Analysing and Preventing Self-Issued Voice Commands

Sergio Esposito

Information Security Group
Royal Holloway, University of London

This dissertation is submitted for the degree of
Doctor of Philosophy

Declaration of Authorship

These doctoral studies were conducted under the supervision of Dr. Daniele Sgandurra and Dr. Daniel O'Keeffe, with the advisory support of Prof. Giampaolo Bella.

I, Sergio Esposito, hereby declare that this thesis and the work presented in it are entirely my own. Where I have consulted the work of others, this is always clearly stated. This work has not been submitted for any other degree or award in any other university or educational establishment.

Sergio Esposito

May 2023

Acknowledgements

First of all, I would like to wholeheartedly thank Dr. Daniele Sgandurra for supervising my PhD from its beginning to its end, even after leaving Royal Holloway and the UK for another job. He taught me a lot about research, about which kinds of details are important, thoroughly analysed all my work, gave me invaluable comments during every meeting, tirelessly answered my questions and comments even during the weekend. He has a lot of passion for research, and one of the reasons why I want to keep going with academia is that he was able to light a spark of that passion for research in me as well.

I would like to thank Dr. Dan O'Keeffe, for accepting to be my supervisor at RHUL after Daniele left, even if he knew little about me and I was abroad for most of my PhD. I also wish to thank Prof. Giampaolo Bella for having encouraged me in pursuing a PhD, something that I thought was *not for me*. He is always there to clarify all the many doubts I have on virtually everything, with his inerrable logic and semantics, which very often was the solution to my supposedly complex problems. We have been working together for six years now, and somehow he is still not tired of my flights of fancy, and he is always interested in what I have to say regarding computer security matters.

I would like to thank all the people in the S3Lab, including those who I was able to meet in person, those who I met virtually, and those who I did not have the pleasure to meet. A special shoutout goes to Robert Markiewicz, who helped me with the testing of adversarial command self-issue on 4th generation Echo devices in the midst of the pandemic, enduring through lots of sam-

ples with the same song in the background. Another special shoutout goes to Jordy Gennissen, whose constructive comments really helped me more than once in improving my work substantially. Additionally, I wish to thank Dr. Jassim Happa, Dr. Fabio Pierazzi and Prof. Peter Komisarczuk for accepting to be my examiners and read my work.

Thank you, from the bottom of my heart, to my wife Priscilla, who supported *in toto* my lunatic decision of pursuing a PhD when I already had an established career in the industry, and she never looked back at it. I am very grateful for having shared this journey with you, and for all the other journeys that are yet to come. Thank you to the new entry in the family, my son Elia, for reminding me of what is truly important in life. You are winning a million tiny battles every day, and you are achieving so much in so little time, and I am so proud of you. I hope you will be proud of me as well, one day.

Thank you to my mom and dad, Cetty and Natale, who always strived for guaranteeing me the best education and a bright future. None of what I achieved in my life would have been possible without you. Thank you to my sisters, Mariuccia and Manuela, who have basically been my second and third mothers, teaching me how to make my way through this pretty, dense, and prickly thicket that is life. Thank you to my grandmother, Concetta, who was able to show me, as a kid, the fascination that knowledge and clarity bear, making me want to pursue them even after so many years. Thank you to Salvo and Rosita, for being amazing parents-in-law, and having always treated me as if I were their child, actively helping me with achieving my biggest goals.

Last but not least, I would like to thank the folks I share my everyday path with — my friends. Sorry if I don't write all your names here, you know who you are. The fact that we all live on the *Planet of Cats* and that we are all part of the *Nerd Team*, walking this road together, is a pleasant thought that makes life way more enjoyable. Finally, I would like to thank Edo — the distance that has always been between us never prevented us from being great friends.

Abstract

In this thesis, I investigate to what extent self-issued voice commands can threaten Voice-Controllable Devices (VCDs) in real-world scenarios. First, I formalise the threat model for self-activation attacks: I identify six phases in it, which I outline in the so-called VOCODES Kill Chain. To verify the feasibility of self-issuing voice commands in real environments and to assess the impact on end users, I test the attack on the Amazon Echo Dot, a commercial smart speaker. In doing so, I found three vulnerabilities, which have been acknowledged and (partially) fixed by Amazon. By exploiting these vulnerabilities, I was able to confirm that an adversary can reliably self-issue commands to the Echo device, severely impacting the user's safety, security, and privacy.

In the second part of this dissertation, I review current voice spoofing countermeasures, finding that they discard all artificially generated voice commands, assuming that they are malicious — however, disabled people often use synthetic voices to interact with their VCDs. Therefore, I present a taxonomy of tolerance levels of artificial voice commands, to allow users to select the desired security/usability balance for their VCD. Level 1 of this taxonomy prescribes that self-issued commands should be discarded, while other synthesised commands can be executed. To implement this level, I propose a new solution based on a Twin Neural Network, which outperforms the current state-of-the-art anomaly detection techniques. I conclude my dissertation by showing that self-activation is an underestimated issue that can be mitigated *by-design*, and by showing possible future directions for research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today, we have the possibility to tackle most of our daily tasks with the help of technology, spanning from electronic calendars to house-cleaning robots. An increasing number of people is getting used to the feeling of being surrounded by technology, and more specifically, by the Internet of Things (IoT). At home, we can control our appliances using our voice only, via software called Voice Personal Assistant (VPA) [75] and the device in which it is embedded, usually called a *smart speaker* [21], or more generically, a Voice-Controllable Device (VCD). This device runs the VPA and has a microphone to capture user queries, along with a loudspeaker to emit related responses. With it, we can turn the heating on, prepare a coffee, automatically turn on the microwave at lunchtime, and even set up our virtual assistant to read us the news and remind us of our appointments for the day as we enter the kitchen, without even saying a word [75]. As we are going to work, our car is connected to the Internet: our personal assistant can follow us there, reading the messages we receive while driving, connecting to our smartphone to interact hands-free [138], or even recognising if we are about to fall asleep while driving on the highway because our newborn had been crying all night.

In 2021, more than 258 million *smart homes* were counted worldwide, and this number continues to grow, indicating that an increasing number of people want to experience the new frontier of IoT commodities, fascinated by

the possibility of a future where everything can be controlled using the voice channel only [145]. Smart commodities such as light bulbs, baby monitors, security cameras, TVs, ovens, and dishwashers are already available in physical and online stores, and they can all be controlled by issuing a simple voice command to one of the aforementioned smart speakers, such as Amazon Echo [14] and Google Nest [65]. For example, "Alexa, make me a coffee" or "Hey Google, turn on Roomba to clean the floor" are all valid commands that will turn on a specific smart appliance to do exactly what it is told to do.

The set of features offered by these smart speakers can also be extended through third party applications [21]: some of them, for example the Paypal skill, allow us to carry out financial transactions and seem to be among the most critical, as our Personal Identifiable Information (PII) is processed [11] along with our financial information. Even applications that process health information can be considered among the most critical ones, for example, the Vocera skill, which allows *"communication between patients and their care team members while they are in the hospital"* [156]. Unfortunately, to date, smart speakers have very little or no means of authenticating the user who is interacting with them, making the processed sensitive information available to anyone who can issue commands to the device [55]. While it is true that these sensitive features can be protected with a PIN number, it is also true that this PIN must be spoken aloud by the user when prompted to, making it lose its secrecy if someone is within earshot of the command [133].

1.1 Current Issues

The aforementioned lack of authentication and authorisation protocols lays the base for a multitude of security issues that have arisen in recent years. In fact, several attacks that undermine the safety, security, and privacy of smart speaker users have already been discovered, and most of these exploit that same voice channel that is the peculiarity of these devices, and that fas-

cinates so many users with its unlimited possibilities. Some of the attacks are designed to hinder command recognition [106], by performing a Denial of Service (DoS) attack on the targeted voice-controllable device. Most attacks, however, aim to inject commands unknowingly to the legitimate user, to allow the adversary to perform unauthorised operations on the victim device. To date, security researchers have found several methods to do this: some attacks work by injecting inaudible commands via an external ultrasonic speaker [175, 120]; other attacks hide voice commands inside an audio file, such as a song [171, 35], which is used as a carrier for the adversarial perturbations; sometimes, commands are transmitted to the voice-controllable device using an unusual vector (i.e., the command is not transmitted directly *over-the-air*), for example using light [147] or a solid carrier, such as a table, by using a Piezoelectric Transducer (PZT) to drive ultrasonic waves through the solid surface [168].

There is also a class of attacks in which adversaries make use of audible commands to activate voice-controllable devices, and they do so from the victim device itself [82, 48, 6], exploiting the so-called *command self-issue* or *self-activation* vulnerability. This attack can be launched against all devices capable of accepting voice commands and playing audio files simultaneously. It simply consists of making the device play an audio file containing a malicious voice command: the microphone embedded in the voice-controllable device will capture the command and the device will execute it as valid, as shown in Figure 1.1. One of the peculiarities of the voice command self-issue is that of eliminating the need of placing rogue equipment in proximity of the target device for the attack to succeed, a constraint that all the aforementioned attacks share. Indeed, to carry out the Denial of Service attack, the adversary must play the Adversarial Music from a rogue speaker near the target; to inject inaudible, ultrasonic commands, the adversary needs to place an ultrasonic speaker near the target device; to use light as a vector for the voice command, the adversary may actually be quite far from the target de-

vice, but they need to have a line of sight and they need nearly 400$ worth of equipment available to carry out the attack [148]; to use a solid carrier, the adversary must first place the PZT transducer on the surface they want to use for the attack [168]. Hence, it should be clear that the main advantage of self-issued voice commands is that the adversary does not need any physical equipment in proximity of the target device.

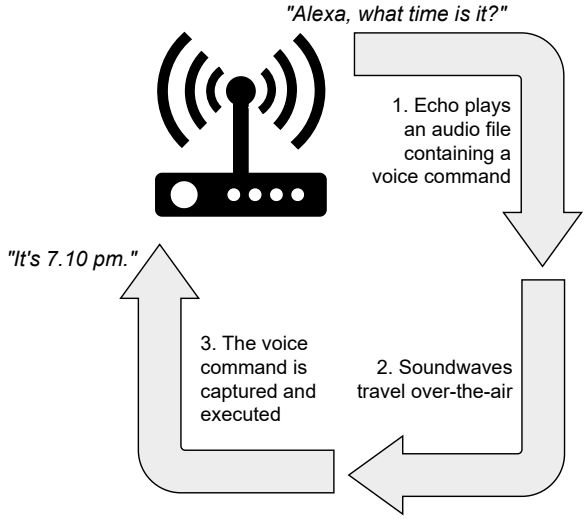

FIGURE 1.1: *Command Self-Issue on an Echo Device*

1.2 Research Overview

This thesis investigates whether the voice command self-issue is actually a threat to users and voice-controllable devices, that is, whether it is a feasible attack in real-life scenarios, and what impact it has on users' safety, security and privacy. The analysis of the self-issue attack was performed by penetration testing a common commercial smart speaker, the Amazon Echo Dot. After validating the attack, six different lab experiments were performed to understand the overall feasibility of the attack. Additionally, with the help of three voluntary households who took part in a field study, and 18 Amazon Echo users who responded to a survey, we were able to confirm that self-

activation is a real threat to users and VCDs in home environments, and its limitations are mostly theoretical.

This thesis also investigates which defensive measures can be applied to counter the attack in real-life scenarios, taking into account different categories of users and their usability and security requirements. To do so, we first analysed the existing security measures already implemented on commercial voice-controllable devices, and we developed a solution against self-activation by fine-tuning a Twin Neural Network. We trained our solution on a newly created dataset consisting of 35 samples and their augmentations, and found that it has 97% accuracy in correctly classifying samples into benign/malicious categories, and outperforms state-of-the-art anomaly detection techniques.

1.2.1 Research Goals

Through this research, we sought to contribute to the achievement of a set of Research Goals (RGs) related to self-issue attacks, namely:

- **RG1: Feasibility of the self-issue attack.** Investigate whether it is possible to exploit the self-issue vulnerability in real scenarios (Section 4.5.2) and to evaluate its limitations (Section 5.4).
- **RG2: Impact of the attack.** Understand whether being able to self-issue commands can lead to critical scenarios that are harmful to the user's safety, security, and privacy (Section 5.3).
- **RG3: Analysis of existing countermeasures.** Understand which countermeasures can users or device manufacturers apply to defend against this attack (Chapter 7).
- **RG4: Feasibility of reliable self-activation detection.** Assess whether we can develop a solution that reliably detects self-activations and that can outperform existing solutions (Section 7.5.2).

- **RG5: Easily trainable countermeasure.** Understand if said countermeasure against self-activation can be trained with a limited number of training samples (Section 7.5.1).
- **RG6: Deployability of the countermeasure.** Investigate if said countermeasure can be deployed with minimal invasiveness and investment by both the user and the provider, e.g., running along the cloud speech recognition service, or on the same VCD (Section 7.6).
- **RG7: Performance of the countermeasure.** Understand if this countermeasure can provide a low number of false positives, especially compared to state-of-art anomaly detection techniques (Section 7.5.3).

In the following section, we will describe the scientific approach we employed to verify these research goals.

1.2.2 Research Methodology

Starting from the above research goals, we followed the standard scientific method composed of questioning, researching, hypothesising, experimenting, collecting data, and drawing conclusions.

- **Questioning**: We started by formulating Research Questions (RQs):
  1. Is the self-issue vulnerability a real threat to users and VCDs in the real world, or is it just a theoretical attack?
  2. To what extent does it allow an attacker to control, in a real scenario, an Echo device without leveraging any external speaker?
  3. What impact does it have?
  4. Do existing countermeasures suffice to thwart self-activation?
  5. Is there a margin for improvement in countermeasures?
- **Research**: We conducted a literature review to assess which attacks already leveraged self-activation, and how recent were these works. For

example, for RQ1, Sections 2.5.3 and 4.8.1 review the literature with respect to voice spoofing and self-activation attacks.

- **Hypothesis**: For each RQ, we formulated some hypotheses to be verified. For instance, for RQ1, we hypothesised the following: *"If an attacker gets to play audio on an Amazon Echo Dot device, they are able to self-issue any valid command to the device"*.

- **Experiment**: We designed and conducted different types of experiments to test the hypotheses. In the hypothesis mentioned previously for RQ1, we attempted to self-issue commands to an Amazon Echo device after getting the possibility to stream audio files on it.

- **Data**: We collected and analysed data from the previous experiments. As an example, for RQ1, we collected and analysed the success rate of the self-issue attacks under various conditions and payloads.

- **Conclusion**: Based on the data collected during the experiments, surveys, field studies and other observations, we accepted or rejected the hypotheses. For example, for RQ1, the experiments demonstrated that the Echo device is susceptible to self-issue attacks, and that some environmental conditions are more favourable than others for the attack, hence, we accepted the hypothesis.

In the end, through the application of the scientific method, we have effectively addressed all research questions in a manner that is both reproducible and verifiable, ultimately enabling us to achieve all research goals.

1.3 Contributions

This thesis presents the following contributions:

- Present a kill chain that generalises self-issue attacks to VCDs, which we call the VOice-COntrollable DEvice Self-Issue (VOCODES) Kill Chain

(Chapter 3). In particular, we build this kill chain by making some key modifications to the Lockheed-Martin Kill Chain [77], and we show that it is able to describe all the required steps to perform self-issue attacks.

- Formalise a threat model for such attacks (Chapter 3), which we call the VOCODES Threat Model. In particular, we use epistemic modal logic to formalise knowledge and possible actions for all actors involved in the attack, i.e., the user and the adversary.

- Describe three vulnerabilities we have found on Amazon Echo devices: (i) the *voice command self-issue*, which enables attackers to control the device using its own speaker, (ii) the *Full Volume Vulnerability (FVV)*, which allows attackers to self-issue commands without any reduction of the device volume, and (iii) the *Break Tag Chain Vulnerability*, which allows attackers to violate the Amazon's Speech Synthesis Markup Language (SSML) policy (Chapter 4).

- Present and evaluate *Alexa versus Alexa*, an attack that exploits the command self-issue vulnerability against Amazon Echo and can chain the two other found vulnerabilities to enable the attacker to get complete and persistent control over an Amazon Echo device (Chapter 4).

- Evaluate AvA's feasibility and impact by analysing the results of a field study among three households, and of a survey administered to a study group composed of 18 Amazon Echo users, to demonstrate that the attack is feasible and that limitations remain theoretical (Chapter 5).

- Analyse security features available on most commercial VCDs available worldwide, namely, personal computers, smartphones, Amazon Echo, Google Nest and Apple HomePod. We demonstrate that countermeasures to voice spoofing attacks cannot be applied pervasively to all devices. Consequently, we also present a taxonomy of the different security levels that should be selectable by users of the device (Chapter 6).

- Describe a countermeasure against voice command self-issue attacks, based on a Twin Neural Network architecture. This approach can protect all voice-controllable devices, without the need for external sensors. We evaluate and compare the proposed countermeasure against a baseline consisting of several anomaly detection techniques, and we show that our countermeasure outperforms all of them. We also measure the performance of our countermeasure on different environments and hardware (Chapter 7).

- Present a new dataset consisting of 70 pairs of captured/played audio files. We recorded it by placing a microphone array on top of a smart speaker, using the former to record voice commands and, simultaneously, the latter to play audio. As no other dataset currently features both the played and the recorded audio of a voice-controllable device while a legitimate user (or an audio file played by the same device) is issuing a command, this dataset is crucial to assess the performance of our solution against self-issued commands and will be useful to evaluate the performance of future countermeasures against the same attack or against spoofed commands in general. For this reason, and to allow the reproducibility of our experiments, the dataset and the source code of our countermeasure against self-issued voice commands are available for download (Chapter 7).

