible to enable Skills by voice[7], it is possible that device D could use inaudible speech injection as described in Sect. 2.2 to activate Skill S by itself.

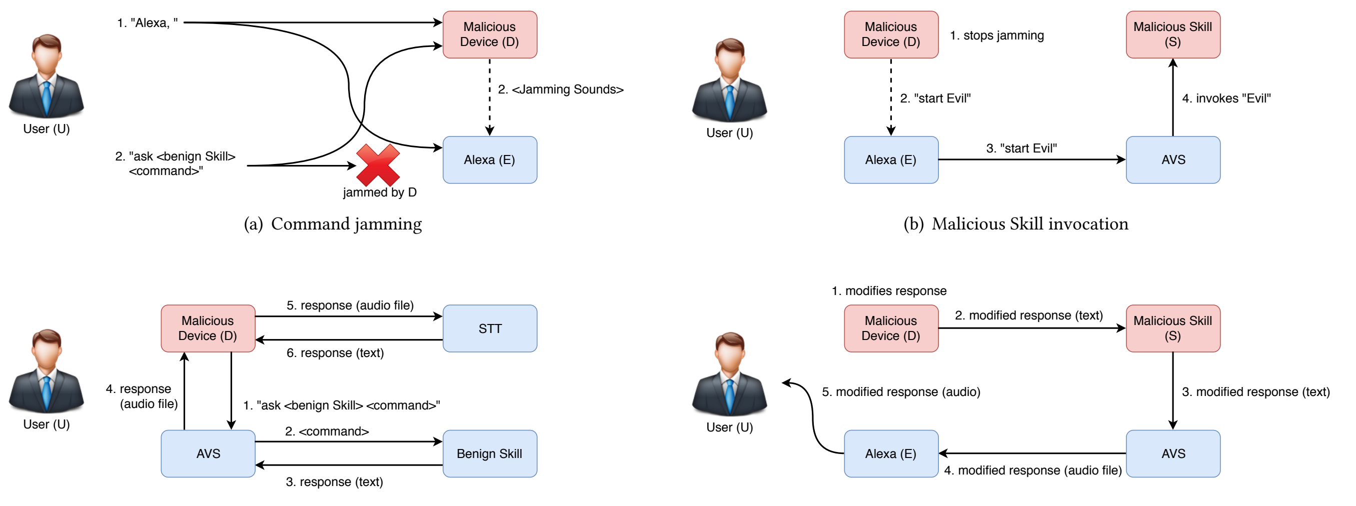
3.2 Attack Components

Our attacks consists of four distinct components, as depicted in Fig. 3 and discussed below.

3.2.1 *Command jamming.*
Figure 3(a) shows how commands issued by user U are inaudibly jammed and simultaneously recorded by malicious device D. When U speaks the wake word ("Alexa") both the benign Alexa-enabled device E and malicious device D are activated (1). Benign device E starts listening for subsequent commands and D starts jamming this command with ultrasound modulated noise, which is inaudible for humans as evaluated by Roy et al. [14] (2). This way the command issued by U can't be understood by E. D simultaneously records the command.

3.2.2 *Malicious Skill invocation.*
The second attack component is depicted in Fig. 3(b). When user U finishes speaking the command (1), D immediately stops jamming and inaudibly injects a Skill invocation command with malicious Skill S's invocation name to E which is still listening for a command (2). E will forward the injected audio command to AVS (3), which interprets it and invokes malicious Skill S (4). Malicious Skill S is now started and can return

[7]https://www.amazon.com/gp/help/customer/display.html?nodeId=201848700



(a) Command jamming

(b) Malicious Skill invocation

(c) Retrieving benign data from benign Skill

(d) Echoing modified data to user

Figure 3: Depiction of the four attack components needed for the Man-in-the-Middle attack

arbitrary text to be echoed through E in Alexa's voice. The inaudible command injection is realized by utilizing the amplitude modulation technique discussed in Sect. 2.2.

3.2.3 Retrieving benign data from benign Skill. The purpose of the third attack component is to fetch benign data from a benign Skill that user U wanted to invoke with his command. This is shown in Fig. 3(c). D first starts a new session with AVS and forwards U's recorded command to it, just like a benign Alexa-enabled device would do (1). AVS will then invoke the requested Skill and pass the command to it (2). Subsequently, the benign Skill will return a textual representation of the response to AVS (3) which will then use the Alexa TTS engine to create a voice audio file out of it and forward it to D (4). Now the malicious device D will send the audio file containing the response to a *STT* service (5) for it to be transcribed back into text (6). Now D knows what data the victim wanted to fetch from the requested Skill.

3.2.4 Echoing modified data to user. The fourth component, depicted in Fig. 3(d), shows how modified data is finally echoed to the victim user U. After arbitrarily modifying the response data received from the benign Skill in the previous step (1), D passes this text to the backend of Skill S (2). S then passes the received text to the already established AVS session (3) which will then create an audio file with the help of Alexa's TTS engine. This audio file is passed to E (4) to be played back to user U (5).

3.3 Hijacking Built-In Alexa Commands

The most straightforward attack variant aims at hijacking built-in Alexa commands like "Alexa, will it rain today?" (i.e., commands containing no Skill name, as shown in Fig. 4 segment 0). The command can be redirected to malicious Skill S by associating the utterance "will it rain today" with an intent of Skill S, and letting malicious IoT device D inject an inaudible voice command "using *Evil*" (where *Evil* is the invocation name of Skill S) when the user stops speaking to Alexa, as shown in Fig. 4 segment 1. This will cause AVS to invoke Skill S using an intent corresponding to this utterance. The malicious Skill S can then return any desired response to AVS, which will be echoed back to user U using Alexa device E, thereby making the user believe he is talking to Alexa's built-in functionality and not to Skill S.

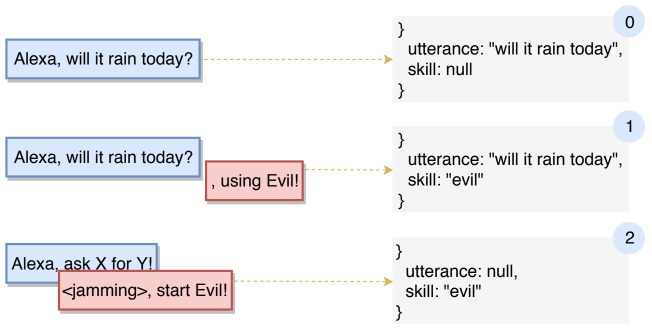

Alexa, will it rain today?
}
utterance: "will it rain today",
skill: null
}
0
Alexa, will it rain today?
, using Evil!
}
utterance: "will it rain today",
skill: "evil"
}
1
Alexa, ask X for Y!
<jamming>, start Evil!
}
utterance: null,
skill: "evil"
}
2

Figure 4: Abstract representation of how injected voice commands lead to different interpretations of the intended command. Segment 0 depicts a command activating a built-in feature, segment 1 and 2 are depicting redirections to the Skill "Evil".

3.4 Skill Redirection Attack

One drawback of the above attack is that it can't be used to redirect commands that are targeted at a particular benign Skill. For example, if the user wants to lock his smart lock *front door* with the help of the *Security* Skill, he will issue the command "Alexa, ask *Security* to lock the *front door*". Just injecting the Skill S's invocation name after the command ("using *Evil*") would not be sufficient, as the resulting command heard by E would contain the invocation name of two different Skills, resulting in AVS ignoring this command.

In order to redirect Skill invocations, we need to enhance the attack by *preventing Alexa from understanding* the genuine Skill's invocation. The obvious way to achieve this is by device D injecting inaudible noise, as depicted in Fig. 4 segment 2, at the same time the user is speaking the command, essentially jamming the user's input so that Alexa will not be able to understand it (a similar jamming attack, however not targeting voice assistants, has recently been published independently to our work by Roy *et al.* [14]).