1.4 Thesis Organisation

The remainder of this thesis is organised as follows:

In **Chapter 2**, we focus on the technologies considered for this thesis, the general classes of attacks already affecting voice-controllable devices, and related privacy issues.

In **Chapter 3**, we illustrate the various phases of real cyber-attacks using different attack models, with a particular focus on kill chains. We show that a

typical self-activation attack has some key differences from other attacks, and then model a bespoke kill chain, which we call the VOCODES Kill Chain. This allows us to generalise the procedure for performing self-issue attacks and to formalise a threat model for these attacks.

In **Chapter 4**, we introduce three vulnerabilities we found on Amazon Echo devices by applying the VOCODES Kill Chain. We describe the Alexa versus Alexa (AvA) attack, which exploits the self-activation of the device and can be chained to the other discovered vulnerabilities to enable the attacker to achieve full and persistent control of the target Echo device.

In **Chapter 5**, we evaluate the feasibility, impact, and limitations of the AvA attack in the real world with the help of three households and 18 Amazon Echo users, who took part in a survey. We show that AvA is practical, has strong implications for the end-users' safety and privacy, and that the limitations of the attack are mostly theoretical.

In **Chapter 6**, we present an analysis of common security features and weaknesses of commercial VCDs. We demonstrate that more granular control is needed when protecting such devices: currently, most commercial devices do not have any protection against spoofed commands in place, however, their introduction needs to be accompanied by the ability to customise the level of protection offered, depending on the user's needs. Hence, we present a taxonomy of possible security levels to be applied to such devices to ensure a balance between usability and security.

In **Chapter 7**, we present our countermeasure against self-issued voice commands, implementing one of the levels of the presented taxonomy. We show that our countermeasure correctly classifies samples in the malicious and benign categories 97% of the times on average, outperforming our anomaly detection baseline.

Finally, in **Chapter 8**, we revise the challenges that have been faced throughout the PhD and the most relevant findings. We then make concluding remarks and talk about possible future developments of this research.

1.5 Publications

This thesis revisits and expands on contents discussed in the following:

- Esposito S., Sgandurra D., Bella G., Alexa versus Alexa: Controlling Smart Speakers by Self-Issuing Voice Commands. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '22)*. 2022.
- Esposito S., Sgandurra D., Bella G., Protecting Voice-Controllable Devices Against Self-Issued Voice Commands. Accepted to the *8th IEEE European Symposium on Security and Privacy (Euro S&P '23)*. 2023.

Chapter 2

Background

In this chapter, we start with the fundamentals of information security, which will be useful to understand key concepts (e.g., the CIA triad, what is a vulnerability, or an exploit) throughout the whole work. After, we introduce Machine Learning and then we delve into Deep Neural Networks and build a bottom-up approach from there, to reach Voice Controllable Devices: hence, we describe how Deep Neural Networks are designed and how they work, and we use this knowledge to illustrate Natural Language Processing problems and solutions, which in turn are used to introduce Voice Controllable Devices, whose protection is the core topic of this thesis. This will allow the reader to fully comprehend how these devices work, so that the explanation of attacks and countermeasures shown in this thesis is fully comprehensible from a technological standpoint.

2.1 Information Security Fundamentals

We start by defining one the most fundamental concept in information security, that is, the CIA Triad. It is common knowledge that security of data is obtained through the achievement of three properties [126]:

- **Confidentiality**: prescribes that access to data must be given only to entities that are authorised to read them. To prove that they are entitled

to read such data, these entities usually need to prove their identity first, through a process called *authentication*.[1] An example of authentication system is the login form, where the user must type their username and password to access confidential data. The system on which the entity is authenticating knows which entities are authorised to read specific data. Hence, not all authenticated entity could be able to read some data, because they also need to be authorised for it.

- **Integrity**: prescribes that data must be protected against improper modifications. This can happen with accidental corruption or with malicious alterations performed by malicious entities. Integrity also prescribes that data must not be improperly cancelled, and while data maintain their integrity, any entity that has issued such data must not be able to claim the opposite (i.e., that the entity has not issued those data, or that data contain different information). Such last property is sometimes defined separately from Integrity, in another different property called *Non-Repudiation*. For example, calculating the hash of a file can help understanding if such file has been altered.

- **Availability**: prescribes that data should always be accessed reliably. In other words, exceptional circumstances should not hinder access to and use of data — for example, several companies implement the so-called *disaster recovery* procedures that, in case of calamities such as earthquakes, inundations, fires, etc., can be activated to timely restore access to data. For example, keeping backups in a different facility than the data center's is a common disaster recovery practice against fire hazard. Similarly, *business continuity* procedures are implemented by companies to keep core functions (such as access to data and business criti-

[1] Authorisation can happen without authentication as well, although today they are increasingly being tied together: for example, an employee who is accessing a secured area within a facility by entering a numeric passcode is confirming they are authorised to access that area, but as they give no information on their identity, they are not being authenticated. A similar thing happened in the past with prepaid public phone cards — the user was authorised to make a call if there was credit within the card, but there was no authentication system in place.

cal operations) alive during disservices. For example, having a backup power generator can help in case of blackout, and redundant network nodes can help in case of failure of a single node.

These three properties, as the reader may have noted, are called the CIA Triad because of their initials. To ensure security of information, these properties must be guaranteed for both data at rest and in transit. Data at rest are data that are stored in memory (e.g., an hard disk, on the cloud, on an USB stick), while data in transit are data that are being moved from one place to another (e.g., data that is being sent through the Internet) [78].

As the CIA Triad is the fundamental pillar of information security, it follows that if one of the properties is negated within a certain system — either a piece of hardware, a software, or any other component — there is a security problem, known as a *vulnerability*. Particular attention is due to the word *system*, as when the same security problem is discussed theoretically, not in relation to any known system, we talk about a *weakness* instead [149]. For example, a weakness would be Cross-Site Scripting (XSS) in general, while a vulnerability would be the possibility to exploit XSS within a specific version of a certain JavaScript library.

Not all vulnerabilities have the same impact, as it depends on what the adversary is allowed to do during and after the exploitation. The Common Vulnerability Scoring System (CVSS) is currently considered the state-of-the-art to measure the impact of existing vulnerabilities [61]. CVSS v3.1 takes into account eight variables to output a severity score in the 0-10 range. Vulnerabilities scored with 0 are considered to have no impact on the CIA Triad, and hence no impact on the security of the component, while vulnerabilities scored with 10 have critical impact on all three components. Factors taken in consideration for the score are:

- **Attack Vector**: whether the adversary is allowed to (i) exploit the vulnerability remotely, or (ii) on the same network, or (iii) they need logical

access, or (iv) they need physical access to the device running the vulnerable component;

- **Attack Complexity**: whether the adversary has to carefully prepare for the attack, or the attack can be performed at will;
- **Privileges Required**: whether the adversary needs to have certain privileges to perform the attack (e.g., root access, user access), or not;
- **User Interaction**: whether the user needs to (often unwittingly) perform some actions to make the attack work;
- **Scope**: whether exploiting the vulnerability allows the adversary to impact other components, instead of just the vulnerable one;
- **Confidentiality**: whether the adversary can violate the confidentiality of (i) all, (ii) some, or (iii) no data via the vulnerable component;
- **Integrity**: whether the adversary can violate the integrity of (i) all, (ii) some, or (iii) no data via the vulnerable component;
- **Availability**: whether the adversary can violate the availability of (i) all, (ii) some, or (iii) no data via the vulnerable component.

Adversaries can leverage vulnerabilities to violate the CIA Triad through scripts, programs or other resources called *exploits* [60].[2] This means that an exploit in the cybernetic domain can have different forms, depending on the attacked system: it can be a piece of code, an archive, a webpage, or even media, such as an image or an audio file. In fact, in Section 3.3 we will analyse an *audio weaponization* process, which prescribes the adversary to craft an audio file to exploit the self-activation vulnerability on VCDs.

[2] It is a common practice to use the word exploit as a verb as well. Hence, we can also say that an adversary *exploits* vulnerabilities through the already mentioned scripts, programs or resources.

2.2 Introduction to Machine Learning

Before delving into Deep Learning, which will be a core topic throughout the whole thesis, we introduce fundamental concepts of machine learning that will help understanding more complex theory regarding Deep Learning.

The most common definition of a machine learning algorithm is given by Mitchell [119] who states that *"a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks T, as measured by P, improves with experience E"*. Hence, a machine learning algorithm is made of three main components: T, P and E. The first step is usually understanding T, that is, the class of tasks we want to solve with machine learning. A few examples are [62]:

- **Classification tasks**: the algorithm is asked to assign a specific category to a certain input, among a list of predetermined categories;
- **Regression tasks**: the algorithm is asked to find a curve that approximates the given inputs, to predict its value for unseen inputs.
- **Anomaly detection tasks**: the algorithm is asked to detect atypical items within a certain list given as input.

These are only a handful of possible applications for machine learning and, therefore, deep learning. We will discuss other tasks within Sections 2.3 and 2.4. Once the task is identified, we need to understand how it will gain experience E — this usually happens by *training* the algorithm on a large set of data, called a *dataset*. There are four main ways to gain experience [62]:

- **Supervised learning**: we use a dataset made of different samples, each with their own features. Each sample is annotated, that is, there is indication of what is the correct prediction for it (e.g., the category it belongs to, or a numerical value). Such annotation is called a *label*.

- **Unsupervised learning**: we still use a dataset made of different features, but it comes with no annotations. In this case, we usually want the algorithm to learn the underlying probability distribution, or simply put, relevant connections between the samples. This is very useful in *clustering* tasks, in which the number of possible categories for the samples is not known in advance.

- **Semi-supervised learning**: we use a dataset that is annotated only in part, while the rest is not. The two parts can be used in conjunction, or the labeled part can be used to label the unlabeled part, to have a larger labeled dataset to use [180].

- **Reinforcement learning**: we make the algorithm interact with an environment in a trial-and-error process to gain experience. Good interactions (success in the task) are rewarded with positive feedbacks, while unwanted interactions (failure in the task) are punished with negative feedback. In this case, there is usually no dataset involved [62].

Finally, we want to specify a performance measure P for our algorithm. We usually distinguish between the *training performance*, which is measured on the training part of the dataset, and the *validation performance*, which is measured on the validation part of the dataset. It is important to keep these parts split, to avoid information on validation samples leaking into the algorithm's experience. In fact, as we ultimately want to measure our algorithm's performance on unseen samples (hence, we are mostly interested in measuring validation performance), if they were part of the training process, they would not be unseen, as the algorithm would already had the chance to attempt its prediction on them, and would have received feedback for it (e.g., in supervised learning). Popular performance metrics are [62]:

- **Accuracy**: defined as the part of correctly predicted samples over the

total number of samples.

$$A = \frac{Correct\ Predictions}{Total\ Samples}$$

- **Error Rate**: the opposite of accuracy. In other words, it is the part of incorrectly predicted samples over the total number of samples.

$$ER = 1 - A$$

- **Balanced Accuracy**: reflects the real performance of algorithms when dealing with unbalanced datasets. For example, let's consider an anomaly detection task, and let's assume we have a dataset consisting of 9 malicious samples and 1 benign sample. If our algorithm always predicts that a sample is malicious, without even analysing it, such algorithm would score an impressive 0.9 accuracy! Instead, balanced accuracy considers predictions for malicious samples (True Positive Rate, TPR) and predictions for benign ones (True Negative Rate, TNR) separately, and returns their average. The same algorithm that scored a whopping 0.9 accuracy scores a mediocre 0.5 balanced accuracy in that same scenario, making us realise that it is not efficient for anomaly detection.

$$BA = \frac{TPR + TNR}{2}$$

$$TPR = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$TNR = \frac{True\ Negatives}{True\ Negatives + False\ Positives}$$

- **F1 Score**: defined as the harmonic mean between Precision (P, true positives over all positive predictions) and Recall (R, i.e., TPR). Similarly to balanced accuracy, it is a better metric than accuracy when dealing with imbalanced datasets, especially if the cost of hitting false positives is dif-

ferent from the one of hitting false negatives. F1 score can also be used with multi-class classification [143].

$$P = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$F1 = \frac{2 \cdot P \cdot R}{P + R}$$

Not all performance metrics are suitable for scoring all tasks and algorithms, hence, the ultimate choice is the analyst's, who can also use multiple metrics at once to better understand how the algorithm is behaving.

Although we earlier said that what we ultimately want to measure is the validation performance of our algorithm, this does not mean that the training performance is useless. In fact, the latter allows us to understand if the algorithm is actually learning something about the training dataset, and, when compared to the validation performance, it allows us to understand how well the algorithm is generalising the task. Therefore, generally speaking, if the training performance is too low, we say that the algorithm is *underfitting* the dataset, that is, it is not able to generalise the problem well enough to actually learn something relevant. Conversely, if the training performance is sensibly higher than the validation performance, we say that the algorithm is *overfitting* the dataset, that is, it is giving too much importance to specific information found within the training dataset, which however does not generalise the problem well enough [62].

While machine learning is a complex field with much more theory behind, we believe the information discussed until now is already enough to understand key concepts of the next section, where we will start taking a closer look to one of the core topics of this thesis, that is, deep learning.

2.3 Deep Neural Networks

Deep Neural Networks (DNNs) form the basic building blocks of Deep Learning, which is a class of techniques and methods employed in machine learning. A DNN is an artificial neural network that consists of a minimum of four *layers* of units, also referred to as *neurons*. The first layer is known as the *input* layer, while the subsequent ones are referred to as *hidden* layers. The final layer is the *output* layer. In a typical feed-forward network, every hidden unit obtains one or more elements from the previous layer, combines them, applies a function, and transmits the data to the next layer [73].

Figure 2.1 displays a simple deep feed-forward network. Nodes labeled with x belong to the input layer, those labeled with h are part of the hidden layers, while y represents the only node in the output layer. Because every node of each layer connects to all nodes in the subsequent layer, this network is also known as a fully-connected network.

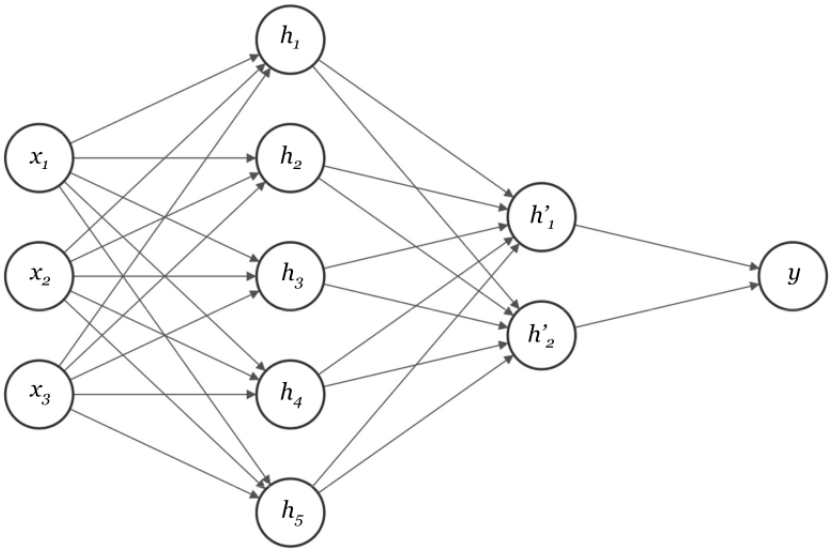

FIGURE 2.1: *Simple Deep Feed-Forward Network*

Deep Learning allows for the resolution of complex problems that are difficult to tackle using traditional programming methods or even classic ma-

chine learning. This is due to either the problem's complexity or the difficulty in determining the features required to build a solution capable of solving it. Some typical examples of Deep Learning applications include [102]:

- **Object identification**: recognising whether an image contains a particular object or identifying and enumerating the objects represented within it. This is a challenging problem because digital images are arrays of pixel values, and numerous variables can influence the object's representation (perspective, lighting, saturation, shape, details, etc.). As such, identifying objects using traditional programming would be a daunting task.

- **Speech-to-text transcription**: generating written text that corresponds to the transcription of an audio file containing a specific utterance. Even in this case, traditional programming faces significant difficulties, as human voices differ significantly from each other and have many characteristics that can influence the final waveform (timbre, pitch, speed, etc.), without taking into account background noise, sound reflections, and other factors.

- **Topic classification**: selecting the correct category amongst a given set, for the paragraph or article under consideration. For humans, this task necessitates an understanding of the text, whereas for a computer program, it requires identifying the correct keywords, determining how sentences are related, and so on.

Deep Neural Networks are usually employed to accomplish the above tasks, with remarkable results. Researchers in Vision Science, the study of visual perception, train Deep Neural Networks to recognise objects by teaching them internal representations of such objects. Neural networks can use pixel values to identify and assemble edges, which are then used to identify motifs, parts, and objects until the whole picture is constructed [101]. This hierarchi-

cal process enables the neural network to learn how to distinguish between similar items' properties and correctly classify new, unseen samples.

The learning process usually occurs using the previously discussed *supervised learning*:[3] human data analysts prepare a dataset that contains many samples that can be used to train the network. For instance, if we are creating a network that aims to recognise whether an input picture depicts a cat or a dog, the training dataset might include 1,000 photos of cats and 1,000 photos of dogs, for a total of 2,000 samples. A separate CSV file generally contains the labels for each photo, indicating the solution to the problem. For example, the label *0* would denote that the picture shows a cat, while *1* would indicate the presence of a dog. The dataset is then partitioned into training and testing parts as previously discussed, usually in an 80:20 ratio.