Our evaluation of jamming of user issued commands (cf. Sect. 5.1.2) showed that to be effective, D needs to jam the user command starting immediately after the wake-word, up until the command is finished. This means D needs to start jamming even *before* it can know which command the user is going to issue. To still be able to provide meaningful responses to the user's command the malicious device D needs to independently record the command of the user while *simultaneously* jamming it, as shown in Fig. 3(a). This is feasible because different microphones are sensitive to different AM carrier wave frequencies. Device D's microphone will therefore not pick up the emitted inaudible jamming signal as it is not sensitive to the same frequency as the microphone of the targeted Alexa device. After recording the command, D uses the speech-to-text service STT to convert the recorded audio to text and forwards it to Skill S's backend. After the user command has been jammed, D executes the Malicious Skill invocation attack component (Fig. 3(b)) by inaudibly injecting the redirection command to invoke Skill S and cause AVS to start a session with it.

If the transcribed user command text returned by STT contains a Skill name or utterance the adversary wants to hijack, the malicious Skill S returns a reply to AVS to be played back to the user, as depicted in Fig. 3(d) (Echoing modified data to user component). This essentially denies the user access to this Skill while simultaneously making him think the response is coming from the genuine Skill.

However, if the user command does not contain Skill names or utterances the adversary wants to hijack, it still needs to provide a plausible response to the user in order to avoid the user realizing that he is the victim of an attack. For realizing this the adversary can utilize the benign functionality of the AVS by playing a *Skill-in-the-Middle* attack as discussed below.

3.5 Skill-in-the-Middle Attack

In this variant of our attack the adversary stages a full man-in-the-middle attack between the user and benign Alexa Skills. After performing the Command jamming and Malicious Skill invocation attack components as described above, device D will use the Retrieving benign data from benign Skill attack component as shown in Fig. 3(c) over a *separate session* with AVS to get benign data from the Skill that user U wanted to invoke. After obtaining the benign data, device D can process it and modify it in any way it wants and use the Echoing modified data to user attack component (Fig. 3(d)) to return a forged response to the user with the help of malicious Skill S. The Skill-in-the-middle attack gives the adversary thus full control over the conversation of the victim user U and any Alexa functionalities, in particular, Skills.

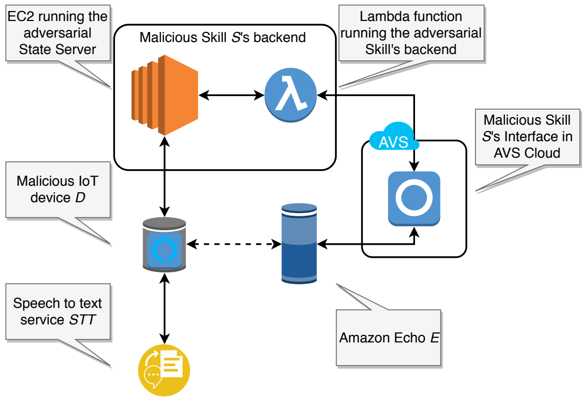

EC2 running the adversarial State Server
Malicious Skill S's backend
Lambda function running the adversarial Skill's backend
AVS
Malicious Skill S's Interface in AVS Cloud
Malicious IoT device D
Speech to text service STT
Amazon Echo E

Figure 5: Components of the attack setup

The Skill-in-the-Middle attack is particularly devastating, as an adversary can use it, e.g., to modify arbitrary responses of benign Skills. For example by manipulating the responses of a Skill providing the latest quotes on share prices it can falsify the quotes of particular companies' stock in a way that may trick the victim into making valuable financial transactions based on false information with potentially far-reaching consequences.

Note that all attack variants presented above use Alexa's genuine voice and device (which the user typically trusts) to deliver the modified or wrong information, giving the adversary the potential to substantially mislead the user or fool him to take specific actions as desired by the adversary.

4 IMPLEMENTATION

The overall design of our system is shown in Fig. 5. It consists of a malicious IoT device D and Skill S. The Skill is implemented with the help of a *lambda function* and a state server that are both under control of the adversary. Our implementation also takes use of an online speech-to-text service STT.

4.1 Malicious IoT Device

To evaluate our attack we constructed a prototype of a malicious IoT device comprising following relatively inexpensive off-the-shelf hardware components shown in Fig. 6: a Fostex FT17H wide-range tweeter speaker[8] (ca. US$ 70) connected to a YAMAHA R-S202D [9] amplifier (ca. US$ 200). With this setup we were able to produce signals in the frequency range of 500 Hz to 50 kHz. The amplifier amplifies signals with frequencies up to 100 kHz without much distortion (±3 dB), this makes it suitable for our purpose.

For recording sounds and measurements we used a *MiniDSP UMIK-1*[10] microphone. To run the software we used a *Lenovo ThinkPad T430*[11] laptop. This laptop has a sound card (and drivers) capable of generating signals with frequencies of up to 192 kHz.

[8]https://fostexinternational.com/docs/speaker_components/pdf/FT17H_rev3.pdf
[9]https://europe.yamaha.com/en/products/audio_visual/hifi_components/r-s202d/index.html
[10]https://www.minidsp.com/products/acoustic-measurement/umik-1
[11]https://www3.lenovo.com/us/en/laptops/thinkpad/t-series/t430/

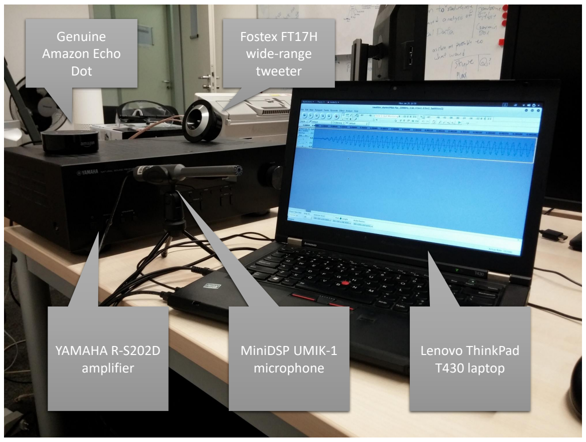

Genuine Amazon Echo Dot
Fostex FT17H wide-range tweeter
YAMAHA R-S202D amplifier
MiniDSP UMIK-1 microphone
Lenovo ThinkPad T430 laptop

Figure 6: Photo of the hardware setup mimicking the malicious IoT device. The laptop is connected to the microphone and the amplifier. The tweeter is connected to the amplifier and aligned in the direction of the Amazon Echo Dot.

The malicious IoT device's software consists of a hotword detection engine called Snowboy[12], a library for communicating with AVS and means to connect to the STT service. The Snowboy engine was modified to also be able to save recorded audio, and return the status of detection. With these modifications it is possible to start the hotword detector with the pre-trained Alexa detection model, which is shipped with Snowboy, and record audio after the hotword is detected.

4.2 Malicious Skill

The malicious Skill S consists of two components: a state server and lambda function realizing the functionality of the Skill.

4.2.1 Adversarial State Server. When the lambda function of Skill S is invoked by AVS in a Skill redirection or Skill-in-the-middle attack, it initially can't know how to correctly react to the user command, as the invocation name of the intended Skill and any command parameters are jammed by D and thus not understood by AVS. Device D needs therefore a way to notify S the hijacked Skill's name and possible command parameters. For this it uses S's RESTful state server, which is continuously polled by S's lambda function. Device D uses POST instructions to store the instructions of what Skill to mimic or what text to reply on the state server, from where the lambda function will receive it by polling the state server. Polling of the state server is performed by the lambda function using GET requests once every second.

4.2.2 Adversarial Skill's Backend Lambda Function. Skill S's lambda function is implemented using *alexa-sdk* and *Node.js* on *AWS Lambda*[13], which is a fee-based Function as a Service (FaaS). AWS Lambda is the platform recommended by Amazon for Skill backends. To get a malicious Skill certified and into the official Alexa Skill store the adversary has to create a functional and apparently benign Skill that can pass the screening performed by Amazon. This needs to be done in a way that the Skill still provides a hidden entry point for the adversary to trigger the malicious functionality after the Skill has passed certification. This can be done either by directly triggering a dedicated intent for the malicious functionality, or, it can be activated when the issued command contains a specific keyword as a command parameter.

Naturally, the utterance triggering a malicious intent should be known only to the adversary, which means that it needs to be hidden from regular users. This is, however, not difficult, as there is no apparent way (as a normal user) to get a list of all utterances a Skill can understand or react to. In the certification process only three different commands have to be declared to be listed in the official interface description of a Skill in Amazon's Skill repository. Any additional commands can be left 'invisible' to regular users.

All commands and intents the Skill understands are tested by Amazon during certification for their functionality, but after passing certification the adversary can change their behavior in its backend system without having to re-certify the Skill. Similarly, the adversary can change the behavior of an intent in its backend in response to specific Slot values triggering malicious functionality. Effectively, malicious Skill S will therefore have two handlers: one for its public 'benign' functionality and a second one realizing malicious functionality. The second handler is triggered by the first handler when it receives a malicious functionality-triggering intent or a specific keyword value as a command parameter.

4.2.3 Malicious Skill's Interaction Model. A Skill's Interaction Model is a mapping of the user's input to intents, which are passed to the Skill's backend. The user's input consists of utterances which can include Slots. From the available Skill types[14] we utilize so-called Custom Skills for our attack since they provide the most flexible invocation options and functionality. An example command to invoke a Custom Skill could be: "Alexa, ask *Recipes* how do I make an omelet." where *Recipes* is the invocation name and "how do I make an omelet" is an utterance for the Recipes Skill to process[15, 16].

Our malicious Skill's Interaction Model is quite simple. There are only four intents of which three provide benign functionality and are publicly advertised as well as one malicious intent that is not disclosed to users. This intent consists of a genuine looking utterance and an utterance consisting solely of two Slots for which we do not provide sample values. Normally a Slot would match exactly one word (or more with provided samples), but this utterance consisting of two times the same Slot separated by a space will match any number of words. This way Alexa can be tricked into passing S the full transcript of the user command as a Slot value and match every possible sentence the user says (as we might not know what the impersonated Skill wants the user to say).

4.2.4 Skill Certification. As stated in Sects. 2.1 and 4.2.2, malicious Skill S needs to be certified to be listed in the Alexa Skill Store so that it can be enabled on every Alexa-enabled device via companion app or voice command. Our Skill S disguises itself as a Skill for

[12]https://snowboy.kitt.ai/
[13]https://aws.amazon.com/lambda/features/
[14]https://developer.amazon.com/docs/ask-overviews/understanding-the-different-types-of-skills.html
[15]https://developer.amazon.com/docs/custom-skills/understanding-how-users-invoke-custom-skills.html
[16]https://developer.amazon.com/docs/custom-skills/understanding-custom-skills.html

retrieving statistics about YouTube channels. It is called "YouTube Statistics (Unofficial)" and it's invocation name is "you statistics". We were able to get our Skill certified and added to the Store.

The benign functionality of Skill S returns statistics about YouTube channels. It can be queried to return for a given channel name the number of subscribers, total number of views, the title of the latest activity or the title of the latest playlist. We didn't disclose the corresponding utterance for querying playlist titles, so this intent can be used for triggering the malicious functionality. In addition, if an utterance with a Slot value "evil" is passed to the skill, a state variable is set, which will cause the malicious handlers to handle all intents instead of the handlers providing the benign functionality.

We minimized the potential exposure of users to our malicious Skill functionality by disabling listening for the utterance or Slot value when no evaluation was done. In addition, the Skill was available only briefly in the Skill store, as development work was done using an unlisted development Skill. We are therefore certain that no user enabled the malicious functionality.

4.3 Attack Signal Generation

We implemented the method for shifting voice samples of utterances and the jamming sound into the ultrasonic spectrum with MATLAB. The details of the used signal modulation are presented in Appendices A and B. We used for low-pass filtering the Equiripple single-rate FIR filter[17] and we were able to automatically create ultrasonic audio files with different carrier wave frequencies from 20 kHz to 40 kHz for evaluation purposes. We encoded the audio files with the Free Lossless Audio Codec (FLAC), a lossless audio format suitable for storing also ultrasonic frequencies.

Stealthiness improvements. With our initial setup a crackling/pop sound could be heard when playback of the ultrasonic audio file was started and stopped. We removed this unwanted noise with a linear fade-in of 0.16 ms corresponding to 30 samples at a sampling rate of 192 kHz and a linear fade-out of one second (192000 samples) applied on an added one second of silence to the original audio file. As a result, the starting and ending of the ultrasound injection can't be heard as a clearly audible sound.

4.4 Putting it All Together

A user typically starts an interaction by uttering the wake-word "Alexa". When the wake-word is detected by the malicious IoT device D, it immediately starts jamming the user's subsequent command by using a pre-recorded and modulated audio file with a duration of > 7 seconds and consisting of sawtooth signal noise. While D is jamming the user command, it also records it. This is possible because D will not record its own jamming sound, as the microphone of D is not sensitive to demodulating signals with the same carrier wave frequency as the Alexa-enabled device's microphone. When the user stops speaking, D stops jamming and inaudibly injects malicious Skill S's invocation name from a pre-recorded and modulated audio file.

To handle the recorded user command D sends it to a speech-to-text Service STT which transcribes it and returns the most probable transcriptions as text. We are using the Bing Speech API[18]. If the transcription contains a command or invocation name that D wants to hijack, it sends an instruction to the state server to mimic this specific Skill and command. Otherwise D sends the recorded audio to AVS. The AVS backend will then process the command and return a JSON payload containing a Speak Directive with a binary attachment containing MP3 encoded audio. To get the response in textual form this audio has to be transcribed by STT as well. After D gets the transcription its sends it as an instruction to the state server telling S to return the transcribed audio text via AVS to the user. If the genuine Skill expects a response from the user, the Skill's reply will prompt the user to provide this response. In this case D will proceed to recording the user's response, send it response to STT for transcription, forward this to the genuine Skill through AVS and, receive the response, transcribe it again through STT and finally send the transcription as an instruction to S for playback to the user. In this way the adversary is are able to conduct a complete MITM attack on the user's genuine Alexa-enabled device and the requested genuine Skill.

5 EVALUATION

The evaluation of our attack was carried out in an isolated office room measuring about 4 times 6 meters. The background noise was around 50 dB. The ultrasonic speaker was placed about 30 cm above an Amazon Echo Dot, our Alexa device, to avoid coupling effects.

5.1 Attack Components' Performance

5.1.1 Adversarial Skill invocation. First we needed to find the carrier wave frequency on which Amazon Echo Dot demodulates the strongest signal. We injected the command "start *Evil*", where *Evil* is our uncertified Skill's invocation name. Our evaluation in App. C shows that using carrier wave frequencies from 22 kHz to 23 kHz provides the best performance in terms that AVS is able to optimally understand the injected voice command.

5.1.2 What to jam. To test how AVS and the Alexa-enabled device (e.g. Echo Dot) react to jamming, we used white noise to jam parts of a command in order to find the best jamming strategy.