Intuitively, the training dataset is then used to train the network. The DNN attempts to classify each sample in the dataset, and after each attempt, it verifies whether its prediction was correct. If not, a function called *backpropagation* allows the error to propagate backward within the network (from the output layer to the first hidden layer) to modify biases and weights of the entire model, making it less likely for the error to occur again. This process is repeated for all samples in the training dataset, completing a *training epoch*. However, the network can be trained for multiple epochs for better accuracy. After the training (or after each training epoch), the network can be validated against the testing dataset. Because the DNN has never seen these samples, it allows us to assess the network's performance on new, unseen, samples. This implies that the network must not have access to the correct labels of the testing dataset, and the backpropagation algorithm must not be triggered after each testing sample. Otherwise, the verification could yield misleading values due to the details of the testing dataset leaking into the network [62].

Among the most common types of neural networks, we find Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

[3] Not all DNNs are trained with *supervised learning*; for example, AutoEncoders are typically trained in an *unsupervised* manner.

2.3.1 Convolutional Neural Networks

Convolutional Networks, also known as Convolutional Neural Networks, are deep feed-forward networks that take an array (or a set of arrays) as input, called a feature map, and output another array representing the position of a certain feature [101]. For example, if we feed the network an image (i.e., a 2D array), the first layer of a working network should output a feature map that highlights the position of the edges within the image. Subsequent layers should highlight motifs, and so on until the last layer, which should have identified all the important features of the represented object. In CNNs, the aforementioned steps are usually achieved through convolution and pooling operations within the network, which are connected by a set of weights [102]. Therefore, it takes more than one layer to identify edges from a color image provided as input.

Figure 2.2 illustrates a simple example of a CNN, in which each convolutional layer comprises of the convolution and pooling operations mentioned earlier, along with an additional step called the detector [62]. The convolution operation is typically represented by the following formula:

$$s(t) = \int x(a)w(t-a)\,da$$

Here, x is the input matrix and w is a filter matrix referred to as a kernel [62]. The convolution calculates the integral of the product between the input and the flipped kernel. The output of the convolution is then fed to an activation function, such as the Rectified Linear Unit (ReLU) or Softmax, which forms the detector step. Finally, the detector's output undergoes pooling, which partitions the input into rectangular regions and generates a new $n * m$ feature map. The values in the new feature map depend on the pooling function employed. For instance, the Max Pool function selects the highest value in each rectangular region to populate the corresponding value in the new feature map, whereas other pooling functions utilise the average of the

values or their norm. This process aids the network in retaining consistency despite small input value translations [62].

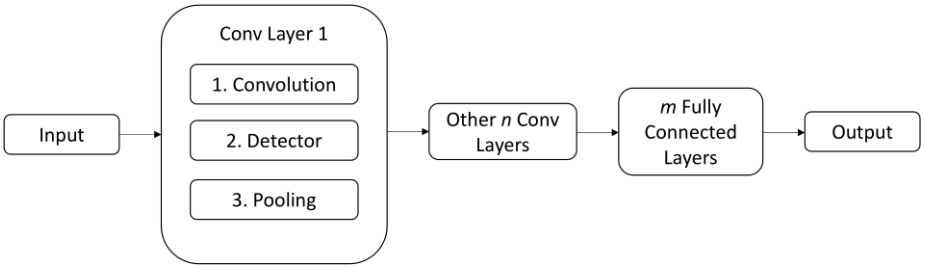

FIGURE 2.2: *Simple Convolutional Network*

We can observe another example in Figure 2.3, where (a) represents a potential input that is divided into 2x2 areas. From such input, the output of the max pool function is represented in (b), while the output of the average pool function is depicted in (c). After all the convolutional layers, the output is usually passed through two fully connected layers, and the final layer is the output layer, which is used for classification purposes.

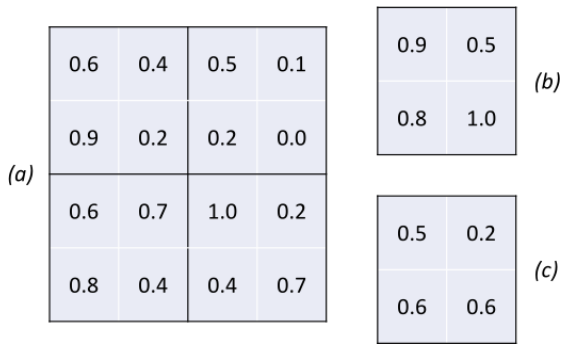

FIGURE 2.3: *Examples of Pooling Operations: (a) Input Matrix After the Application of a 2x2 Filter, (b) Max Pooling Output, (c) Average Pooling Output*

2.3.2 Recurrent Neural Networks

While CNNs excel at working with data that has a clear grid-like structure, such as arrays and matrices, they are not the best choice for dealing with one-dimensional sequential data. In this case, Recurrent Neural Networks (RNNs) are usually preferred [62]. The general idea behind RNNs is to have shared

parameters across different parts of the model, rather than separate layers connected with certain weights as in CNNs. Instead of a set of layers, each specialised in extracting a certain feature (edge, motif, etc.), RNNs have a set of nodes that share information and can retain knowledge of different parts of the analysed input. This is particularly useful when working with language, as there can be multiple formulations of the same sentence. For example, to humans, "Yesterday, I played a tabletop game" and "I played a tabletop game yesterday" have the same meaning, whereas it may not be as obvious for a neural network. The discussed information sharing among the whole model helps RNNs to comprehend this [62]. An example of an RNN that takes input and produces an output at each time step is shown in Figure 2.4.

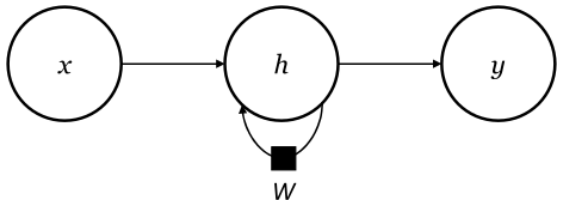

FIGURE 2.4: *A Simple Recurrent Neural Network*

In the picture, we can see the input x, the hidden unit h, the weight matrix W, and the output y. It is worth noting that this is not a *deep* RNN because it only has one hidden layer. This network takes an x_t input and produces an output y_t at any given time t. The hidden unit h at any given time t also receives its previous output as input, which is parametrised by W. Because this network can give as many outputs as the inputs it receives, it is also called a *many-to-many* network, in which the input layer has the same dimension as the output layer [38]. Other types of RNNs include *many-to-many* networks with an input layer that has a different size than the output layer, *many-to-one* networks with many inputs but only one output, *one-to-many* networks consisting of one input but many outputs, and *one-to-one* networks, which have a single input and a single output [38].

2.4 Natural Language Processing

Natural Language Processing (NLP) is a field that enables computers to understand and utilise human languages, such as English, through writing or speaking [62]. It merges various fields such as computer science, artificial intelligence, and linguistics, requiring interested scientists to be proficient or skilled in these related areas [124].

NLP is closely related to the Voice Personal Assistants mentioned in Chapter 1. To interact with the user via the voice channel, the device must (i) capture the user's voice command, (ii) convert it into text, (iii) comprehend the user's request, (iv) execute the command, (v) generate a text response to the command, (vi) convert that response into artificial speech, and finally (vii) output it to the user via the device's speakers. Although these steps are natural for humans when conversing with each other, they involve numerous complex technologies and algorithms when performed by computers. While NLP has several other applications, the following tasks are the most relevant for this thesis and can be grouped into four categories: Speech Recognition, Language Understanding, Language Generation, and Speech Synthesis [43]. In Section 2.5.3, we will discuss Speech Synthesis, while we will provide information on the other three tasks below.

2.4.1 Speech Recognition

In the previous sections, we mentioned Speech-To-Text (STT) transcription as one of the most common applications of Deep Learning. Also known as Automatic Speech Recognition (ASR), the formal goal of this task is to create "*a function f^*_{ASR} that computes the most probable linguistic sequence* ***y*** *given the acoustic sequence* χ:" [62]

$$f^*_{ASR}(\chi) = \arg\max_y P^*(y \mid X = \chi)$$

In this function, $\chi = x^{(1)}, x^{(2)}, ..., x^{(T)}$ is an acoustic input sequence, each audio frame $x^{(t)}$ typically being 20ms long, and $y = y_1, y_2, ..., y_N$ is the set of output words, i.e., the STT transcription, and P^* is the true conditional distribution from χ to y [62].

Although researchers started approaching ASR with technologies like Gaussian Mixture Models (GMMs) and Hidden Markov Models (HMMs) towards the end of the last millennium, it was verified in 2012 that, given the advances in both hardware components and machine learning algorithms, DNNs were more suitable than GMMs to perform this kind of task, and could be used in conjunction with HMMs or other technologies to outperform existing solutions at that time [73]. Recently, the End-to-End (E2E) modeling has revolutionised the ASR field once again, outperforming the current state-of-the-art. There are three main advantages of using E2E approaches over hybrid DNN ones [105]:

- They make use of a single objective function instead of multiple ones, which has the advantage of being able to find a global optimum.
- Their output is a direct list of characters or words, while other approaches require further steps before being able to read a valid ASR transcription.
- Because only one network is used for the task, the solution is overall more compact, enhancing its deployability.

There are several E2E techniques that are currently being used for ASR, but the Recurrent Neural Network Transducer (RNN-T) [66] is the most popular in the industry due to its natural streaming capability [105].

Figure 2.5 shows the design of RNN-Ts, illustrating their three main components for ASR, which are the encoder, predictor, and joiner [109, 81]:

- **Predictor**: takes the previous output y_{u-1} as input (which is `null` for the first iteration) and generates a feature g_u.

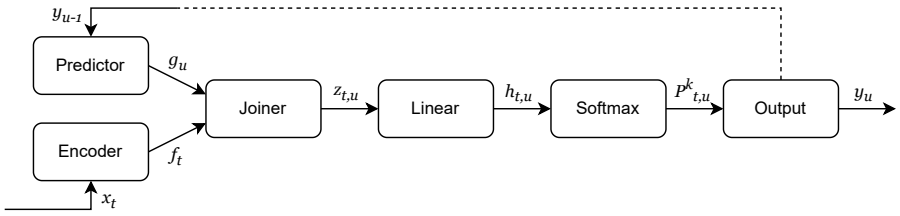

FIGURE 2.5: *A Typical Recurrent Neural Network Transducer (Adapted from [81])*

- **Encoder**: the most critical component in all E2E ASR models. It converts the audio input x_t into a feature representation f_t [105]. While Long Short-Term Memories (LSTMs) were initially used as encoders in RNN-Ts for ASR, Transformers are now preferred due to their attention mechanism [105], which enables them to outperform LSTMs [174].

- **Joiner**: a feed-forward network that combines the encoder's and predictor's outputs. Its output $z_{t,u}$ typically undergoes linear and Softmax functions to generate a probability distribution $P^k_{t,u}$ over all possible labels k (e.g., letters, phonemes, words, or the `null` value).

Once a label is selected, the predictor will use it as input for the next cycle, where the next audio time frame is fed to the encoder. This process repeats until the entire audio is analysed, i.e., no more time frames x_t need to be fed to the encoder.

2.4.2 Language Understanding

After receiving a text utterance, a personal assistant still needs to understand the user's intended action or request. While old-style text-based computer games like Zork[4] included the possibility of giving text commands to a program since the 1970s, these programs lacked the ability to understand commands that were not directly hard-coded.

[4] `https://store.steampowered.com/app/570580/Zork_Anthology/`

Modern Natural Language Understanding (NLU) technologies, sometimes called Spoken Language Understanding (SLU), attempt to semantically interpret the given text input, usually by executing three steps [173]:

- **Dialogue Domain Identification**: understanding the general topic of the user's question.
- **User Intent Extraction**: determining the specific action or information the user wants to obtain.
- **Slot Tagging**: matching keywords in the user utterance to specific slots within the selected intent.

This system is called Domain-Intent-Slot and is usually trained with a large annotated dataset. For skills and actions, whose set of actions is limited compared to all possible utterances a Virtual Personal Assistant (VPA) can receive as input, the application designer trains a smaller model with all intents related to their application. In this case, the domain is the skill itself, while slots can be seen as parameters of a specific intent [40]. For example, in *"Alexa, tell me how to complete the computer game Zork"*, the domain would be *computer games* or *video games*, the intent would be *how to complete* or *completion*, and there would be a single slot with the value *Zork*.

Alternatives to the manual annotation of training datasets for Domain-Intent-Slot solutions are still being developed. For example, Zeng et al. [173] propose a solution to automatically detect intents and slots in a domain-independent environment. They perform three steps:

- **Intent-Role Labeling**: classifying each sub-word of an input utterance into an approximate label, using BERT [46].
- **Concept Mining**: refining each label to make it more precise, by means of mention embeddings (e.g., Word2vec [117]) and clustering.

- **Intent-Role Pattern Mining**: applying the Apriori algorithm [164] to obtain the most frequent patterns for the extracted intent-role, and determining which intent has the highest probability of being correct.

Another work by Kim et al. [90] leverages enriched word embeddings and a bidirectional LSTM to improve state-of-the-art intent detection.

2.4.3 Language Generation

Language generation can be seen as the inverse of language understanding, where the former maps abstract concepts or data to written text, while the latter maps text to its meaning [142]. Before the advent of the Neural Network revolution in the 21st century, Natural Language Generation techniques relied on canned text, template filling, or more complex language generators consisting of a discourse planner and surface realizer [142]. Today's natural language generation techniques have evolved and can be classified into various categories, each with its own sub-problems and state-of-the-art solutions, often revolving around deep neural networks [50]:

- **Text Abbreviation**: this category involves summarising a long text into a shorter one by extracting its key concepts. The three sub-problems within this field include text summarisation, question generation, and distractor generation.

- **Text Expansion**: this category involves expanding a short text to make it longer. The two sub-problems within this field include short text expansion and topic-to-essay generation.

- **Image-to-Text Generation**: this category involves extracting information from a video or image in the form of text. The three sub-problems within this field include image captioning, video captioning, and visual storytelling.

- **Text Rewriting and Reasoning**: this category involves extracting the main concepts from a sentence and applying them to generate new short sentences or questions. The two sub-problems within this field include text style transfer and dialogue generation.

Dialogue generation is a crucial sub-problem that concerns Voice Personal Assistants (VPAs) as they need to constantly answer user queries based on data found online. While skills and actions still rely on template filling, VPAs can leverage state-of-the-art dialogue generation techniques when no skill is involved in the conversation. For example, the ChatGPT Virtual Assistant, built around Generative Pre-Trained Transformers (GPTs), is one of the current state-of-the-art solutions for language generation and text classification [136]. Other state-of-the-art solutions include Recurrent Neural Networks, Knowledge Graphs, Generative Adversarial Networks, and more [50].

2.5 Voice-Controllable Devices

In Chapter 1, we already discussed how the Internet of Things has drastically changed our lives over the last decade. Our homes are now filled with interconnected devices, some of which we can simply control with our voice. We start this section by illustrating how many kinds of VCDs are there. After, we discuss that, while these devices may improve quality of life for some users, there are also concerning issues related to their security and to their user's privacy. Hence, we provide a brief summary of such problems.

2.5.1 Diversity of Personal Assistants and VCDs

VPAs are not the only existing assistants, as mentioned earlier. Past research classifies personal assistants in five clusters, or *archetypes* [92]:

- **Adaptive Voice (Vision) Assistants**: they assist the user through the voice channel and, if possible, through visual cues or screens.

- **Chatbot Assistants**: they assist the user through a text channel.
- **Embodied Virtual Assistants**: they display a virtual character on a screen, and provide a human-like interaction to their user, mostly for educational purposes such as e-learning.
- **Passive Pervasive Assistants**: they do not interact with the user, but proactively perform operations depending on information they gather from the environment.
- **Natural Conversation Assistants**: they resemble human-to-human interactions through speech. They can have different goals, such as acting as a call center agent, or just pure conversation.

We can easily map the Voice Personal Assistants we talked about until now to the first category of this taxonomy. However, we still need to identify which kind of devices can be controlled through these VPAs, that is, which kinds of VCDs are there. Kumar et al. [96] define a list of fourteen smart home devices that can be found within a user's network: computers, network nodes, mobile devices, wearables, game consoles, home automation, storages, surveillance systems, work appliances, home voice assistants (which we usually refer to as *smart speakers* in this thesis), vehicles, TVs or media devices, home appliances and other generic IoT devices.

While reading through this list, the reader will have probably noted that we already introduced many of these devices. All of these devices can be voice-controllable, although some devices will need to be paired to a VCD running a VPA to work, as they do not directly embed the VPA. For example, while a computer running Windows will have direct access to Cortana and is therefore directly voice-controllable, a smart vacuum cleaner will have to be paired to a smart speaker to be voice-controllable. In other words, the user will need to issue the command to the smart speaker to turn on the vacuum cleaner, and will not be able to talk directly to the latter. It follows that all

these devices have different capabilities, and the user will be in charge of giving the right command to the right device (e.g., "Hey Google, tell the Roomba vacuum cleaner to make me a coffee" will not be a valid command).

Similarly, given the vast amount of kinds of voice-controllable devices, analysing the whole attack surface for all said categories is outside the purpose of this thesis, as we want to investigate their vulnerability to self-issued commands specifically, and we will focus on the security assessment of their voice channel in particular. For this reason, we will only analyse the main security features offered by the majority of VCDs in Section 6.2 and, apart from that, we will focus on the security of the voice channel only.