Setup. To evaluate what we have to jam and how AVS reacts to it we used the (syntactically incorrect) command: "Alexa, Ask X for Y, ask Evil for Z", with the invocation names X and Evil and utterances Y and Z. For testing purposes we used audible white noise and added it to parts of the command audio sample recorded from a male person. We experimented with jamming different parts of the command in order to see which parts of the command needed to be jammed to make AVS not understand the benign command anymore and instead understand our injected command. The results are shown in Tab. 1.

Jamming benign Skill name. Jamming the benign Skill's invocation name "X", resulted in the effective command "Alexa, Ask for Y, ask Evil for Z". AVS started "Z" unreliably (in $< 50\%$ of trials) if it was also a built-in functionality like "tell me a joke". If "Z" was only an utterance for "Evil" we were not even able to make AVS launch "Evil" with the utterance "Z".

[17] https://www.mathworks.com/help/signal/ref/equiripple.html

[18] https://azure.microsoft.com/en-us/services/cognitive-services/speech/

Table 1: Results of jamming different parts of the command.

Part to jam	Result
Only X	Alexa starts unreliably Z if it is also a built-in function
Ask and first part of X	Alexa starts unreliably Evil with Z
Ask X for Y	Alexa starts reliably Evil with Z

Command: “Alexa, Ask X for Y, ask Evil for Z”

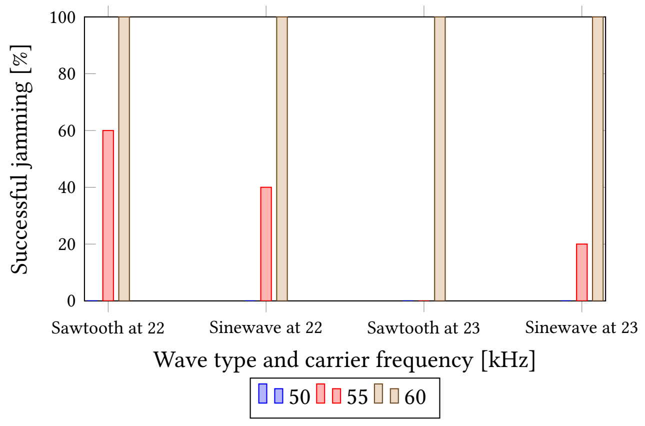


Figure 7: Results of jamming the voice sample “tell me a joke” recorded from a male person, arriving with an air pressure of 73 dB **at the Amazon Echo Dot, with a sawtooth and sine wave signal at** 1000 Hz**, modulated with different carrier waves, from** 30 cm **distance, at different volume levels (50-60/100)**

Jamming Ask and first part of X. We only jammed “Ask” and first part of “X” leaving the recognizable part of the command as “for Y, ask Evil for Z”. If the added noise was loud enough so that AVS was not able to understand the jammed part correctly it interprets it as “ask Evil for Z” unreliably (in $< 50\%$ of trials). This shows that the leftover utterance “for Y” is still confusing AVS.

Jamming the whole benign command. Finally we added noise to the complete benign command “Ask X for Y”. AVS was now able to understand “ask Evil for Z” reliably, and therefore started our malicious Skill (Evil) with the utterance “Z”.

Result. The results show that jamming has to start immediately after the wake-word and last just until the inaudible command injection starts (i.e., the entire benign command has to be jammed). This means that jamming needs to start *before* the attacker knows which command the user will issue, otherwise Alexa will recognize the genuine Skill name and will ignore the redirection to the malicious Skill S or will be confused by those parts of the benign command that have not been jammed.

5.1.3 User command jamming. To evaluate command jamming, we recorded a human male saying “tell me a joke”. This command starts the built-in functionality of Alexa to tell a joke and therefore should be easy for Alexa to understand correctly, because it uses no arbitrary Skill names. This sample was played with a sound pressure of 73 dB at the Echo Dot. We modulated a 1000 Hz sawtooth and a 1000 Hz sine wave signal with a carrier wave with a frequency of 22 kHz and 23 kHz to induce a noise signal in Echo Dot’s microphone. The results of jamming with 10 trials each are shown in Fig. 7. At volume level 60 all signal types are able to jam the voice sample successfully. The sawtooth signal was barely audible whereas the sine wave-based signal was even less audible.

5.1.4 Attack performance. We evaluated injection and jamming at a distance of 30 cm. The greater the distance between D and E, the higher sound pressure [14, 25] is required. With our setup the maximum distance for reliable injection and jamming we could achieve was 140 cm between the ultrasonic speaker and Echo Dot with a success rate of 100% at a volume level of 70/100 for injection and 60% for jamming with 10 trials each.

We evaluated the Skill redirection attack on 10 “top skills” of the US Alexa Skill store and two smart home security Skills “Nuki” and “Homematic IP” in an isolated room. We invoked each Skill using the 1-3 example utterances displayed on each Skill’s store page. Every Skill was invoked 10 times by two different persons. Each attack was executed as described in Sect. 4.4, i.e., by jamming the user command after the wake-word and injecting a command to invoke the malicious Skill after the user stopped speaking. This results in an attack success rate of 75 %. Tab. 2 shows the percentage of failure for each module (jamming, injection and STT). The most prevalent reason for the attack to fail was due to a failure in the injection of the malicious Skill’s invocation name. We observed that even when the recorded audio of the injected invocation-name was very good (the audio captured by AVS can be inspected through the “history” tab in the Alexa Apps settings), AVS still sometimes misheard the command, resulting in a failure to start the malicious skill. Jamming and speech-to-text translation (STT) of the command were almost always successful. An asterisk next to the STT value in Tab. 2 indicates minor transcription errors (e.g. “new key” instead of “Nuki”, or “jurassic park” instead of “jurassic bark”) which were systematic and can therefore easily be corrected by malicious device D using an extended keyword list to capture desired utterances. They were thus not counted as STT module failures.

We also tested how accurately Skills are triggered without adversarial influence by invoking each Skill ten times, as described above. In the benign setting the requested Skill was successfully started with the correct command in 87 % of the trials. An invocation trial was considered successful, if the command was correctly transcribed by AVS as displayed by the “history” tab in the Alexa app and the correct Skill was started. As can be seen, the attack deteriorates the accuracy of Alexa’s responses from a user point of view, but only slightly (from 87 % to 75 %). It is therefore questionable whether a user would be able to notice the presence of attacks only based on the changed behavior of AVS.

To evaluate whether different room layouts or interiors affect the attack we carried out tests in a conference room, different office rooms with up to 6 persons in it and even a crowded exhibition hall. We found no degradation of the attack success rate based on these factors, however, a large enough background noise pressure like in the crowded exhibition hall can cause our attack to fail (when the injected command sound level is less than the background noise

Table 2: Results of attacking the 10 "top skills" from the US Skill store (Nov. 2018) and two smart home security Skills, using example commands (1-3) listed on the store page

Skill Name	Jamming	Injection	STT
Flash Briefing Skills			
Reuters TV (U.S.) Fox News NPR News Now	7 %	10 %	0 %*
Custom Skills			
Jeopardy!	5 %	0 %	15 %
Find My Phone	0 %	22 %	6 %
Jurassic Bark	0 %	22 %	0 %*
Question of the Day	0 %	15 %	0 %*
Sleep Sounds: Ocean Sounds	5 %	20 %	0 %
Sleep Sounds: White Noise	25 %	25 %	0 %*
Sleep and Relaxation Sounds	0 %	11 %	6 %
Nuki	5 %	20 %	0 %*
Smart Home Skills			
Homematic IP	0 %	15 %	5 %*

level) but such noise levels are not common in private homes where Alexa devices are typically deployed.

5.2 Proof-of-Concept Attack

We implemented two proof-of-concept implementations of our attack for two Alexa-controlled smart-home devices, the Homematic IP Security System [19] and the Nuki smart lock[20]. Both security Skills can be activated or locked through Alexa. The goal of the attack is to prevent the user from locking his smart lock or arming of the smart home security system while simultaneously leaving him believing Alexa has armed or locked it properly. The Homematic IP Security System uses a Smart Home Skill for arming the home security system. When the Skill-redirection attack is run and the user says "Turn on absence mode", the command is redirected to the malicious Skill which will return a reply that is identical with that of the benign Skill, requesting further information on what device is to be armed: "Sorry what device?". The user can reply to this anything, the malicious Skill will always just return "Okay." without notifying or contacting the genuine Skill at any stage. The result of the attack is that the security system remains disarmed, even though the user thinks it is armed, because the exact same wording is used by the malicious Skill than what the genuine Skill would use. The Nuki Smart Lock uses a Custom Skill. If a user wants to close a lock and says "Ask Nuki to lock *front door*.", where *front door* is the name of the lock to be locked, the malicious Skill will directly return "Smart lock front door is locking.", again using a wording that is identical with the benign Skill's reply.

[19]https://www.homematic-ip.com/
[20]https://nuki.io/en/smart-lock/

In a Skill-in-the-middle scenario if the user wants to hear recent stock prices for a company using a popular stock market Skill called *Stock Prices by Opening Bell*[21] and asks "Ask *Opening Bell* for the price of Amazon", malicious device D forwards this command to Alexa's backend, and gets a response which is formed like "Amazon is trading at 1599.01. Down 1.97%.". Device D can modify these values at will and return this forged response with the help of malicious Skill S. Again the adversary used the exact wording of the genuine Skill to make the user believe the provided information is real because it comes from his trusted stock Skill.

5.2.1 Feasibility on IoT Hardware. To evaluate the susceptibility of smart home devices to the Lyexa attack, we examined its feasiblity given the typical hardware set-up of IoT devices. The used malicious device could be a cheap two-way security camera like the Apexis J011-WS. The J011-WS is equipped with a generic 1 W, 8 Ω small loudspeaker and the Cirrus WM8731 driver[22], which is able to process PCM audio with a sampling rate of up to 96 kHz. We tested the speaker and were able to inaudibly inject commands into the Echo Dot from a distance of up to ≈ 30 cm and inaudibly jam commands from up to ≈ 15 cm, with an AM carrier frequency of up to 25 kHz even though the speaker is not rated for frequencies beyond 20 kHz (this is the maximum we could safely achieve without overheating the speaker, for a one time attack, even more power could be used to extend the range of attack). We also tested a 3 W and 2 W generic small form factor non-tweeter speaker sold for less than $5. We were able to successfully perform command injection at a distance of up to 45 cm and 40 cm and jam at up to 30 cm and 20 cm, respectively. This shows that no special hardware is needed to accomplish this attack and even cheap off the shelf IoT devices can be used.

Cheap IP cameras, including devices from Apexis, are reported to be vulnerable to many attacks. This (generic) IP camera family is branded by bigger companies like Logilink, which sells the model WC0030A[23]. Remote code execution vulnerabilities for 1250 different camera models, including brandings of this generic camera model (Apexis, Logilink, etc.) were published[24]. In addition to that our model automatically gets a public hostname on power on which also suffers from an enumeration vulnerability.

All this combined could enable an attacker to enumerate devices, access them remotely and upload a malicious firmware or ultrasonic modulated audio files to the devices to attack Alexa devices in vicinity. This shows that many devices, even without special hardware like a tweeter speaker may be used remotely by an attacker to emit ultrasonic frequencies and attack other devices.

6 RELATED WORK

To the best of our knowledge there are no published man-in-the-middle attacks against Virtual Personal Assistants like Amazon Alexa, especially attacks utilizing service plug-ins like Skills. We are also not aware of approaches combining both jamming and injecting of inaudible commands to implement more complex attacks.

[21]https://www.amazon.com/Stock-Prices-by-Opening-Bell/dp/B01E9Z3UVC
[22]https://www.cirrus.com/products/wm8731/
[23]http://www.logilink.com/WLAN_Indoor_Pan-Tilt_IP_Kamera_mit_Nachtsicht-Bewegungsmelder_2-Way_Audio.htm
[24]https://seclists.org/fulldisclosure/2017/Mar/23

We focus therefore on reviewing existing approaches that aim at issuing commands to Virtual Private Assistants without victim users noticing.

Audible Synthesized Voice Commands. Diao *et al.* [6] discuss attacks against the Google Voice Search (GVS) app on Android smart phones. To carry out these attacks, a malicious app must be installed on the device. The malicious app activates GVS and simultaneously plays a recorded or synthesized command over the built-in speakers which is then picked up by the microphone, on which GVS is listening. This way the malicious app can access restricted resources by using GVS without asking the user for permission. The app can also call the attacker through GVS, enabling the attacker to issue commands via the call. This attack, however, does not work anymore on Android 7, as it uses now voice cancellation techniques to improve voice calls by reducing recorded noise. Alepis *et al.* [1] therefore extend the work of Diao *et al.* [6] by proposing to use multiple devices to perform the attack and to use Firebase to return transcribed audio replies.

These attacks only work on Android smart phones and are limited to injecting commands when the victim user is not listening. It is therefore not applicable in our scenario. Nevertheless we utilize the idea of using synthesized audio to control Virtual Private Assistants and that the (synthesized speech) answer can be reliably transcribed back into text, essentially allowing us to flexibly interact with a voice-controlled assistant via text.

Audible Mangled Voice Commands. Vaidya *et al.* [21] propose a method to mangle human voice so that it is no longer human comprehensible but still recognizable by the Voice Recognition Systems. The audio mangler takes raw voice and MFCC parameters as input and returns mangled voice which is hard for humans to understand but Voice Recognition Systems detect the same text in this mangled audio as in the raw audio.

Carlini *et al.* [3] extended the work of Vaidya *et al.* [21]. In addition to the black box approach Carlini *et al.* evaluate a white box approach against the open-source CMU Sphinx where the underlying mechanics are known to the attacker. Because the inner workings of CMU Sphinx are known, a more precise attack is possible. Our attack does not use command mangling for injecting or jamming because if the attack is repeated several times this would raise the suspicions of the victim as the mangled sounds would always need to occur at the same time he interacts with Alexa.

Inaudible Voice Commands. Song *et al.* [19] describe an attack using the non-linearity of microphones to induce low frequency sounds with the use of ultrasonic frequencies in microphones of voice-controlled virtual assistants (cf. Sect 2.2). Independently Zhang *et al.* [25] evaluated the demodulation properties of MEMS and ECM microphones for a single modulated frequency and voice, which consists of multiple frequencies. We adopted techniques in these papers for realizing command injection performed by malicious device D and combining it with inaudible command jamming to realize a complete attack framework against the Alexa virtual assistant. In contrast to their attack that is limited to simple command injection, our attack framework allows to hijack the entire interaction between user and Alexa or Skills allowing for much more powerful attack scenarios than mere issuing of single commands.

Independently to our work Roy *et al.* [14] have developed an approach for realizing jamming of spy microphones using ultrasound signals while not interfering with human conversations. Our method builds on similar findings than what also they report.