2.5.2 Privacy Concerns

Privacy has been one of the primary concerns for users of smart speakers ever since their release, primarily due to the fact that their microphones are always active. In addition, sometimes VPAs misclassify noise or certain words as wake-words, leading to the recording of potentially private conversations that the user did not intend to disclose [15]. According to a study by Abdi et al. [1], 17 volunteers who owned a smart speaker had different perceptions about the data processing, storage, sharing, and learning performed by voice-controllable devices. They were also unaware of how to protect themselves from existing attacks and were not aware of the existence of malicious skills. It is important to note that VPA applications do not need to be malicious to cause privacy issues. Alhadlaq et al. [7] analysed over 11,000 skills and found that 75% of them did not have a privacy policy, so their users did not know what kind of data was being processed or how it was used. Furthermore, among the skills with a privacy policy, 11% had an invalid policy that redirected users to non-relevant pages or returned an error page.

Other researchers were able to perform a voice fingerprinting attack, which is the inference of a voice command issued by a user by solely eavesdropping on the encrypted traffic sent and received by the VPA device [87].

To do this, they used Wireshark to capture network traffic and then converted PCAP network capture files into CSV network trace files, which were used to train their model. The researchers argue that this attack can successfully infer 33.8% of the captured voice commands and propose a countermeasure using a padding method called BuFLO. However, this mitigation would introduce a high communication overhead.

The work by Mhaidli et al. [114] suggests that users view microphones as one of the most intrusive and privacy-violating sensors. To address this issue, they propose a way to dynamically turn on and off the voice-controllable device's microphone by using interpersonal communication cues, such as the user's gaze: the device's microphone turns on only if the user is directly looking at the device. This method received positive feedback from users who tested it, although there were some usability issues due to the fact that it was hard for users to determine whether the microphone had turned on or not.

2.5.3 Security Issues

As consumer-grade VPAs are relatively new, research has begun to explore new angles for exploiting such devices. For example, adversarial noise commands are voice commands generated through Adversarial Machine Learning, in such a way that an ASR system misclassifies their content. Although the human ear can hear a sentence *s* (or even no sentence at all [175]), the ASR system classifies it as another sentence *s'*, chosen by the attacker. Such adversarial attacks can have different goals, such as hindering command recognition [106] or bypassing security measures [33, 94]. Other attacks against smart speakers deceive the user into thinking they are talking with the VPA or a legitimate application when they are actually talking with an attacker-controlled application or device [95, 120, 177]. However, most of these works [120, 35, 106] use external speakers to issue commands to a VPA, reducing the overall likelihood of the attack. Another category of attacks against voice-controllable devices leverages misclassification, where the VPA wrongly tran-

scribes the user command, leading the user to perform actions they did not intend to do [23, 95]. In this section, we discuss the most relevant classes of attacks found in the literature within the last decade.

Skill Squatting and Misclassification

In their research, Kumar et al. [95] identify three categories of errors made by VPAs when interpreting voice commands: (i) *homophones*, which are two words pronounced the same way but with different spellings; (ii) *compound words*, which can be split into their components (e.g., "outdoors" and "out doors"); and (iii) *phonetic confusion*, which is the misclassification of one phoneme with a similar one, resulting in the transcription of a different word. When this misclassification occurs during the transcription of a skill name, the VPA may open a skill that the user did not intend to activate. Adversaries can exploit this vulnerability by publishing malicious skills with names that are similar to popular skills. This attack is known as Skill Squatting. While this attack is often associated with Alexa's context (i.e., skills), it is also valid on other VPAs, such as Google's *actions*. However, Lentzsch et al. [104] found that Skill Squatting attacks are not being used systematically in the wild and noted that multiple skills can share the same invocation name, which can cause the VPA to activate the wrong skill even if the name is correctly transcribed. In another study, Bispham et al. [23] explore nonsense attacks on Google Assistant, demonstrating that it is possible to trigger actions using words that are nonsensical to humans but are interpreted as valid commands by the VPA. They also provide a proof-of-concept for the so-called missense attacks on Amazon Alexa, which consist in triggering skill intents with voice commands unrelated to the intents themselves.

Voice Masquerading Attacks

Zhang et al. [177] propose the concept of a Voice Masquerading Attack (VMA), in which a malicious application impersonates the VPA to deceive the user

and exfiltrate personal data. Once the user runs the malicious application (or the adversary finds a way to run it on the target device, for example, if they are in physical proximity of an unattended device), the adversary can capture all commands given to the application and give arbitrary replies while the user believes they are interacting with the VPA. This can lead to both safety and privacy violations, as the adversary can tamper with replies to actions like turning on the heating or locking the front door, making it seem as if the actions have been taken even if they have not. Additionally, the adversary can ask for more details and trick the user into disclosing personally identifiable information. During the VMA, the adversary may also pretend to invoke a skill, impersonating it and faking its termination when appropriate.

The Lyexa attack [120] is the first example of a successful implementation of a VMA. The attack leverages a rogue device featuring a microphone and an ultrasonic speaker that the attacker places near the target device. The rogue device jams legitimate voice commands and records them, modifies them to be performed with an evil skill, and sends them to the target device via ultrasound. For example, an *"Alexa, what time is it?"* command can be modified to *"Alexa, what time is it? Use Evil skill"*, and the command will be redirected to the malicious application controlled by the attacker. As ultrasounds are unhearable for humans, this allows the attacker to intercept, edit, and send to the target VPA all legitimate commands while also tampering with the reply.

Voice Spoofing Attacks

Voice spoofing attacks can be categorised into four different types: impersonation, replay, voice conversion, and speech synthesis [162, 32, 167]. Although the term *voice spoofing attack* usually refers to attacks that target Automatic Speaker Verification solutions in literature, some of these attacks also target the underlying Automatic Speech Recognition system. Therefore, in this section, we discuss some examples of each of these types of attacks. We also show that adversarial noise attacks can be included in the list of voice spoof-

ing attacks as they can target both Automatic Speaker Verification (ASV) and ASR systems: this was also done by Khan et al. [88], who list adversarial noise attacks under the *"indirect"* category in their attack taxonomy.

Impersonation Attacks. Impersonation attacks feature an adversary who attempts to mimic the legitimate user's voice to bypass an Automatic Speaker Verification system without any technological aid. It is also known as *human mimicking*. It is still unclear how effective impersonation attacks are against ASV systems, as studies have only been performed with a handful of speakers and results appear inconsistent across various ASV systems and feature representations [162]. Intuitively, voice impersonation attacks are more successful when the impersonator's voice is similar to the target user's voice. However, even professional impersonators are not consistently able to deceive current ASV systems [71, 112].

Replay Attacks. In replay attacks, attackers record the legitimate user while they issue a voice command and then use it to bypass the Automatic Speaker Verification system. Attackers can also use a pre-recorded speech sample, featuring the legitimate user uttering a voice command, collected by someone else [162]. Hence, replay attacks are relatively easy to perform as the adversary does not need any technical expertise to carry out the required steps. Recently, a replay attack involved the adversary surreptitiously recording the legitimate user's voice by using a recording device that is identical to the one embedded in the target device. They then use a mouth simulator[5] to replay the voice command and perform the attack, bypassing state-of-the-art ASV solutions [169].

Voice Conversion Attacks. Voice conversion systems manipulate an audio file containing an utterance spoken by a certain user *u'*, with the intention of tricking an Automatic Speaker Verification system to classify the utterance as

[5] Mouth simulators are pieces of hardware that attempt to closely replicate human voice and must meet certain standard requirements to be classified as such.

coming from the legitimate user u [162]. The so-called Parallel Voice Conversion requires the adversary to have voice samples of both u and u' to use as a baseline (e.g., for training purposes). Additionally, these voice samples must have the same content — in the case of spoofed voice commands, they would have to feature the same utterances. In the case of Non-Parallel Voice Conversion, the adversary still needs to have samples from both users, but utterances do not need to be aligned [84, 85]. Cross-language voice conversion is also possible: Zhou et al. [179] leverage Phonetic PosteriorGrams and average modelling to perform voice conversion between English and Mandarin speakers. Recent work in the field is also exploring *many-to-many* and even *any-to-many* voice conversion, as demonstrated by Guo et al. [68].

Speech Synthesis Attacks. Speech synthesis attacks use Text-To-Speech (TTS) systems to convert written sentences into natural-sounding artificial speech [162]. While generic TTS solutions cannot bypass ASV systems, they can still issue commands to most commercial voice-controllable devices. Consequently, the term *speech synthesis* sometimes refers to the synthesis of speech with the goal of impersonating a particular user, whereas *TTS synthesis* refers to speech created for non-adversarial purposes.

Advanced methods for speech synthesis include Generative Adversarial Networks, like SEGAN [129], and probabilistic models like WaveNet, which is a deep neural network designed to generate raw audio waveforms [154]. WaveNet has the unique ability to generate natural-sounding speech as well as other types of waveforms, such as music. Recent research in speech synthesis has produced VALL-E, a language model that uses conditional language modeling to generate customised speech with just three seconds of training on the target voice [157]. This technique, known as "zero-shot" learning, is achieved by training VALL-E on over 60,000 hours of English speech.

Adversarial Noise Attacks. Adversarial noise attacks often rely on machine learning to develop customised solutions against Automatic Speaker Verifi-

cation, Automatic Speech Recognition, Liveness Detection, or other countermeasures. Abdullah et al. [2] conducted an analysis of attacks and defenses on automatic speech and speaker recognition and provided a taxonomy of threat models on voice processing systems. The authors also tested the transferability property for adversarial noise commands, demonstrating that it is still challenging to craft adversarial noise samples that are transferable.

In a white-box attack against Mozilla's DeepSpeech ASR implementation, Carlini and Wagner [29] developed targeted audio adversarial noise examples that can alter any waveform into another that is almost identical to the original and gets classified by DeepSpeech as any other text chosen by the adversary. However, this attack is not feasible over-the-air.

Recently, Yu et al. [170] developed a solution that applies semantic perturbations to any audio file containing a spoken sentence. This technique ensures that the sentence appears unaltered to the human ear, but the VPA converts it to a completely different utterance. The attack is undetectable to the human ear, works over-the-air, and was tested against Amazon Echo, confirming its dangerousness.

Devil's Whisper [35] embeds hidden commands into songs, similarly to CommanderSong [171] from which it stems. This approach targets black-box systems using a transferability-based approach and two models to generate adversarial noise samples for most of the commercial ASR systems.

DolphinAttack [175] modulates voice commands on ultrasonic carriers with a frequency greater than 20 kHz, achieving complete inaudibility of the malicious voice command. However, this technique requires an ultrasonic speaker to be performed. The already discussed Lyexa [120] also leverages voice commands modulated on ultrasonic carriers.

An adversarial attack that does not leverage audio files is discussed in the work by Sugawara et al. [147]. This technique aims amplitude-modulated light towards an exposed microphone aperture in a VPA device and is able to inject arbitrary commands even when more than 100 meters away from

the target device. Similarly, Yan et al. [168] make use of a piezoelectric transducer to guide ultrasonic waves through a solid transmission media to inject inaudible commands.

2.5.4 Discussion on Security Issues

Although not all the attacks discussed in Section 2.5.3 are strictly related with the rest of the thesis, and comparison of our work with these will be done in the next chapters when appropriate, with this section we want to show that some form of initial access is always required to perform attacks on the voice channel. This is not to be considered a problem, as in fact, even in the "classical" domain of system security exploitation, obtaining exploits that do not require any precondition is quite uncommon. Table 2.1 presents the initial foothold required by all the attacks discussed in Section 2.5.3. For simplicity, they are categorised as done in Section 2.5.3 within the table as well. As can be seen, most attacks necessitate a speaker, whether it is an ultrasound one or not, near the target device. This is because the malicious audio must be reproduced in proximity to the target device to ensure that it hears and executes the command. Social engineering is required for some attacks, primarily because they necessitate user interaction, such as unwittingly opening a malicious skill or interacting with a malicious skill instead of the VPA. A few attacks require direct input over-the-line, which may be impractical in some circumstances. Some attacks require setting up specific equipment near the device or having a line of sight. Finally, for some attacks, particularly those in the Voice Spoofing category, the required initial foothold is not explicitly stated in the papers because they target human listeners or Speaker Verification systems over-the-line rather than VCDs. However, it can be reasonably assumed that the malicious audio files would need to be played in the vicinity of the victim device for these attacks to work, necessitating the use of a rogue speaker near the target, or a way to feed the malicious audio file over-the-line needs to be found.

While this shows that all mentioned attacks on Voice-Controllable devices require an initial foothold to work, this does not undermine their overall impact, which remains critical to the user's safety, security, and privacy.

TABLE 2.1: *Discussed Attacks to VCDs and Required Initial Footholds*

Paper	Category	Required Foothold
Bispham et al. [23]	Squatting/VMA	Speaker near victim device
Kumar et al. [95]	Squatting/VMA	Social Engineering
Mitev et al. [120]	Squatting/VMA	Ultrasound speaker near victim device
Zhang et al. [177]	Squatting/VMA	Social Engineering
Guo et al. [68]	Spoofing (Conversion)	Not discussed
Hautamäki et al. [71]	Spoofing (Impersonation)	Not discussed
Kaneko and Kameoka [84]	Spoofing (Conversion)	Not discussed
Kaneko et al. [85]	Spoofing (Conversion)	Not discussed
Pascual et al. [129]	Spoofing (TTS)	Not discussed
van den Oord et al. [154]	Spoofing (TTS)	Not discussed
Wang et al. [157]	Spoofing (TTS)	Not discussed
Yoon et al. [169]	Spoofing (Replay)	Mouth simulator and microphone near victim device
Zhou et al. [179]	Spoofing (Conversion)	Not discussed
Carlini and Wagner [29]	Adversarial	Direct input over-the-line to the victim device
Chen et al. [33]	Adversarial	Speaker near victim device
Chen et al. [35]	Adversarial	Speaker near victim device
Kreuk et al. [94]	Adversarial	Not discussed
Li et al. [106]	Adversarial	Speaker near victim device
Sugawara et al. [147]	Adversarial	Equipment, line of sight on the victim device
Yan et al. [168]	Adversarial	PZT Transducer near victim device
Yu et al. [170]	Adversarial	Speaker near victim device
Yuan et al. [171]	Adversarial	Speaker near victim device
Zhang et al. [175]	Adversarial	Ultrasound speaker near victim device

2.6 Amazon Echo and Alexa

As in Chapter 4 we discuss an attack to self-issue commands to Amazon Echo Dot specifically, it seems useful to give some more details regarding said voice-controllable device and its VPA, Alexa. Although it is possible to install Alexa on smartphones, computers and tablets if the Amazon Alexa application is available on the OS that is running on the device, Amazon has released lots of devices running Alexa natively, which come in different designs and can be split in two categories: home devices and portable devices. Among the home devices we find:

- **Amazon Echo**: the main Amazon Alexa voice-controllable device. Until the third generation it was cylinder-shaped, but the fourth one, which is the latest, has the shape of a spherical cap, with 13 cm of height and 14

cm diameter. It has a 76,2 mm woofer and two 20 mm tweeters. The version that comes with an embedded clock is called Amazon Echo Plus.[6]

- **Amazon Echo Dot**: a cheaper VCD with lower overall sound quality. Until the third generation it was cylinder-shaped, but fourth and fifth generation Echo devices are in the shape of a spherical cap. They are all approximately 9 cm tall with a diameter of 10 cm. The fifth generation Echo Dot embeds a 44 mm loudspeaker.[7]

- **Amazon Echo Studio**: a more expensive VCD with very high quality sound, embedding a 25 mm tweeter, a 133 mm woofer and three 51mm midrange loudspeakers.[8]

- **Amazon Echo Show**: a VCD that also embeds a touchscreen. Differently from the others, this Echo device can allow interaction with the user via text or by touching the screen. It has a few more functionalities, as it can be used to perform videocalls, or as a digital photo frame, and there are a few widgets that always keep some information on screen. Versions of Echo Show are named after their screen size: 5", 8", 10" and 15". They all embed different loudspeakers and microphone arrays.[9]

- **Amazon Echo Pop**: a small device designed for small spaces. It has the shape of a semi-sphere with approximately 9 cm diameter. It embeds a 49.5 mm loudspeaker.[10]

Amazon does not always explicitly list how many microphones are there in the microphone array embedded in each home device. However, it is fair to assume that more expensive devices have better or more microphones within the array: for example, it has been reported that Echo embeds a seven-microphone array, while Echo Dot only has a four-microphone array.[11] An-

[6] https://www.amazon.com/Echo-bundle-Made-Amazon-Mount/dp/B08N6D9DYR
[7] https://www.amazon.com/dp/B09B8V1LZ3
[8] https://www.amazon.com/Echo-Studio/dp/B07G9Y3ZMC
[9] https://www.amazon.com/smart-display-Alexa-Remote-included/dp/B0BFZVFG6N/
[10] https://www.amazon.com/dp/B09ZXJDSL5/
[11] https://www.digitaltrends.com/home/amazon-echo-vs-dot/

other example is the Echo Show: the cheaper 5" version only has a four-microphone array, while the more expensive 15" version embeds a eight-microphone array.[12] Among the portable devices, we currently find only two devices, as others are now out of production:

- **Amazon Echo Buds**: voice-controllable earbuds that feature the Alexa VPA and embed two microphones per bud.[13]

- **Amazon Echo Auto**: a device that allows hand-free interaction with the Alexa VPA while driving. The second generation of these devices is the latest and embeds a five-microphone array.[14]

As we can see, there are many hardware differences between all the available Echo devices. Instead, regarding software implementations, unfortunately Amazon did not release any detail about the stack underlying the Alexa Voice Service, hence we do not have any detail regarding the Alexa ASR either. This is because Alexa Voice Service runs on the cloud, hence, we have no means to analyse it.

On the usability side, however, by simply using the device we can tell that Amazon has implemented several software features: for example, all Amazon Echo devices implement some sort of *sonification*, which is a non-verbal signal that is able to convey certain information to a human listener, who can learn what it means [44, 160]. Therefore, when a timer goes off, the device emits the sound *a*; when it is time to remind the user a previously mentioned item, the device emits the sound *b*; when a new notification is ready to be read, the device emits the sound *c*; when the Echo device is receiving a call, it emits the sound *d*. This allows the user to avoid confusion if they receive all these sound notifications one after another, because they are all designed to convey different messages.

[12] See Footnote 9.

[13] `https://www.amazon.com/All-new-Echo-Buds-2023-Release-True-Wireless-Earbuds/dp/B09JVG3TWX`

[14] `https://www.amazon.com/Echo-Auto-Adjustable-Vent-Mount/dp/B0BFCJF66B`

2.6.1 Considerations for Self-Activation Attacks

As we observed, Echo devices come in a multitude of shapes, with varying speaker and microphone architectures. Even without considering tablets, laptops and smartphones, each with their own custom speakers and microphones, we can already witness several combinations of hardware and device shapes that will affect how the sound is emitted and received from the device itself. This hinders self-activation attacks from an adversary's perspective, as there is no fixed and general condition in which the attack can take place. All home Echo devices are designed to be placed in a fixed spot within a room, without the need of moving the device anymore. However, if the user has a battery to make their Echo device portable,[15] this introduces another obstacle for the self-activation attack.

In Chapter 4 we will perform a self-issue attack on a 3rd Generation Echo Dot device, but for the above reasons, payloads created to work against that device will not necessarily work against another device of the Echo family, as the sound will be emitted and captured in a different way.

2.7 Further Key Concepts

There are other key concepts that the reader should know to fully understand this thesis. However, we will explain such key concepts, when needed, within the "Related Work" sections that each of the next chapters embed. This is because such sections illustrate similar works and past literature that is strongly related to what is presented within the chapter, so that the reader can instantly compare our contributions with the existing literature, without having to go back to this chapter to check relevant information. Key concepts that will be explained in the following "Related Work" sections are:

- Attack Models, such as Kill chains and OODA loops (Chapter 3);

[15] https://www.amazon.com/All-New-Made-Amazon-Battery-generation/dp/B09RV39989

- Threat Models and relevant frameworks (Chapter 3);
- Countermeasures to Voice Spoofing Attacks, such as Liveness Detection and Automatic Speaker Verification (Chapter 7).

2.8 Summary

In this chapter, we explored the fundamentals of information security. After, we investigated the basics of Machine Learning, and then delved into Deep Learning and its applications. We also provided an overview of Natural Language Processing and how its various sub-topics are interwoven to create voice-controllable devices, which have become ubiquitous in modern smart homes. Given their widespread adoption, it is crucial to protect these devices from security threats and privacy breaches. As we have seen, numerous attacks and privacy concerns have surfaced in recent years, revealing a growing number of attack vectors and techniques. These challenges underscore the need for robust security measures and ongoing research to ensure the safe and secure use of voice-controllable devices in our daily lives.

Chapter 3

The VOCODES Framework for Attacking VCDs

In this chapter, we introduce a formalisation of attacks on voice-controllable devices, focusing specifically on attacks leveraging the voice command self-issue. To this end, we propose a tailored kill chain that lists the necessary steps to perform self-activations, and we define a threat model to illustrate the typical conditions under which attackers will operate when engaging in these types of offensive activities. Together, these two tools make the VOCODES (VOice COntrollable DEvice Self-issue) Framework.

3.1 Introduction

Defining and assessing environmental conditions before performing experiments is vital to allow their reproducibility, and attacks on the cyber-physical domain make no exception. Hence, the VOCODES Framework that we introduce in this chapter and that was followed throughout the whole research will allow future researchers and security analysts working on voice command self-issue to perform other experiments (or reproduce ours) within the same environment and conditions. This also allows researchers to compare results and draw conclusions based on the performed comparison.

The VOCODES Framework enables all of this by showing (i) *which* steps an attacker has to perform to successfully execute a self-activation attack, by means of our tailored kill chain, and (ii) *how* they should do so, by means of the threat model defined in the following pages. We show that the VOCODES Framework builds upon existing literature in a tailored way, by excluding concepts that are not relevant in the self-activation context, modifying those elements that still make sense in such a context but that may need further customisation, and including new ideas that are applicable exclusively in this context. For example, in the creation of the VOCODES Kill Chain, we did not include any step that requires the *execution* of source code, which is a common step in other kill chains, then we revisited the meaning of other basic steps, such as the classical *reconnaissance*, and finally, we introduced new techniques such as the *audio weaponization*.

Going forward, in Section 3.2 we illustrate other attack models, with particular focus on kill chains, and we discuss threat models; in Section 3.3 we explain our tailored kill chain: finally, in Section 3.4 we formalise our threat model for self-activation attacks.

3.2 Related Work

In this section, we illustrate different attack models, starting from kill chains, useful to quickly pinpoint the current stage of an ongoing attack, and then delving into other models. We then discuss threat models, which allow us to formalise the conditions in which an attack takes place.

3.2.1 Attack Kill Chains

"Kill Chains" refer to the steps an attacker needs to take to carry out a successful attack. This concept is essential for security researchers and analysts to predict and react to potential attacks. Kill chains allow researchers to visualise an attacker's possible actions, enabling them to understand the tech-

niques used during an attack. Offensive security researchers can also use kill chains to pinpoint where they are in the attack process, identify possible next steps, and evaluate previous attempts [77]. In this section, we will examine several essential kill chains discussed in the literature to understand the underlying concepts of these valuable tools.

Lockheed Martin's Cyber Kill Chain®

While the concept of kill chains was not new in the military context [152] and in general in the cyber-physical domain [159], Lockheed Martin's Cyber Kill Chain® (CKC) [77], or Intrusion Kill Chain, is the first work in literature to formalise steps for a successful intrusion within a certain secured and trusted digital boundary. Authors describe kill chains as a "*systematic process to target and engage an adversary to create desired effects*" [77], and they are represented as a "chain" because a single failure in one of the steps will cause a failure in the entire process. The Intrusion Kill Chain has seven steps:

1. **Reconnaissance**: research and selection of targets to attack.
2. **Weaponization**: creation of a deliverable that can be used to perform the intrusion, such as a file containing malware.
3. **Delivery**: actual delivery of the weapon within the security boundary to be breached.
4. **Exploitation:** activation of the weapon and execution of its code, be it a Microsoft Word file with a malicious macro on it, or a SQL Injection within a website.
5. **Installation**: creation of a backdoor that enables the adversary to maintain persistent control of the target.
6. **Command and Control**: connection of the target to an attacker-controlled channel, to enable the adversary to remotely control one or more targets at once.

7. **Action on Objectives**: achievement of general mission goals: collection, exfiltration, and deletion of data, for example.

After detailing the seven steps, the authors explain how this information can aid security analysts in quickly identifying the current stage of a live attack, as well as efficiently categorising the different phases of security incidents or red-teaming activities during post-event analyses.

Derived Kill Chains

While the Intrusion Kill Chain certainly sets a milestone in literature for both offensive and defensive security activities, the research community later identified two subtle issues within its structure [89].

The first problem is that kill chains in the digital domain are *not exactly chains*, for two reasons: (i) the adversary may fail one or more steps listed within the CKC, such as Installation or Command and Control, and be able to complete their mission nonetheless; (ii) during a real attack, the adversary might want to execute steps in a different order, or might need to "go back" and execute a previous step within the chain again — in other words, the CKC and other *linear* models do not allow for cyclicality of actions and do not show that some attacks, such as those performed by Advanced Persistent Threats (APTs), theoretically do not end if the threat (e.g., a malware) is not detected and removed.

The second problem with the Intrusion Kill Chain is that it has no depth, that is, it does not reflect the fact that some systems are not directly exposed to the adversary. This means that, if the attacker wants to target them, they need to breach another security boundary and execute a successful pivoting activity from there, before being able to breach the target.

Modified Kill Chain The Modified Kill Chain (MKC) [89] tries to address both problems at once: authors start with an explanation of the two problems and a selection of *linear* and *circular* models derived from the CKC that fail

to address them. Afterward, they present their solution as a model with two layers: an external one (External Threat Layer) and an internal one (Internal Threat Layer).

- **External Threat Layer**: consists of the seven steps found in the Intrusion Kill Chain, plus an Exit step that however is not explained within the paper.
- **Internal Threat Layer**: very similar to the previous one, consists of five steps: Internal Reconnaissance, Weaponization, Delivery, Exploitation, and Installation — all already known steps in this new internal context.

Note that this model also allows for cyclicality, although a somehow limited one: while Command & Control can lead to Reconnaissance, and (internal) Installation can lead to Internal Reconnaissance, creating some cycles, it is not clear why they could not lead to other steps as well.

Expanded Kill Chain Continuing from the second issue discussed in Section 3.2.1, the Expanded Kill Chain (EKC) [110] proposes a solution by treating the Intrusion Kill Chain as a process for gaining access to a secure perimeter that does not yet contain the target system. The adversary must perform additional actions before interacting with and breaching the system. Therefore, the Expanded Kill Chain divides the attack into three distinct parts: Legacy Kill Chain, Internal Kill Chain, and Target Manipulation Kill Chain

- **Legacy Kill Chain**: consists of the seven steps found within the Intrusion Kill Chain, in the same order. Because the adversary is trying to get access to the external perimeter, the *Exploitation* step is known as *External Exploitation* in the EKC.
- **Internal Kill Chain**: once the adversary gets access to the perimeter, they usually have to look for their target. Hence, the Internal Kill Chain consists of five steps: Internal Reconnaissance, Internal Exploitation,

Enterprise Privilege Escalation, Lateral Movement, and Target Manipulation. While the first two steps are (again) clearly already known steps within a different context, Enterprise Privilege Escalation and Lateral Movement allow the adversary to perform horizontal and vertical movement within the network, to look for their target. Once found, the adversary can proceed with the Target Manipulation, that is, they can proceed with the final kill chain.

- **Target Manipulation Kill Chain**: at this point, the adversary only needs to gain access to the target system and complete the mission. The five steps of this final kill chain are: Target Reconnaissance, Target Exploitation, Weaponization, Installation, and Execution, with this last step being the one in which the adversary activates the malicious payload(s) to thwart system availability, steal or destroy data, etc.

While this work explains very well the differences between the actions that the adversary has to perform within the different environments, it does not address the problem of the cyclicality of actions.

The Unified Kill Chain

The Unified Kill Chain (UKC) [132] tries to combine the points of strength of the most used kill chains while also maintaining a practical and qualitative approach. The author describes the process for its creation, starting from a literature review of the most important kill chains and then going through a refinement process of the identified steps with the aid of different case studies. The output of this activity is a detailed kill chain consisting of 18 steps, divided into three different cycles: In, Through, and Out.

- **The In Cycle**: describes the activities that the adversary has to perform in order to enter a certain security boundary. Steps in this phase include Reconnaissance, Weaponization, Delivery, Social Engineering, Exploitation, Persistence, Defense Evasion, and Command & Control.

While most of these are already known, Social Engineering separates the human contribution to the attack from the actual exploitation, while Defense Evasion gives further granularity on what is needed to gain access to the organisational perimeter.

- **The Through Cycle**: the adversary is now in the targeted network, but they do not have yet all the required privileges to access the target assets and complete the mission. This phase includes Pivoting, Discovery, Privilege Escalation, Execution, Credential Access, and Lateral Movement. While Privilege Escalation, Lateral Movement, and Execution are already known from the EKC, Pivoting and Discovery further detail how the adversary should explore and communicate with the other devices within the internal network.

- **The Out Cycle**: the adversary has now access to the target system and gains the privileges to execute the final tasks and complete the mission. Actions in this phase include Access, Collection, Exfiltration, Impact, and Objectives, giving granular detail over which actions on assets and data the adversary might want to perform.

By describing the different contexts in which the attack takes place as "cycles", this kill chain elegantly solves both problems discussed in Section 3.2.1. Furthermore, the authors acknowledge that the adversary does not necessarily need to perform all the actions to succeed — they do not even need to follow the order in which they are presented, and one or more failures do not necessarily imply a mission failure. With its high level of detail and practical design, the Unified Kill Chain is currently considered the state-of-the-art regarding kill chains.

MITRE ATT&CK®

It is also worth mentioning the MITRE ATT&CK® (ATT) matrix [150], although it is not considered a kill chain, but more of a knowledge base. This matrix

lists all known adversary tactics and techniques observed in real-world attacks, divided into 14 different categories: Reconnaissance, Resource Development, Initial Access, Execution, Persistence, Privilege Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement, Collection, Command and Control, Exfiltration and Impact. While these should all be familiar names for the reader by now, ATT&CK® goes the extra mile by presenting all known techniques to perform each step. For example, *Initial Access* might be performed via *Phishing*, the use of *Valid Accounts*, or because of a *Trusted Relationship*. The added value in this is that, for all of these techniques, ATT&CK® also provides all known sub-techniques and tools that can be used for the purpose, making it a valuable reference for both red and blue teaming activities.

3.2.2 Other Attack Models

In this section, we discuss other attack models and we compare them with kill chains, to highlight why the latter were chosen to illustrate self-activation attacks in this work.

OODA Loops

Similarly to kill chains, OODA Loops were firstly introduced in the military context [27], but nowadays they have multiple fields of application, including information security [26, 172]. OODA Loops help in defining hypotheses and tests to make (or, generally speaking, actions to perform next) by gathering and analysing all information available from multiple contexts. OODA Loops comprise four steps [27]:

1. **Observation.** The analyst gathers information from multiple sources, for example by interacting with the environment (e.g., scanning a subnet), looking at information publicly available (e.g., performing Open Source Intelligence), or simply watching as events unfold (e.g., stumbling upon a website that exposes error information).

2. **Orientation.** The analyst assesses all information they have, taking into account the old information, the newly acquired one (perhaps thanks to the Observation process), previous analyses and experience, and even their cultural background.

3. **Decision.** The analyst decides which action will be performed next. This decision is strongly influenced by the two previous processes, and in turn will influence the next Observation process.

4. **Action.** The analyst performs the selected action and acquires new information from it. This process is strongly influenced by all three previous processes, and in turn will influence the next Observation process.

From the description above, it should be clear why OODA Loops have applications in a vast number of fields: these are concepts that are implicitly learnt and executed by all humans from the birth, hence, we are very familiar with the whole process and we are easily able to adapt it to fit several contexts.

Therefore, we can also use OODA Loops to describe ongoing cyber attacks: to do so, Bautista [18] mapped all steps of the Lockheed Martin kill chain to the OODA Loop, coming to the following conclusions: (i) the Observation step can be mapped to both Reconnaissance and Action on Objectives, the former because the adversary is gathering information to attack their target, and the latter because after the whole chain, they can look for further exploits or targets to attack; (ii) both the Orientation and Decision steps are mapped to Weaponization, as the adversary has to analyse all the obtained information to decide how to attack the target; (iii) the Action step is mapped to Delivery, Exploitation, Installation and Command & Control, as these represent what the adversary does to specifically attack the target.

While OODA Loops are very practical for describing the decisional processes of the adversary and their consequences, we can see how any modification to the loop, for example adding a step, would radically change its structure and would not make it an OODA Loop anymore. Additionally, most

of the adversarial actions are mapped under the Act step of the loop, so it becomes increasingly harder for an analyst to pinpoint all the actions mapped under the Act step to a certain goal of the adversary, which is what we want to do in this work.

MITRE CAPEC

Common Attack Pattern Enumeration and Classification (CAPEC) [121] contains details, for each known weakness, on how it can be found and exploited. The main difference between CAPEC and ATT&CK is that the former is focused on the exploitation process on application security, while the latter describes the whole adversarial lifecycle, from initial recon to the final impact.

A CAPEC entry contains all information to exploit a certain weakness, sometimes even more than one at once. For example, CAPEC-63[1] contains details on how to exploit XSS weaknesses, and it is tied to both CWE-79 (XSS itself) and CWE-20 (Improper Input Validation). The description on how to exploit the vulnerability generally contains three steps: Explore, Experiment, Exploit. In the Explore phase, the adversary tries to find entry-points to exploit the vulnerability, such as user-controlled inputs. In the Experiment phase, the adversary tries different payloads to check whether the entry-points are sinks that allow exploitation. Finally, in the Exploitation phase, the adversary manages to exploit the vulnerability and is allowed to perform actions on the target. Each of these phases can comprise multiple steps.

From the above description, we can see how CAPEC could help in defining the steps that the adversary would have to perform to exploit the self-activation vulnerability. However, kill chains remain a more powerful tool to do so. The reason behind this is twofold. First, CAPEC is able to give a good insight on the process of exploiting of a vulnerability, but it does not give the chance to describe other steps of the attack, for example, how can an adversary chain this vulnerability to another to gain persistence, or how to actually

[1] `https://capec.mitre.org/data/definitions/63.html`

deliver payloads to the victim, or the techniques to use for weaponization. Although we could describe these processes in such a way to make them fit the Explore, the Experiment or the Exploitation phase, this is not regulated by CAPEC in any way and leaves most of these processes unformalised and their discussion subjective to who is filing the CAPEC entry. Second, the CAPEC entry is thought as a linear process to exploit a weakness: hence, it does not allow for cyclicality and it has no depth, which is something we discussed as a limitation of the original Lockheed Martin Kill Chain.