Voice Commands over the headset jack. Kasmi *et al.* [10] were able to induce voice into a headset connected to a smart phone using intentional electromagnetic interference (IEMI) and control the listening Virtual Personal Voice Assistant this way. Another attack requiring physical access to devices is described by Young *et al.* [23]. The attack uses a RaspberryPi 2 with an audio board, which is connected to the victim's device over the headphone jack to activate various functions on the smartphone. However, since the typical virtual assistants targeted by our attack are not equipped with headphone jacks, we did not consider these approaches in constructing our attack framework.

Skill Squatting. Kumar et al. [12] identified utterances that the Alexa STT engine misinterprets systematically and created an attack, dubbed *Skill Squatting* utilizing these misinterpretations to trick the user to open a malicious Skill. Zhang et al. [26] attack uses Skills with a similar pronounced or paraphrased invocation-name to hijack the command meant for that Skill. They do however not elaborate on how their attack would work in practice, as for it to be effective, e.g., for Skills requiring pairing to a device, both the benign and the malicious Skills would need to be activated on the victim's Alexa account (the malicious after the benign Skill was paired). Note that these kinds of attacks can only be used to redirect the command flow to a different Skill, but not a full man-in-the-middle attack, where communications between users and benign Skills are modified on-the-fly, like Lyexa does.

Hide commands in other audio. Schönherr et al. [16] and Yuan et al. [24] describe a method for hiding commands recognizable by deep neural network-based voice recognition algorithms but not by humans inside other audio files. Carlini et al. [4] showed how to create an audio file with a similar waveform which transcribes into a totally different sentence when processed by Mozilla's DeepSpeech.

IoT attacks. Fernandes et al. [8] did a security analysis of the smart home framework for Samsung's SmartThings which allows third parties to write apps for this service. Ho et al. [9] reviewed the security of smart home locks against physically-present attackers and relay attacks. Our attack differs from these works in that it does not attack IoT devices directly, but achieves malicious functionality through weaknesses in the AVS Skill ecosystem.

Other audio based attacks. Numerous works have used audio-based components as parts of security attacks similar to our approach. Son et al. [18] used sound played at the resonance frequency of MEMS gyroscopes to crash drones. Trippel et al. [20] used an acoustic injection attack to control the output of a MEMS gyroscope which made them able to inject fake steps into a Fitbit. Cisse et al. [5] published an approach on how to create malicious (perturbed) inputs for image classification and speech recognition. Finally, Schlegel et al. [15] created a low-permission Android app that can extract PINs and passwords from phone calls.

7 COUNTERMEASURES

Our attack is enabled by the non-linearity of microphones used in Amazon Echo and other Voice Assistant devices and loopholes in the Skill model. In this section we will discuss countermeasures against both vulnerabilities.

Non-linearity. An effective countermeasure to defend against Lyexa would be to change the hardware of microphones to not pick up ultrasonic frequencies or to implement a hardware module between amplifier and LPF to detect AM signals and silence the baseband of the demodulated signal, to suppress the demodulated voice commands. Furthermore, using machine-learning classifiers can be used to distinguish between demodulated and normally recorded audio. However, realizing such defenses in already deployed devices is not feasible. For new devices, this would also imply significant changes in device and in particular microphone designs, making it unlikely to be adopted in the near future. A feasible option could therefore be to resort to a dedicated device that would monitor the audio environment and alarm the user or AVS or even jam the injection, if it detects an abnormally high sound pressure in the ultrasonic sound spectrum. After-market solutions like this are not uncommon as similar products for securing the home network[25, 26] are emerging.

Skill model. Our attack works because Alexa continues to listen even when it hears only noise. Alexa's logic could therefore be changed to ignore commands that are issued after extremely loud noise in order to prevent jamming of commands. In addition, Alexa could audibly announce which Skill is activated. This would allow users to notice if an unintended Skill is invoked.

The recommended way of running a Skills backend is to use the Amazon Web Services (AWS) Lambda service which is called directly by AVS when a user invokes the Skill. Amazon could force Skill developers to use Lambda and certify also the Skills' backend source code to make sure the Skill backend doesn't contain malicious functionality. As Lambda is under control of Amazon, it could require re-certification when the source code on Lambda is changed, essentially preventing post-certification code changes. While this countermeasure may not be feasible (e.g. increased cost of certification, developers are limited in possibilities) it could be used to prevent backdoors in Skills.

8 DISCUSSION

Note that the attacks described above are not achievable by previous attack approaches that only inject commands (e.g. Dolphin [25], which records voice and replays it inaudibly) as the invoked Skill will return an audio reply which the user will be able to hear and which will make him suspicious. Without the use of a malicious Skill it is not possible to return modified or false information to the user. Skill Squatting attacks [12, 26] could in principle be used to realize our Skill redirection attack, but have a significant practical obstacle. To hijack an interaction with a specific Skill the adversary would need to bring the user to activate the malicious Skill with an invocation name that is similar to the benign one *in addition to* the already used Skill, which is unlikely.

Using our man-in-the-middle attack it is also possible to realize attacks requiring explicit entry of passwords or PIN codes, as the user can be fooled to reveal these during the interaction with the attacked virtual assistant. Previous approaches utilizing only clandestine command injection will not be able to do this, as these are not known to the adversary.

In the Alexa companion app there are two lists where a user can see the history of commands issued to Alexa, possily revealing the attack to the user. The first is the main window, where all previous 'home cards' issued by Skills are listed. Issuing home cards is, however not mandatory for Skills, so that the malicious Skill can simply omit issuing home cards for malicious functionality. The second is the "true" history of commands, which is located on a third submenu level among all other companion app settings, making it unlikely that regular users would actively monitor it.

Loud background noise or drastic changes in background noise intensity can lead to unsuccessful command injections. However, this does not impact the effectiveness of the attack in the most common deployment environments of virtual assistants, which is at home. There the environment is mostly silent, making the setting favorable for our attack.

There may finally be commands we cannot successfully hijack without the user noticing it. An example could be an interaction like "Alexa, close my last used lock!" where Alexa would respond with "Okay, your lock front door is now locked.". Without the information which lock was locked last, the adversary cannot reliably return a satisfying response to this command. Getting this information from Alexa is also not possible, as using the command would have the side effect of closing the lock.

9 FUTURE WORK

To be able to attack a victim's device reliably without a manual setup, our malicious device has to be able to set up and calibrate itself automatically, according to the physical characteristics of the targeted victim device. There are different Amazon Echo variants (Dot, Echo, Echo Plus) and two different generations of these devices. Even two identically looking devices of the same generation and version could use different microphone hardware. In addition to that, the distance between the two devices requires a signal of specific volume level to be used. If it is too low, no demodulation of the signal is possible. If it is too high, we jam the microphone by suppressing (quieter) voice by the automatic gain control [14], instead of injecting voice. To adjust the correct volume level it is possible to play, e.g., the Alexa wake-word at different volumes and modulated with different carrier frequencies until the recognition audio cue is played by the Echo device, indicating that the injection is working.

Even though we eliminated the crackling sound when an ultrasonic audio file starts playing or ends, we still get a slightly audible crack if we stop playing it halfway through. This happens when the jamming audio stops abruptly because the injection audio has to be played. We could use a pure tweeter speaker instead of a wide-range one, to get rid of the crackling sound. A software approach (a one second fade out) could be too long and Alexa might stop listening.

We could use also modify the attack to use Alexa itself as the STT service. If we have an utterance consisting solely of slots (which

[25]https://www.bitdefender.com/box/
[26]https://www.f-secure.com/en_US/web/home_us/sense

are able to catch everything) and a user says e.g. "Alexa, ask Nuki to lock garden door" we could jam only "ask Nuki" and append ", using Evil". "to lock garden door" will then be transcribed by Alexa and passed to our Skill's backend as the Slot value. Drawbacks are that we don't know exactly which Skill the user wanted to invoke (we could guess it from the rest of the command) and would need a very precise way to jam only the correct part of the command. In this case we wouldn't need to record the user's command and transcribe it using a third party service.