Howard's Model

In 1998, Howard and Longstaff [74] presented a taxonomy for computer security incidents. In their taxonomy, they distinguish between:

1. Security **events**, which are *actions* towards a *target*. Note that they are not necessarily malicious;

2. Security **attacks**, which are defined as a sort of chain. They start with a *tool* that is used to exploit a *vulnerability*, which in turn executes a security *event* (i.e., item 1 in this list) that leads to an *unauthorized result*. Hence, if there is an *unauthorized result*, then an attack took place;

3. Security **incidents**, which are security attacks (i.e., item 2 in this list) performed by an *attacker* to reach a certain *objective*.

Security attacks resemble what we have seen in kill chains, OODA loops and CAPEC, that is, a list of steps against a target to exploit a vulnerability and achieve objectives. In this sense, Howard's Model is not that different from a kill chain — furthermore, it also allows for ciclicality. In fact, a single iteration of the aforementioned "chain" of components is not enough to perform a complete attack.

For example, the adversary might want to *scan* (action) all machines on a certain *subnet* (target) to check for known vulnerabilities to exploit. In this

case, they perform a security event, and not directly an attack. Afterwards, they go back to the start of the chain, and they use *Metasploit* (tool) to *exploit* (action) the *buffer overflow* (vulnerability) that the scanner had found on an *exposed service* running on a machine in the subnet (target) to get a *reverse shell* (unauthorized result). In this case, the adversary performed a security attack or an incident, depending on whether the *adversary* has reached their overall *objective*. However, if the adversary wants to perform more malicious actions (e.g., getting persistence on the breached machine, performing lateral movement), they have to restart the chain again.

This means that the Howard's Model is not a tool to describe the different phases of an ongoing attack, but rather a tool to describe all the possible actions the adversary could do to perform an attack, in a fine-grained way. Hence, in the Howard's Model, the analyst does not care whether the adversary is performing Exploitation, Installation, Command & Control, Lateral Movement, or else — they only care about splitting the incident in a sequence of *actions* towards *targets*, and then analysing the context (*tool*, *vulnerability*, *unauthorized result*) to determine whether the adversary has performed an attack or not. However, this makes us lose the general vision of the attack, as for every action we do not know at which stage of the attack we are, as a real attack is potentially made of tens or hundreds of actions.

While we can certainly model every single action of our adversary in the self-activation context using this model, we will show in the next sections how epistemic modal logic [20] is a more powerful tool for this purpose, as it also allows us to model the bits of knowledge that the adversary and all the other actors involved in the attack have before, during, and after a successful attack.

Attack Graphs and Trees

Other models that allow us to define all actions that an adversary could perform during an attack are the attack graph [41] and the attack tree [139]. Lallie et al. [99] performed an extensive review of attack graphs and trees, finding

that they share enough similarities to be grouped under a single category, that is, the *graph-based* methods for attack modeling. Other categories in their taxonomy include *use-case* methods, which model attacks based on practical use cases, and *temporal* methods, which list steps of the attack to be performed one after another — in fact, kill chains fall in this category.

Both attack trees and graphs take into account the preconditions for the attack to happen, albeit in different ways, and with different levels of detail. In attack trees, the attack is described with a bottom-up approach, and an example is in Figure 3.1. At the top of the tree, we have the goal of the attack, in this case, getting admin privileges. As we descend the tree, we read what the adversary must do to achieve the goal in the parent level. For example, the adversary is able to get the admin's password if they either (i) get a password file, (ii) successfully perform brute-forcing, (iii) get to snoop on the admin while they are typing the password, (iv) get to install a keylogger on the admin's machine, or (v) bribe the admin to give them the password.

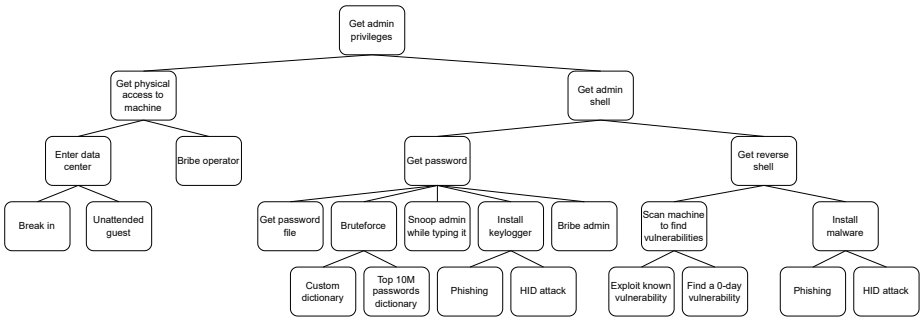

FIGURE 3.1: *Example of Attack Tree (Adapted and Expanded from [139])*

While in attack trees preconditions and actions are combined together to achieve the top goal, in attack graphs these are listed separately as two different types of nodes. Let's consider a vulnerable blog platform with three roles: admin, writer and user. The admin has access to the general control panel and is the highest role available, the writer can publish posts in the blog and can access a writer-only control panel, while the user can just search for posts and comment them. In Figure 3.2 we see a possible attack graph to al-

low a user to become admin, where preconditions are rectangle-shaped and exploits are ellipse-shaped. To begin with, the user can only comment blog posts or type queries. As the search form is vulnerable to SQL Injection, the adversary can just append a query that makes them become an administrator; another possible way is to exploit an XSS on the blog comments section, so that when a writer sees the adversary's comment, the XSS payload would activate and give the writer's session key to the adversary, who can now login as writer and access the writer control panel. The adversary can then exploit a logic vulnerability to access the general control panel, as its webpage only checks if the entity that is trying to access the page is not a normal user, but it is not checking whether it is an admin or a writer.

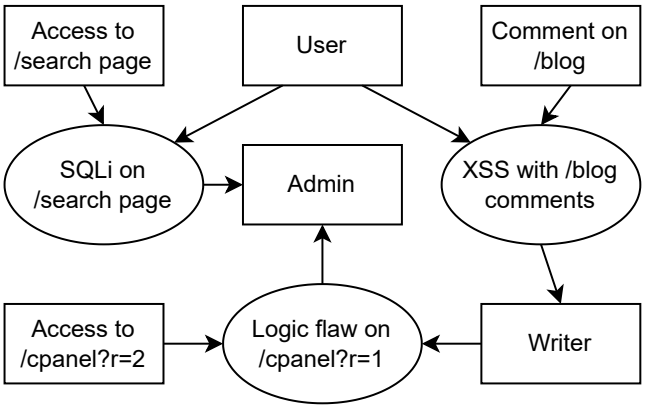

FIGURE 3.2: *Example of Attack Graph*

Even if both attack trees and graphs are powerful tools to understand preconditions for the different phases of an attack and all the possible actions, we will use epistemic modal logic for this purpose, as previously said, because it also allows us to better define the attacker's knowledge in a fine-grained way throughout the whole attack and with a sleek notation.

3.2.3 Threat Models

In literature, the threat model has been defined as *"a process that can be used to analyze potential attacks or threats"* [163, 153] and it has been widely used

to describe the conditions in which an attack takes place, or to prove properties of security protocols. We hereby make an exemplificative, non-exhaustive list of common threat models.

The Dolev-Yao model is perhaps one of the most popular models in the literature, designed to assess the security of cryptographic protocols that make use of public key encryption. In this model, the adversary has the ability (i) to intercept all messages between two parties that want to communicate in a secure manner, (ii) to modify these messages, and (iii) to use them in any conversation with said two parties, resulting in the powerful capability to completely control the communication channel [49]. In the General Attacker model, this concept is taken to the extreme, and every actor involved in the conversation is considered to be a Dolev-Yao attacker. This is interesting because every actor at this point can choose, at every step of a protocol, if they want to follow it or not, introducing two new concepts: (i) *retaliation*, that is, the capability of an actor to retaliate if someone else violates the protocol, hence introducing potential risks when deciding to pursue malicious actions, and (ii) *anticipation*, that is, leveraging another actor's misconduct to achieve personal goals [16].

Biggio and Roli [22] describe threat models in the machine learning environment by introducing first the attacker's goal. Then, they introduce the adversary's knowledge and their capabilities, that is, the actions they can perform. Finally, they use these data to express an optimal strategy for the adversary in the form of an objective function. It can be noted that, although this work explicitly uses the adversary's knowledge and capabilities as parameters to formulate the objective function, the other aforementioned works also make use of these concepts, albeit in a less visible way. Additionally, Biggio and Roli [22] also describe how such parameters change, depending on the considered scenario — for example, in the white-box scenario, the adversary knows all details of the target system (e.g., learning algorithms, hyperparameters, etc.), while in the black-box scenario, they know none of them.

In this thesis, we will model the adversary's knowledge and capabilities for the self-activation environment against VCDs by using epistemic modal logic [20], and we will assume a black-box scenario, in which the adversary knows nothing about the target system and the victim user.

3.3 The VOCODES Kill Chain

While several kill chains have been discussed to comprehensively describe all the necessary steps for an attack, they may not always be the most suitable for certain attack scenarios.

In the case of voice command self-issue attacks, we find that the Intrusion Kill Chain is sufficient to describe the attack steps in Chapter 4, as every step of the attack can be associated with a step in the Intrusion Kill Chain. However, other kill chains may not make sense or be necessary in this context. Therefore, we introduce the VOCODES Kill Chain, which tailors the Intrusion Kill Chain to the specific environment of voice-controllable devices.

The VOCODES Kill Chain outlines the steps necessary to carry out self-activations on voice-controllable devices and incorporates modifications to the Intrusion Kill Chain to make it more relevant in this context. Although the changes made to the Intrusion Kill Chain are not essential, they aid in better formalising the requirements and objectives of each step in self-issue attacks on voice-controllable devices. By using the VOCODES Kill Chain, security analysts can better understand and predict the actions of attackers targeting voice-controllable devices through self-issue attacks.

3.3.1 Steps

The VOCODES Kill Chain comprises six steps: Reconnaissance, Audio Weaponization, Initial Foothold, Exploitation, Persistence, and Actions on Objectives. Figure 3.3 illustrates a graphical representation of these steps and the possible cycles between them. The related steps of other kill chains for

each step of the VOCODES Kill Chain are listed on the right. While some of the steps of the VOCODES Kill Chain retain the same names as in the Intrusion Kill Chain, it is crucial to discuss their meaning in this new context. Notably, (i) the term "Weaponization" has been updated to "Audio Weaponization", (ii) Installation, Delivery, and Command & Control have been excluded, while (iii) Initial Foothold and Persistence have been included. In the subsequent sections, we will elucidate the rationale behind these modifications.

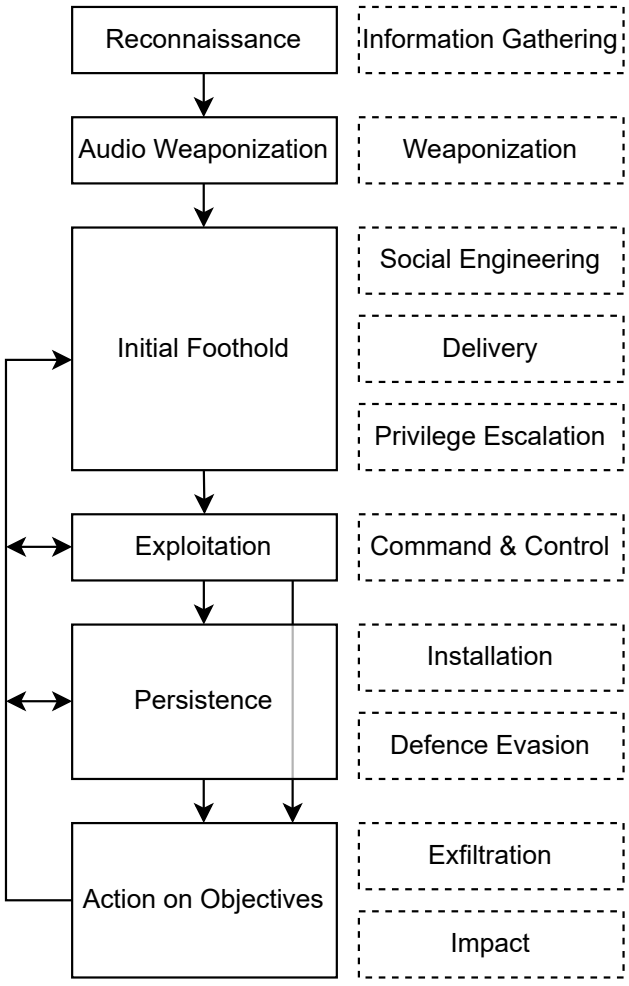

FIGURE 3.3: *The VOCODES Kill Chain*

Step 1. Reconnaissance

The reconnaissance (*recon*, for brevity) step starts by identifying and selecting targets for the attack. In the case of commercial voice-controllable devices, the adversary can obtain them by simply buying them online. However, this is not necessarily true for other devices running open-source Voice Personal Assistants. In fact, while some open-source VPAs such as MyCroft have their own commercial smart speaker [123], others such as Leon [67] do not. This means that while assessing the presence of the self-issue or other related vulnerabilities is relatively easy on commercial devices, as they share configuration and hardware, the adversary will have a harder time understanding if they can self-issue commands to an open-source VPA running on custom user-owned hardware. Hence, the attacker has to take this into consideration while listing the devices and the VPAs they want to attack.

The reconnaissance step does not terminate when the adversary has selected all target devices but continues with the identification of the related wake-words and possible actions the device can perform. For example, if the adversary wants to attack a device running the Alexa VPA, they have to know that possible wake-words include not only "Alexa", but also "Amazon", "Computer", and "Echo". Likewise, the command context is also important. For example, if the adversary wants to attack the BMW Intelligent Personal Assistant [25], they need to know not only that "Alexa" is most likely not a valid wake-word[2] but also that "turn on the microwave oven" does not make much sense within a car (as of today, at least!).

From the above discussion, it is easy to see why recon in this environment is substantially different from the general strategy that consists of looking for machines on a subnet, scanning ports to check running services, obtaining operating system fingerprints, etc., because the component that we want to attack, that is, the input voice channel, is already known. Once the attacker

[2] Cunningly, BMW IPA allows the user to choose a custom wake-word, so "Alexa" could potentially be a valid wake-word.

knows how to activate such an input channel and what the set of valid commands is, they can proceed to the next step.

Step 2. Audio Weaponization

In the context of voice command self-issue, weaponization does not involve writing code to directly exploit a vulnerability found in the previous step. Instead, the adversary's goal is to generate a malicious audio file containing an arbitrary voice command crafted for the specific context. This step involves crafting the audio file to include the wake-word followed by the command.

However, simply generating an audio file with the desired contents may not reliably trigger the self-issue vulnerability. As Chapter 4 will show, the voice profile used to generate the commands plays an important role in exploiting the self-issue vulnerability. Some voice profiles generate commands that are more easily recognised by certain Voice Personal Assistants.

Therefore, it is worth investigating the acoustic voice properties that characterise human voices and how an adversary can manipulate them to create audio files that are easily recognised by the target VPA. According to Dasgupta [42], the human voice has four main attributes: pitch, loudness (or sound pressure), timbre, and tone. Although speech rate is not among these main attributes, it is still useful for our purposes, as the time gaps between words can be used as a feature for detecting human emotions. Thus, we summarise the five acoustic voice properties that an adversary should manipulate to generate voice command audio files:

- **Pitch**: the perceived frequency of vibrations emitted by a sound, measured in Hz.
- **Loudness**: strongly related to sound pressure, measured in dB.
- **Timbre**: the perceived quality of a sound, based on the type of sound production. It has no measurement unit, but a digital representation can be found by analysing the shape of a soundwave and its spectrum.

- **Tone**: the variation of pitch in language, used to distinguish or emphasise words. It has no measurement unit.

- **Speech rate**: the speed at which a sentence is pronounced, measured in words per second or as the average time gap between words, in seconds.

While an adversary could manually record their own voice commands and test their effectiveness on the target VPA, this approach is not scalable or easily automated. To generate malicious audio files, attackers can instead utilise various text-to-speech (TTS) services available online and adjust the five acoustic voice properties discussed earlier by modifying the parameters and variables exposed by the TTS service. For instance, Azure TTS [116] offers customisable settings such as:

- **Voice Profile**: a preconfigured voice that reads out the intended text.

- **Pitch**: the frequency of the voice used to read the text. It can be adjusted by the user and corresponds to the *pitch* property mentioned earlier.

- **Style**: the manner in which the text is spoken, with options like *cheerful*, *sad*, and *friendly*. It determines the *tone* of the voice.

- **Speed**: the playback speed of the text. The default speed is 1 but it can be increased or decreased by the user, corresponding to the *speech rate* property discussed earlier.

Using these TTS services to generate malicious audio files, attackers can experiment with different voice profiles and settings to craft voice commands that are more easily recognised by specific VPAs. While some TTS services only allow customisation of a limited set of voice properties, Azure TTS offers additional flexibility through the use of Speech Synthesis Markup Language (SSML) tags. By using SSML, users can customise the five acoustic voice properties in more detail:

- The **pitch** attribute of the `prosody` SSML tag allows for even more precise customisation of the *pitch* property, giving the user the option to choose between absolute values (in Hz), values relative to the current setting (specified as a variation in Hz or as a percentage), or constant values from a preset list.
- The **volume** attribute of the `prosody` SSML tag allows for customisation of the *loudness* property, by setting an absolute value (with 0 being the quietest and 100 the loudest), a relative value, or a constant value.
- The **styledegree** attribute of the `mstts:express-as` SSML tag allows for the intensity of the chosen *tone* property to be adjusted further. For instance, an utterance within the tag `<mstts:express-as style='terrified' styledegree='3'>` will sound more scared than one within an element with `style='terrified'` and `styledegree='1'`.
- The **rate** attribute of the `prosody` SSML tag allows for customisation of the *speech rate* property, which can be set to a relative value (with 0.5 indicating half the normal speed and 2 indicating double the normal speed), or a constant value.