Currently we are looking into Google Assistant and its Actions. Samsung's Bixby and Apple's Siri have at the time of writing no comparable add-on functionality like Alexa.

10 SUMMARY

We presented and evaluated the Lyin' Alexa attack. We showed that with this attack it is possible to completely hijack a full conversation between Alexa and the user while simultaneously maintaining the interaction semantics from the point of view of the user. We were able to deny access to Skills or built-in functionality of Alexa and return false but plausible looking data to the user in the name of Alexa. Furthermore we suggest countermeasures and limitations of this attack as well as possible modifications as future work.

Acknowledgements. This work was founded in part by the German Research Foundation (DFG) within CRC 1119 CROSSING (S2 and P3) and the Intel Collaborative Institute for Collaborative Autonomous and Resilient Systems (ICRICARS). We would also like to thank Cisco Systems, Inc. for their support of this work.

11 RESPONSIBLE DISCLOSURE

We have contacted Amazon and notified them about our findings. We are currently engaged in discussions with their security team.

REFERENCES

[1] E. Alepis and C. Patsakis. 2017. Monkey Says, Monkey Does: Security and Privacy on Voice Assistants. *IEEE Access* 5 (2017), 17841–17851. https://doi.org/10.1109/ACCESS.2017.2747626

[2] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the Mirai Botnet. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1093–1110.

[3] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden Voice Commands. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 513–530. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/carlini

[4] Nicholas Carlini and David Wagner. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. *arXiv preprint arXiv:1801.01944* (2018).

[5] Moustapha M Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. 2017. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In *Advances in Neural Information Processing Systems*. 6977–6987.

[6] Wenrui Diao, Xiangyu Liu, Zhe Zhou, and Kehuan Zhang. 2014. 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 &#38; Mobile Devices (SPSM '14)*. ACM, New York, NY, USA, 63–74. https://doi.org/10.1145/2666620.2666623

[7] Sam Edwards and Ioannis Profetis. 2016. *Hajime: Analysis of a decentralized internet worm for IoT devices.* Technical Report. Rapidity Networks.

[8] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security analysis of emerging smart home applications. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 636–654.

[9] Grant Ho, Derek Leung, Pratyush Mishra, Ashkan Hosseini, Dawn Song, and David Wagner. 2016. Smart Locks: Lessons for Securing Commodity Internet of Things Devices. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIA CCS '16)*. ACM, New York, NY, USA, 461–472. https://doi.org/10.1145/2897845.2897886

[10] C. Kasmi and J. Lopes Esteves. 2015. IEMI Threats for Information Security: Remote Command Injection on Modern Smartphones. *IEEE Transactions on Electromagnetic Compatibility* 57, 6 (Dec 2015), 1752–1755. https://doi.org/10.1109/TEMC.2015.2463089

[11] John Koetsier. 2018. Amazon Echo, Google Home Installed Base Hits 50 Million; Apple Has 6% Market Share, Report Says. https://www.forbes.com/sites/johnkoetsier/2018/08/02/amazon-echo-google-home-installed-base-hits-50-million-apple-has-6-market-share-report-says.

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

[13] Nirupam Roy, Haitham Hassanieh, and Romit Roy Choudhury. 2018. BackDoor: Sounds That a Microphone Can Record, but That Humans Can'T Hear. *GetMobile: Mobile Comp. and Comm.* 21, 4 (Feb. 2018), 25–29. https://doi.org/10.1145/3191789.3191799

[14] Nirupam Roy, Haitham Hassanieh, and Romit Roy Choudhury. 2017. BackDoor: Making Microphones Hear Inaudible Sounds. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '17)*. ACM, New York, NY, USA, 2–14. https://doi.org/10.1145/3081333.3081366

[15] Roman Schlegel, Kehuan Zhang, Xiao-yong Zhou, Mehool Intwala, Apu Kapadia, and XiaoFeng Wang. 2011. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011.* http://www.isoc.org/isoc/conferences/ndss/11/pdf/1_1.pdf

[16] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. 2018. Adversarial Attacks Against Automatic Speech Recognition Systems via Psychoacoustic Hiding. *arXiv preprint arXiv:1808.05665* (2018).

[17] Claude Elwood Shannon. 1949. Communication in the presence of noise. *Proceedings of the IRE* 37, 1 (1949), 10–21.

[18] Yun Mok Son, Ho Cheol Shin, Dong Kwan Kim, Young Seok Park, Ju Hwan Noh, Ki Bum Choi, Jung Woo Choi, and Yong Dae Kim. 2015. Rocking drones with intentional sound noise on gyroscopic sensors. In *24th USENIX Security symposium.* USENIX Association.

[19] Liwei Song and Prateek Mittal. 2017. Inaudible Voice Commands. *CoRR* abs/1708.07238 (2017). arXiv:1708.07238 http://arxiv.org/abs/1708.07238

[20] Timothy Trippel, Ofir Weisse, Wenyuan Xu, Peter Honeyman, and Kevin Fu. 2017. WALNUT: Waging doubt on the integrity of MEMS accelerometers with acoustic injection attacks. In *Security and Privacy (EuroS&P), 2017 IEEE European Symposium on.* IEEE, 3–18.

[21] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. 2015. Cocaine Noodles: Exploiting the Gap between Human and Machine Speech Recognition. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*. USENIX Association, Washington, D.C. https://www.usenix.org/conference/woot15/workshop-program/presentation/vaidya

[22] Tim Yeh, Dove Chiu, and Kenney Lu. [n. d.]. Persirai: New Internet of Things (IoT) Botnet Targets IP Cameras. https://blog.trendmicro.com/trendlabs-security-intelligence/persirai-new-internet-things-iot-botnet-targets-ip-cameras/.

[23] Park Joon Young, Jo Hyo Jin, Samuel Woo, and Dong Hoon Lee. 2016. BadVoice: Soundless voice-control replay attack on modern smartphones. In *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*. 882–887. https://doi.org/10.1109/ICUFN.2016.7537163

[24] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiaofeng Wang, and Carl A Gunter. 2018. CommanderSong: A Systematic Approach for Practical Adversarial Voice Recognition. *arXiv preprint arXiv:1801.08535* (2018).

[25] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. DolphinAttack: Inaudible Voice Commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. ACM, New York, NY, USA, 103–117. https://doi.org/10.1145/3133956.3134052

[26] Nan Zhang, Xianghang Mi, Xuan Feng, XiaoFeng Wang, Yuan Tian, and Feng Qian. 2018. Understanding and Mitigating the Security Risks of Voice-Controlled Third-Party Skills on Amazon Alexa and Google Home. *arXiv preprint arXiv:1805.01525* (2018).