It is worth noting that some TTS services, such as Amazon Polly, allow customisation of the *timbre* property as well. Amazon Polly achieves this by using the `<amazon:effect vocal-tract-length>` SSML tag for the standard TTS format.

Once an attacker has crafted several payloads with customised voice properties, they must test them to ensure they trigger the self-activation of the target VPA. If the payloads are not successful, the attacker must adjust the parameters until a successful combination is found.

Step 3. Initial Foothold

Initial Foothold is the step in which the attacker gains the necessary privileges to play an audio file containing a voice command payload on a target voice-

controllable device, thereby delivering it to trigger the self-issue. As previously mentioned, the self-activation vulnerability can only be exploited on devices that can play audio files and accept voice commands concurrently. Chapter 2 explained that the Initial Foothold is a crucial requirement for all attacks on Voice-Controllable Devices, such as using a rogue speaker near the target device or a PZT transducer within the room. We identified two methods by which an attacker can achieve the Initial Foothold in the self-issue scenario: social engineering and temporary access.

In the case of *social engineering*, the attacker must trick the user into executing a malicious application that provides the attacker with the ability to play weaponized audio files. If the target VPA runs on a personal computer, social engineering tactics may include sending phishing emails, links to scam websites, or shipping malicious USB devices, with the expectation that the user will execute the malicious application or insert the USB device. However, if the user can only interact with the device via the voice interface, as with smart speakers, the social engineering strategy requires making the user run a malicious VPA application. There are two primary methods to do so:

- **Convincing the user to run the malicious application**: this approach might work with inexperienced users who may run the application for any reason, such as applications claiming to offer "free money", "music", or "feedback".
- **Performing a Squatting Attack**: in this method, the attacker exploits the fact that ASR systems often misclassify one phoneme with a similar one, resulting in the transcription of a different word. The attacker deploys an application to intercept the misclassification of the application name by the VPA, which opens the malicious application instead of the intended one, such as PayPal/PayPaul [95].

In the case of *temporary access*, the attacker can use the device for a limited time, either with the user's permission or when the device is unattended.

The attacker can then surreptitiously execute the malicious application. This is reminiscent of privilege escalation techniques in which temporary, unauthorised, or non-privileged access is utilised to gain higher privileges on the target. Moreover, if the attacker has temporary physical access to the target device, connecting a device they hold may be sufficient to play audio files without the need to execute an application on it. For instance, it is feasible to connect any Bluetooth device to a smart speaker and play any audio file.

Once the target device executes the malicious application or connects to the attacker's device, it is *de facto* connected to an audio *C&C* server that the attacker can use to issue voice commands to the target. It should be noted that this operation generally does not require any user privileges on the target device, as playing audio files is usually permitted. Furthermore, identifying that the connection to the audio streaming service or the attacker's device is malicious would be challenging for antivirus software or any other protection mechanism. This is because these actions are innocuous in nature, and the malicious component in the communication is the voice command within the played audio files, which is challenging to fingerprint due to the variety of TTS services available on the internet and the possible customisation of the weaponized audio files seen in the previous step.

Step 4. Exploitation

In the context of VPA self-issue, the exploitation process does not differ much from its classical interpretation. During this step, the audio payload sent via the *C&C* server is executed on the target device, exploiting the self-issue vulnerability. Once the payload is executed, any command contained within the audio files played by the voice-controllable device is captured by its microphone and executed successfully. As a result, the adversary gains the ability to issue any permissible voice command to the device through the remote or local audio streaming service.

However, if the target device gets disconnected from the malicious

streaming server for any reason, the attacker would no longer be able to issue any more commands to the device, thus ending or pausing the attack. To avoid this, the attacker may try to establish persistent access to the device.

Step 5. Persistence

Similarly to the Exploitation phase, the Persistence step also shares some similarities with its classical interpretation, depending on the target device that the attacker wants to establish permanent access on. For instance, on a Windows device, the malicious application already running on the target device can use the Windows Service Control (`sc.exe`) to create a service containing the malicious application, which would run every time the machine is booted up. On Linux systems, a similar strategy involves using `systemctl`.

However, even if the traditional malware cannot be executed, and the attacker is only capable of self-issuing voice commands, they can still achieve persistence on the victim device. For example, on Android systems, the attacker can use the Voice Access feature to perform any permissible operation, such as opening applications, tapping buttons on the screen, typing in search bars, visiting websites, etc. The attacker can then download and execute any rootkit available for the device to gain persistent root access to it.

While this is not possible with smart speakers, as their applications run on the cloud instead of the device, the attacker can protect the malicious audio streaming by running a silent application on top of it, or by running an application that performs a Voice Masquerading Attack to avoid detection by the user. For instance, if an Echo device is playing a malicious radio station and a skill is subsequently opened, if the user says “Alexa, stop”, this command would only close the skill, but not the malicious radio station, enabling the attacker to retain control of the device.

Step 6. Actions on Objectives

Once the attacker gains the ability to issue any permissible voice command to the device and has possibly established permanent access to it, they can perform a range of malicious operations on the device itself, such as buying items, tampering with calendars and local files, sending emails, setting up or dismissing alarms, and more. If the target is a smart speaker, the attacker can also potentially control other connected smart appliances, allowing them to manipulate heating systems, unlock smart locks, turn off lights, and perform actions that could put the user's physical safety at risk.

In the context of VCD attacks, attackers typically have one or more of these three objectives: (i) violating the user's privacy by obtaining sensitive information like passwords, PINs, or Personally Identifiable Information; (ii) executing malicious actions for financial gains, such as purchasing premium features from applications owned by the attacker or transferring funds to the attacker's account via PayPal; (iii) undermining the user's physical safety, such as by opening their smart locks, turning on heating during a hot day, or turning off lights during the night.

3.3.2 Discussion

Unlike the Lockheed-Martin Kill Chain, our VOCODES Kill Chain does allow for cyclicality of actions. In fact, the attacker may need to repeat one or more steps from 3 to 6 in response to events that occurred during the attack. For instance, if the attacker failed to establish persistent access or skipped the persistence step for any reason, and the device gets disconnected from the C&C server, the attacker may need to restart the attack from step 3. This is also the case when the attacker wants to target multiple devices of the same family — they will need to repeat steps 3 to 6 for each device.

The reader may have observed that not all steps of other kill chains are connected to a step of the VOCODES Kill Chain. This is because they do not

fit into the structured process that the attacker must follow to execute a self-issue attack against a voice-controllable device. These steps include:

- **Pivoting [UKC]**: even if the attacker can control smart appliances by self-issuing voice commands to a smart speaker, they cannot attack them or exploit their vulnerabilities. Moreover, the adversary does not have control over the tunneled traffic via such smart appliances or other connected devices. Hence, this differs significantly from a classic pivoting scenario.

- **Discovery [UKC, ATT]**: although the attacker can self-issue commands that prompt the user to disclose any smart appliances they have, such as setting up an application that intercepts all legitimate voice commands, this is a very limited internal reconnaissance tool that is closer to *social engineering* than *discovery*.

- **Lateral Movement [UKC, ATT, EKC]**: while the adversary can issue commands to smart appliances connected to the target device, this is not considered proper lateral movement since the adversary does not gain a shell or any access to them.

- **Collection [UKC, ATT]**: the adversary does not have to collect information before exfiltration since the attacker should always be listening to or recording the legitimate user's utterances, so any sensitive information is captured the moment it is spoken by the user.

- **Exit [MKC]**: the authors of the paper [89] where this step is presented unfortunately did not elaborate on its meaning. We can assume that it implies that the adversary may want to stop the attack at some point, either because they achieved their mission goal or to avoid detection. However, this is not necessarily true: there are numerous attacks, especially those including C&C and zombie devices, where the attacker's goal is to gain control of as many devices as possible, and being de-

tected is part of the game. In other words, the attack never ends. In the VPA self-issue context, we have a similar scenario: assuming that the attacker is gaining control of voice-controllable devices remotely, they ideally do not want to terminate the attack since they would stop gathering sensitive information and give up control of all breached devices. While an adversary could proceed with this step at any time during the attack, it is not *ideal* or *necessary* for the VOCODES Kill Chain.

Furthermore, to simplify the model, we have excluded several steps from other kill chains that were redundant with existing terminology.

- **Execution [UKC, ATT]**: code execution can occur during the *initial foothold* or *persistence* phases, but we do not consider the exploitation of the self-issue vulnerability as code execution, as it involves executing an attacker-controlled *voice command* rather than *source code*.

- **Credential Access [UKC, ATT]**: the adversary can obtain any passwords or PINs entered by the legitimate user during the *action on objectives* phase, but this is covered by the *exfiltration* step.

- **Resource Development [ATT]**: the adversary requires two resources for the attack: the malicious application and weaponized audio files. While developing the former is not explicitly mentioned in the VOCODES Kill Chain, because the adversary could use an existing solution to cast a radio station or use Bluetooth streaming to issue voice commands, the latter is already included in the *audio weaponization* step, making this step redundant.

- **Manipulation [EKC]**: manipulation includes further *reconnaissance*, *weaponization*, *exploitation*, *installation*, and *execution* on the main target. In the Expanded Kill Chain, this step was relevant because the adversary had performed these actions on other scopes (i.e. external, internal) and it was time to execute them on the target. However, this is

redundant in the VOCODES Kill Chain, as there are no multiple scopes and the adversary is already executing all these actions on the target.

In summary, we excluded these steps for simplicity and clarity in VOCODES.

3.4 The VOCODES Threat Model

The VOCODES Kill Chain provides a clear guide on *what* actions an attacker should take — and in which order — to attack and gain control of a voice-controllable device. Therefore, we will now introduce more formal details on *how* the adversary must perform the required actions and *who* the actors are during the attack. This will involve defining a threat model for self-issued commands on voice-controllable devices. As it is common practice to design and describe threat models by formalising the attacker's *knowledge* and *capabilities* in the given scenario, we will use epistemic modal logic [45] to achieve this, while modelling actors and their interactions with the environment and devices. Our syntax will be very similar to the one used by Bella et al. [20], with the addition of explanations for the different statements as we introduce them.

It is important to note that we only need to model knowledge and capabilities of all actors involved: this is to better understand the attack and be able to formally describe any condition of all actors at any given time during the attack. As epistemic modal logic gives us exactly the ability to do so, we do not need to perform any more threat modelling activities, such as threat identification. The reason behind this is that the threat we are facing is already known, that is, the self-activation attack. Hence, we do not need to implement and use threat identification frameworks such as STRIDE [115] or other tools such as the already discussed Attack Trees [139], which could potentially help in understanding the capabilities of an attacker, but could not help as much in understanding or defining their knowledge. Later steps of classical risk analysis activities, such as cyber risk quantification (i.e., evalu-

ating impact and feasibility of the attack) and risk management (i.e., identifying if the impact of the attack can be mitigated, avoided, delegated, or must be accepted — in our case, we chose to mitigate it because it was not possible to avoid or delegate it, and accepting it is the worst possible scenario) are discussed respectively in Chapters 5 and 7.

3.4.1 Modelling Actors and Devices

In our scenario, we only consider human actors, whom we identify as a. Actor *Alice* is the victim user and does not pose any threat to the VCD, however, we will see that some of her actions can influence the outcome of the attack. She is the owner of a VCD, which is the target of every attack. Although in Chapter 4 we discuss a practical attack targeting Alexa on Amazon Echo Dot devices, the claims made in this section are general and apply to any voice-controllable device. Hence, we use p to identify a generic Voice Personal Assistant that runs on the voice-controllable device owned by Alice:

$p ::= Alexa \mid Cortana \mid Google \mid Siri \mid \ldots$.

The three dots at the end indicate that our list is not complete, as only the most famous commercial VPA software has been listed for brevity. The other actor is *Eve*, the adversary. Eve is the only actor who poses a threat to Alice, as she tries to attack and gain control of her voice-controllable device. Eve can be either physically near such a device or she can try to attack it from a distance:

$a ::= Alice \mid Eve.$

$conn ::= local \mid remote.$

3.4.2 Modelling Actors' Knowledge

In our threat model, if an actor a has *knowledge* of a piece of information i, then i must be the truth. In other words, once a acquires any information, they cannot be mistaken about it. The concept of *knowledge* is opposed to the concept of *belief*, which allows a to be mistaken (i.e., allows i to not be

true) [28, 17, 76, 58], however, it is not indispensable in the current version of our threat model. Hence, when we want to formalise that a knows i, we write $[[a]]i$. This allows us to model knowledge of our actors, starting from Alice, who knows sensitive information that Eve would be interested in stealing. Some of this information allows Alice to use sensitive functions of her Voice Personal Assistant p, such as banking or health applications — in this case, the secret would be a PIN or a password. In other cases, the information is sensitive in nature, as happens with Personally Identifiable Information (PII):

$s_k ::= PIN \mid password \mid PII \mid \ldots .$

$S ::= \{s_k,\ k = 0 \vee k \in \mathbb{N}\}$

$[[Alice]]s\ \forall s \in S.$

Hence, Alice knows all the information included in the set S. If k is 0, this means that the set of secrets is empty, as Alice could, in fact, have no secrets to share with her voice-controllable device, either because she does not use functionalities that require her to do so, or because she is not willing to share them with her VPA, for any reason. As S is the set of Alice's secrets, it follows that Eve does not know them initially:

$\neg\ [[Eve]]s\ \forall s \in S.$

Eve, however, *does* know what VPA she will have to attack, as she chooses beforehand the device or the set of devices she wants to target. Hence, she can start the setup of the attack with the information $[[Eve]]p$. In fact, Eve might not even know who Alice is, in the case of $conn == remote$. Depending on the chosen attack vector, Eve might even target multiple victim users.

Concerning the Automatic Speech Recognition software underlying every voice-controllable device, and thus every Voice Personal Assistant software, we assume that neither Alice nor Eve have any detail about the ASR's machine learning model. To formalise this, we use the definition by Biggio and Roli [22], who characterise the attacker's knowledge of machine learning models in terms of a space Θ made of four elements: the training data $\mathcal{D}$, the feature set $\mathcal{X}$, the learning algorithm f, and the (hyper)parameters w. Assuming

that these elements refer to the ASR algorithm used by p, in our notation we formalise the described black-box scenario as:

$\neg\ [[Alice, Eve]]\ \mathcal{D}, \mathcal{X}, f, w.$

Table 3.1 summarises the above statements on actors' knowledge.

TABLE 3.1: *Actors' Knowledge within our Threat Model*

Rule	**Description**
$[[Alice]]s\ \forall s \in S.$	Alice knows all her secrets.
$\neg\ [[Eve]]s\ \forall s \in S.$	Eve does not know any of Alice's secrets.
$[[Eve]]p$	Eve knows which VPA to attack.
$\neg\ [[Alice, Eve]]\ \mathcal{D}, \mathcal{X}, f, w.$	Neither Alice nor Eve know the ASR model's details.

3.4.3 Modelling Actors' Capabilities

In our threat model, each actor has unique actions they can perform to alter the environment and the state of the VPA p. Intuitively, Eve's actions are performed with the intent of attacking p, while Alice's actions are supposedly harmless, as they are executed during normal interactions with the environment and the device. We formalise that an actor a can execute an action f in the following notation:

$[a]f(arg_1, arg_2, ..., arg_n).$

In this notation, arg_n is an argument[3] of the action f, and $n \in \{0 \cup \mathbb{N}\}$. Note the single square brackets, which indicate who is executing the action, as opposed to the double square brackets already introduced in the previous section, which indicate who knows some information.

Modelling Alice

Alice can interact with her device p as much as she wants to. She can do so by issuing a voice command cmd to her device. Once Alice has issued the

[3]Hence, we execute actions as if they were functions in pseudocode. In this chapter, we will use the terms *action* and *function* interchangeably.

command, p receives it and executes it. It follows that p gains knowledge of said command as well:

$$\forall p, cmd \in C.\ [Alice] giveCommand(p, cmd) \implies [[p]] cmd.$$

In this statement, C is the set of all commands ever given by Alice. Depending on the functionality requested by Alice with her command, the device could ask for a secret s, such as a password or a PIN. In fact, to protect sensitive information, some VPAs require the user to preemptively set up a security code to authenticate the user whenever access to such information is subsequently requested. Alternatively, the device could ask for PII, for example, it might ask to confirm the shipping address for the order that was just placed. We formalise Alice sharing a secret s (and subsequently p knowing it) as follows:

$$\forall p, s.\ [Alice] shareSecret(p, s) \implies [[p]] s.$$

Note that the implication symbol $\implies$ does not necessarily mean that p has *just* learnt about s — it only means that p knows it. In fact, when sharing a password, p needs to already know the password to check if it matches the one that was just pronounced by Alice; however, in other scenarios, p might learn something new (a new shipping address, for example).

Alice can also run applications on her voice-controllable device. These applications can be run in the form of executables that run on the device itself (as it happens on Windows and Android systems), or in the form of cloud services, where p only takes the input command from the user and then reads the reply aloud (as it happens with Alexa's skills or Google's actions). We formalise as $APPS_p$ the set of all available applications for p. Moreover, p knows the application that Alice runs on it, as formalised within the following:

$$\forall p, app.\ [Alice] run(p, app, method) \wedge method == voice \implies [[p]] app.$$

Note that the run function implies $[[p]]app$ only if app was run by using a voice command (as indicated by the third argument, $method$, in the function). There are other ways to open applications, such as double clicking an executable, clicking a hyperlink, etc., and p might know app in these cases as

well under certain circumstances, but we do not formalise them because they are not useful for our purposes.