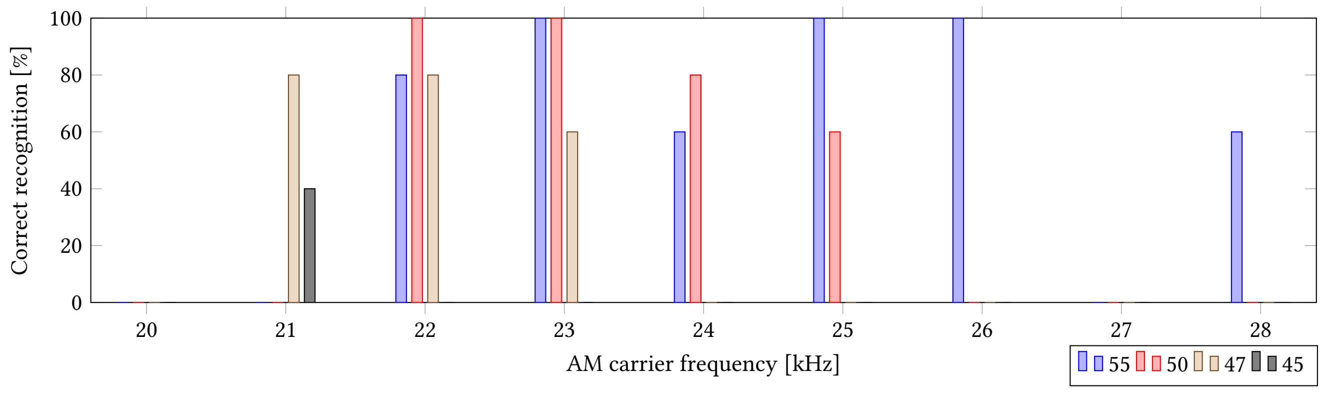


Figure 8: Results of injecting the voice sample “start Evil” recorded from a male human, modulated with different carrier waves and at different volumes (45-55/100), into an Amazon Echo Dot from 30 cm distance.

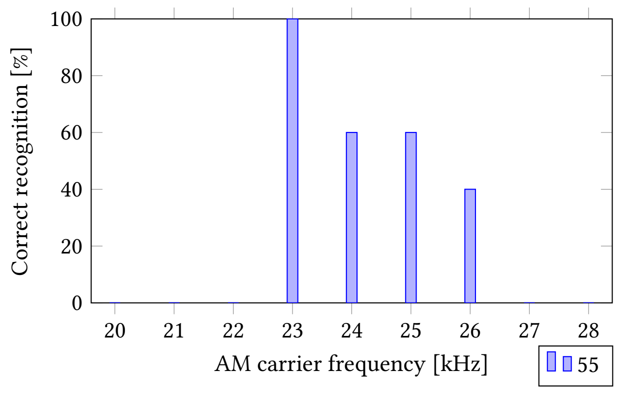


Figure 9: Results of injecting the voice sample “start Evil” generated by a female voice TTS service, modulated with different carrier waves, into an Amazon Echo Dot from 30 cm distance.

APPENDIX

A CRAFTING INAUDIBLE VOICE SIGNALS

The non-linearity of the transducer (the system consisting of a microphone, amplifier, LPF and analog-digital converter (ADC)) will output a signal consisting of more terms as the input. If S_{in} is the input signal transported over the air (voice, music, single frequency e.g. $cos(2\pi ft)$), the recorded signal, due to the non-linearity, would be:

$$S_{out} = \sum_{\infty}^{i=1} G_i S_{in}^i = G_1 S_{in} + G_2 S_{in}^2 + \cdots \tag{1}$$

where G_i is the gain for each term ($G_i S_{in}^i$), decreasing rapidly with increasing i. The attack uses the demodulation properties of the second term (the most powerful except the linear term). First we need to low-pass filter our audio signal at $8kHz$. Voice is mainly concentrated on the lower frequencies and therefore we can remove all frequencies above $8kHz$. This way our frequency spectrum reaches from $0 - 8kHz$. After that we need to upsample it to be able to convert it to ultrasound ($> 20kHz$) later on. The upsampling frequency should be the maximum frequency our device can process but no less than $56kHz$. We denote this signal as S_{up}.

The next step is to amplitude modulate it onto an ultrasonic carrier wave with frequency f_c to shift the signal into the ultrasonic spectrum and make it inaudible for humans. We get:

$$S_{modu} = n_1 S_{up} cos(2\pi f_c t) \tag{2}$$

where n_1 is a normalization factor. This will create sidebands ranging from $f_c - 8kHz$ to $f_c + 8kHz$. This is why we need at least an upsampling frequency of $58kHz$. Given the Nyquist-Shannon sampling theorem [17] we are able to sample a signal with a frequency of up to $28kHz$ without loss using a sampling rate of 56kHz. With the lower sideband ranging down to $f_c - 8kHz$ f_c should be at least $28kHz$ to make it inaudible. After that we need to add the carrier wave to the signal, so it can be demodulated by the non-linearity of the speaker. We get:

$$S_{added} = n_2(S_{modu} + cos(2\pi f_c t)) \tag{3}$$

where n_2 is another normalization factor.

Suppose $S_m = cos(2\pi f_m t)$ then the attack signal would be:

$$S_{in} = n_2(n_1 cos(2\pi f_m t) cos(2\pi f_c t) + cos(2\pi f_c t)) \tag{4}$$

After demodulation we get the frequency components f_m, $2(f_c - f_m)$, $2(f_c + f_m)$, $2f_c$, $2f_m$, $2f_c - f_m$ and $2f_c + f_m$. Every component is above $20kHz$ and will be removed by the LPF of the microphone except f_m which is $8kHz$ at the maximum. That way the original S_m is perceived by the microphone without ever played and thus inaudible for humans.

B ATTACK SIGNAL PARAMETERS

The ultrasonic attack description of Song *et al.* [19] and Zhang *et al.* [25] leaves many implementation and parameterization questions open. To craft the ultrasonic signal, we have to normalize it with the variables n_1 and n_2, as described in Sect. A. Obviously we cannot set them to 1 otherwise we will get our signal clipped at the maximum amplitude of 1. We have to scale one or both amplitudes down. Fortunately we don't have to evaluate it but we can calculate it easily as follows:

Applying Eq. 4 to Eq. 1 to calculate the second term of S_{out} which we denominate S_{out_2}, we get:

$$\begin{aligned} S_{out_2} = G_2 \frac{1}{8}(n_2^2 n_1^2 cos(4\pi f_c t - 4\pi f_m t) \\ + n_2^2 n_1^2 cos(4\pi f_c t + 4\pi f_m t) + 2n_2^2 n_1^2 cos(4\pi f_c t) \\ + 2n_2^2 n_1^2 cos(4\pi f_m t) + 2n_2^2 n_1^2 + 4n_2^2 n_1 cos(4\pi f_c t - 2\pi f_m t) \\ + 4n_2^2 n_1 cos(4\pi f_c t + 2\pi f_m t) + 8n_2^2 n_1 cos(2\pi f_m t) \\ + 4n_2^2 cos(4\pi f_c t) + 4n_2^2) \end{aligned} \quad (5)$$

With this we we see that if we want to maximize the amplitude of the demodulated S_m we have to maximize $n_2^2 n_1$ while the maximum possible amplitude of S_{in} is 1 to avoid clipping of the constructed signal. Suppose S_m has an amplitude of 1 like the carrier wave and the added wave. With Eq. 3 we get that the amplitude of S_{in} is $n_2(n_1 + 1)$. We have to find n_1 and n_2 of:

$$max \left\{ n_1 n_2^2 | (n_1 + 1) n_2 = 1 \right\} \quad (6)$$

which is at $(n_1, n_2) = (1, \frac{1}{2})$.

C EVALUATION OF PARAMETERS

The Echo Dot was activated by unmuting it by pressing the button on top of the device, after which the modulated recording of a male person saying "start Evil" was played back at different volumes and modulated with different carrier waves of different frequencies. The results are displayed in Fig. 8. We see that lower sound pressure signals modulated with a lower frequency are successfully demodulated. Unsuccessful injections usually mean that Alexa does not understand anything, but it occasionally also understood "Even", "Not", "Not you" and only "Evil", as reported by the Alexa App. As stated earlier this can be mitigated by using an invocation word composed of two words which do not occur anywhere else. Evaluating with the same setup but using a voice sample generated by a female TTS service is depicted in Fig. 9. Song *et al.* [19] came to the same conclusion, that this yields much worse results, as the female voice is centered much higher than that of a male and is clipped by the low-pass filter.