Alice is not always around the device running p. Hence, at times she will not be at home, or she will be far from the device. We consider Alice to be *away* if the device running p is at least at 7m of distance from her. Further details on why we chose 7m as a threshold are given in Chapter 5. It follows that, if Alice is away, she cannot hear anything that is being output by p at the same moment. This can happen, for example, if Alice asked p to remind her of something on a certain date, but she is not at home when the reminder is triggered. If we denote with $[p]say(output)$ any voice output from p, we get:

$$\forall p, output.\ [Alice]away(p) \implies \neg[[Alice]][p]say(output).$$

Vice versa, from the previous statement, we also get that if Alice is not away — i.e., she is in close proximity of the device — she can logically hear any output from p. Hence, we get the following statement:

$$\forall p, output.\ \neg[Alice]away(p) \implies [[Alice]][p]say(output).$$

Although other actions on the device might be undertaken by Alice (e.g., increasing or lowering the volume, shutting down the device, listening to music, etc.), we emphasise that the purpose of this section is not to explain formally all possible actions that can be performed by involved actors, but only those that are necessary to describe the self-issue attack in a precise fashion.

Table 3.2 summarises Alice's capabilities discussed above.

Modelling Eve

Eve cannot interact freely with p. As said in the previous section, if $conn == remote$, Eve might not even know who Alice is, and where p is placed. Nonetheless, if $conn == local$, Eve *does* know Alice and p's location and can act accordingly. In any case, all actions that will be defined for Eve will apply to both situations. Eve starts planning the attack by choosing p, and generating different voice commands to self-issue:

$$\forall p, cmd.\ [Eve]genCmd(p, cmd) \implies [[Eve]]cmd.wav.$$

TABLE 3.2: *Alice's Capabilities within our Threat Model*

Rule	**Description**
$\forall p, cmd \in C.\ [Alice]giveCommand(p, cmd) \implies [[p]]cmd.$	Alice can give commands to the VPA, implying that the VPA will know them.
$\forall p, s.\ [Alice]shareSecret(p, s) \implies [[p]]s.$	Alice can share her secrets with the VPA for verification or setting purposes, implying that the VPA will know them.
$\forall p, app.\ [Alice]run(p, app, method) \wedge method == voice \implies [[p]]app.$	Alice can run any application for VPAs with her voice, implying that the VPA will know the running app.
$\forall p, output.\ [Alice]away(p) \implies \neg[[Alice]][p]say(output).$	Alice can move away from the device, implying that she will not hear anything said by the VPA.
$\forall p, output.\ \neg[Alice]away(p) \implies [[Alice]][p]say(output).$	If Alice is not away from the device, she will hear any sound output by the VPA.

After the execution of the $genCmd$ action, Eve has at least one .wav file containing the desired voice command. Knowing p is crucial for this step because every voice command is made of two elements: the wake-word and the command. The wake-word is a phrase that activates the VPA, such as "Hey Google". It can also be a word such as "Alexa". Once the VPA captures the wake-word, it starts recording the command, which will be sent to the ASR for analysis. Hence, knowing p means knowing the related wake-word — a command without a valid wake-word will not be detected by the VPA, hence, it will not be executed. It is true that some VPAs, such as Alexa, allow the user to change the wake-word, making them choose between a small set of words, such as "Echo" or "Computer". However, because the given set is very small, it is feasible for Eve to generate commands for all valid wake-words of p, even if she does not know which one was actually selected by Alice. On the contrary, Eve cannot generate commands for all values of p, because the set of the valid values for p (that is, the set of the existing VPAs) is indefinitely large.

Even if Eve generated such .wav files containing voice commands, she still cannot make p self-issue them. If $conn == local$, Eve will have to find a local

vector to self-issue commands to p. If $conn == remote$, the easiest option for Eve to self-issue commands to p is to develop malware that exploits the self-issue vulnerability and make Alice run it via social engineering. In other words, Eve needs to deploy a malicious application on p's application store, if it has one, or to make it available by other means. If we call mal that application, and recall that $APPS_p$ is the set of the existing applications for p, then we formalise the deployment of mal as:

$$\forall p, mal.\ [Eve]deployApp(p, mal) \implies mal \in APPS_p.$$

A real instance of how Eve can make Alice run the application, and of how Eve can establish a local connection with p is discussed in Chapter 4. In general, malware may be delivered by other means, e.g., on Windows systems, malware is usually not an application for Cortana but for Windows itself, and on Android systems, malware is not necessarily a Google VPA application but software that runs directly on the operating system.

Finally, Eve sets up a Command & Control server to give commands to all devices that run the malicious application if $conn == remote$, or connected to the attacker by other means (Bluetooth, for example) if $conn == local$. In Chapter 4 we will show a real implementation of both scenarios against Amazon Echo Dot devices. Hence, if D is the set of the devices connected to the attacker (with any $conn$ value), and $p_d \mid d \in D$ is a device that runs p:

$$\forall d, conn, cmd.wav.\ [Eve]c2Server(d, conn) \implies [Eve]giveCommand(p_d, cmd.wav).$$

3.4.4 Attack Success

We assume Eve's attack to be successful if she permanently gains the privilege to execute any command on p, which was an ability unique to Alice. In other words, the attack is successful if:

$$\forall p, cmd.wav.\ [Eve]giveCommand(p, cmd.wav)$$

Eve could then perform any malicious operation on p, for example, she could buy items on Amazon on behalf of Alice, or turn on any smart appliance in Alice's home, potentially undermining her physical safety if heating, microwave ovens, smart locks, etc. are maliciously operated. Note that in the *temporary access* scenario described in the VOCODES Kill Chain, the above statement is not true (hence, the attack is not successful yet) because the permission given to Eve to use the device is not permanent. Table 3.3 summarises the aforementioned Eve's capabilities and her goal.

TABLE 3.3: *Eve's Capabilities within our Threat Model*

Rule	**Description**
$\forall p, cmd. \quad [Eve]genCmd(p, cmd) \implies [Eve]cmd.wav.$	Eve can generate audio files containing malicious voice commands for the VPA, implying she has an audio file with malicious audio content.
$\forall p, mal. \ [Eve]deployApp(p, mal) \implies mal \in APPS_p.$	Eve can deploy malicious applications for the VPA.
$\forall d, conn, cmd.wav. \ [Eve]c2Server(d, conn) \implies [Eve]giveCommand(p_d, cmd.wav).$	Eve has a C2 server that she can use to remotely control one or more devices running a VPA.
$\forall p, cmd.wav. \ [Eve]giveCommand(p, cmd.wav).$	The attack is successful if Eve manages to permanently obtain the privilege to issue commands to the target VPA.

3.5 Summary

We discussed kill chains for cyber attacks and derived a new kill chain that is tailored to the voice command self-issue scenario — the VOCODES Kill Chain. We discussed its six steps, detailing how they differ from their most classical counterparts. We also briefly mentioned other steps of relevant kill chains and explained why they do not find a place within the VOCODES Kill Chain. Finally, we formalised the attack scenario within the VOCODES Threat Model using epistemic modal logic, defining the knowledge and capabilities of the actors involved: the legitimate user, Alice, and the adversary, Eve.

Chapter 4

The Alexa versus Alexa Attack

In this chapter, we explore in-depth the voice command self-issue attack. We first introduce Alexa versus Alexa (AvA), an attack that leverages self-activation alongside two additional vulnerabilities we found on Amazon Echo Dot devices by applying the VOCODES Kill Chain. Then we compare AvA with other instances of the self-issue attack in Windows and Android systems, and with other similar work. Contents of this chapter have already been partially discussed in the following paper:

- Esposito S., Sgandurra D., Bella G. Alexa versus Alexa: Controlling Smart Speakers by Self-Issuing Voice Commands. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '22)*. 2022.

4.1 Introduction

Our attack, "ALEXA VERSUS ALEXA" (AvA), is the first to exploit the self-issue vulnerability on Echo devices, allowing an attacker to issue arbitrary commands and to control such devices for a prolonged amount of time. Within the previous sections, we have already seen that attacks against Voice-Controllable Devices need an Initial Foothold, that is, they usually require an external speaker or other equipment in proximity of the target device. With

this attack, we remove the necessity of having rogue equipment near the target device, increasing the overall likelihood of the attack. AvA still needs an Initial Foothold, albeit a stealthier one. In fact, the attack starts when the Echo device begins streaming an audio file that contains voice commands, and this can be done, for example, by opening a malicious *skill*[1] that makes the Echo device tune in a radio station that streams such malicious audio files: in this case, getting the user to run the malicious skill would be a possible Initial Foothold for the AvA attack.

We show that, with AvA, the attacker can exploit the self-issue vulnerability on the Echo device to make it perform any permissible action, such as controlling smart appliances in the victim's household (e.g., lights and door locks), calling any phone number or starting other skills. We also illustrate how the adversary can leverage AvA to open another malicious skill, which is able to gather user commands and spoof other skills' behaviour, hence performing a Voice Masquerading Attack [177]. This allows the attacker to execute further Actions on the Objective, such as personal data theft. Furthermore, we demonstrate that the adversary is able to keep the malicious skill running for a prolonged amount of time, independently from user interaction, establishing Persistence on the Echo device.

We now begin to look at the AvA attack through the VOCODES Kill Chain, starting from Section 4.2 onwards. Recall the epistemic modal logic notation introduced in Section 3.4, as we will use it to formalise the actions and the achievements of the adversary throughout the attack.

4.2 Reconnaissance

In the Reconnaissance step, we tried to get as much information as possible on the voice personal assistant we wanted to attack, that is, Amazon Alexa.

[1] Henceforth, we will always refer to applications for VPA as *skills*, which is the name used in the Amazon Alexa's context. For the sake of brevity, we sometimes say that the skills "run on the Echo device", however, skills actually run on their cloud hosting.

Hence, we get:

$p ::= Alexa.$

First of all, we assessed the devices on which Alexa runs, to obtain a target voice-controllable device, and found that such voice assistant is embedded on lots of different devices such as smart TVs and headphones. However, Alexa also has a dedicated smart speaker that is largely deployed worldwide: Amazon Echo Dot. When we began working on this study, 3rd Generation Echo Dot devices were already available for purchase, while 4th Generation ones were released only slightly after. Hence, we targeted 3rd Generation Echo Dot devices for our tests, and we write the following:

$p_d ::= EchoDot3.$

Continuing, we assessed the possible wake-words for our device, discovering that the user is able to select between four of them: Alexa, Amazon, Computer, and Echo. They can be discovered from the Amazon Alexa Companion App, by entering the Echo device's settings and selecting "wake-word". As the set of wake-words is shared between all Echo devices, the adversary can get them from the control panel of their device, and then use them to generate valid commands for any other Echo device. However, these wake-words may change depending on the device's language and its geographical position.

We then assessed the possible commands that can be issued to the smart speaker, discovering there are three main categories of commands:

- Commands that use internal functionalities of Alexa or of the Echo device. Examples: *"Alexa, what time is it?"*, *"Alexa, set a 10 minutes timer"*, *"Alexa, add tomatoes to my shopping list"*.

- Commands that control smart appliances within the household. Examples: *"Alexa, lights on in the living room"*, *"Alexa, make me a coffee"*.

- Commands that make use of third-party applications (skills) or services. Examples: *"Alexa, check my balance on Capital One"*, *"Alexa, play rock music from Spotify"*, *"Alexa, search who Ada Lovelace was on Wikipedia"*.

We then tried to look for specifications or details on the implementation of the ASR system adopted by Alexa, however, we were not able to find them, as Amazon has not disclosed them as of today. This is coherent with the VOCODES Threat Model, and we get the following:

$\neg\,[[Eve]]\,\mathcal{D}, \mathcal{X}, f, w.$

Being the attackers, in this chapter we will always assume to play the role of Eve. Finally, we assessed the conditions in which the device can operate. The Echo device can be placed anywhere inside the house, however, as soundwaves emitted by Echo are reflected differently if there are obstacles nearby, we consider the three scenarios depicted in Figure 4.1:

- **Open Scenario**: there are no obstacles near Echo, as it happens on a conference table;

- **Wall Scenario**: Echo is placed near a wall, and the distance from the wall is approximately 1.5cm to 4cm, while the closest obstacle is farther than 8cm;

- **Small Scenario**: Echo is placed on a surface with other objects on it (the wall can count as an object), and the distance from at least 2 obstacles must be 1.5cm to 8cm.

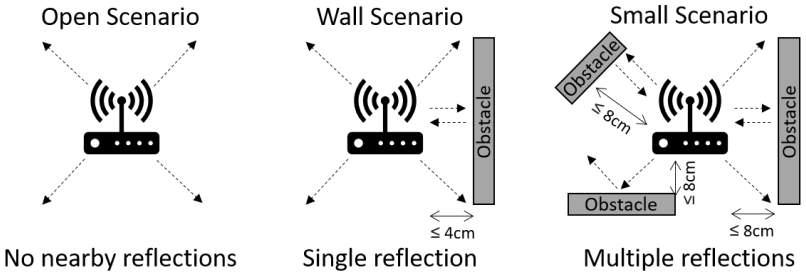

FIGURE 4.1: *Soundwave Reflection in the Different Scenarios*

4.3 Audio Weaponization

As discussed in Section 2.5.3, we have five category of voice spoofing attacks: impersonation, replay, voice conversion, speech synthesis and adversarial noise. We can observe that, among these, impersonation and replay attacks are performed using real voices, while voice conversion and speech synthesis consist in artificially generating an audio sample with a certain voice. We then have adversarial perturbations that do not fall in either of the previous two categories. Hence, we identify three possible ways to generate voice commands that can be self-issued:

1. **Text-To-Speech (TTS) Commands**: generated via any TTS solution.

2. **Adversarial Noise Commands**: generated via solutions that output adversarial noise samples working against Alexa *over-the-air*.

3. **Real-Voice Commands**: recorded by the attacker with their own voice or with someone else's.

In the list above we do not explicitly mention voice conversion solutions and impersonators because they are meant to attack the speaker recognition system and not the ASR system as the self-activation attack tries to do — we did not test them, however, their payloads should theoretically work as self-issued commands. In this case, voice conversion solutions would fall in the TTS category, while impersonators would fall in the Real-Voice category.

With regards to the first payload type, henceforth we refer to Google TTS to generate malicious audio commands and evaluate AVA. When not explicitly specified, it is assumed that we are using a voice command generated with pitch set to 0.00 and speed set to 1.00, with no SSML tags that can affect the TTS pronunciation and timbre. As already discussed in Section 3.3, the choice for the TTS service to use is arbitrary and the attacker can choose any other TTS solution. We decided to use Google TTS because of the quantity

of available presets for WaveNet voices (i.e., 10), which was higher than other platforms at the time of the experiments. To generate our pool of samples to use, we built a script that saved the utterances generated via Google TTS in .wav format.

For what concerns the second payload type, we performed extensive tests with state-of-the-art tools for the generation of adversarial noise commands that work over-the-air: some adversarial noise commands were successfully self-issued, however, the success rate was rather low, hence an attacker cannot reliably use them for a real attack. Nonetheless, we report our findings as our results could serve as a baseline for future work.

Finally, regarding the third payload type, we argue that the adversary would rather not use their own voice for the attack, and recording other people while issuing commands (hence, performing replay attacks) can be very impractical. Hence, the third payload type is not ideal for AVA and we will not discuss success rates for it.

After generating the malicious audio files, the adversary will have at least one file for each command they would like to self-issue. This translates to the following:

$$\forall cmd.\ [Eve]genCmd(Alexa, cmd) \implies [[Eve]]cmd.wav.$$

4.4 Initial Foothold

After generating the weaponised audio files, we need to explore the potential methods of playing them on an Echo device. Hence, we looked at the available documentation online for Amazon Alexa, for the Alexa Skill Kit, and for Echo Dot, while also empirically experimenting with the device to find ways to play audio files. Our research revealed three approaches, or vectors, for playing audio on an Echo device:

- **Vector 1 - Radio Station**: the Echo device tunes into a radio station. This can be done by means of Music and Radio skills;

- **Vector 2 - Bluetooth Audio Streaming**: another device, e.g. a PC or a smartphone, connects to Echo via Bluetooth and streams audio on Echo's speaker;

- **Vector 3 - SSML** `audio` **Tag**: a skill that contains an `audio` Speech Synthesis Markup Language (SSML) [12] tag is opened, and the audio file specified by such SSML tag is played by Echo.

Nevertheless, not all vectors are compatible with AVA. In fact, we have discovered that to self-issue a command to Echo, the vector must meet a condition we refer to as the *non-exclusivity of the audio channel*. This condition states that if the Echo device is currently streaming an audio file and simultaneously receives a voice command, the audio streaming should continue uninterrupted. Therefore, we will now assess all three vectors to determine their suitability for AVA.

4.4.1 Vector 1: Radio Station

To make the Echo device tune in to a malicious radio station, the user must open a malicious skill first. This can be done in different ways: the attacker could trick the user into calling such malicious skill with a social engineering attack, or the same skill could be squatting another one, by means of homophones, compound words, or phonetic confusion [95]. An example of skill squatting is shown in Figure 4.2, in which the user wants to open the skill "World Time" but the voice personal assistant misinterprets the utterance and opens the malicious "Word Time" skill instead. Another way to trick the user into opening the malicious skill is giving it an invocation name that is already used by another skill [104], so that the malicious skill might be called instead of the legitimate one.

Once Echo is tuned into the radio station, it streams the voice commands that the adversary had previously generated. As soon as the Echo device captures the wake-word, the audio played by the radio station is turned down,

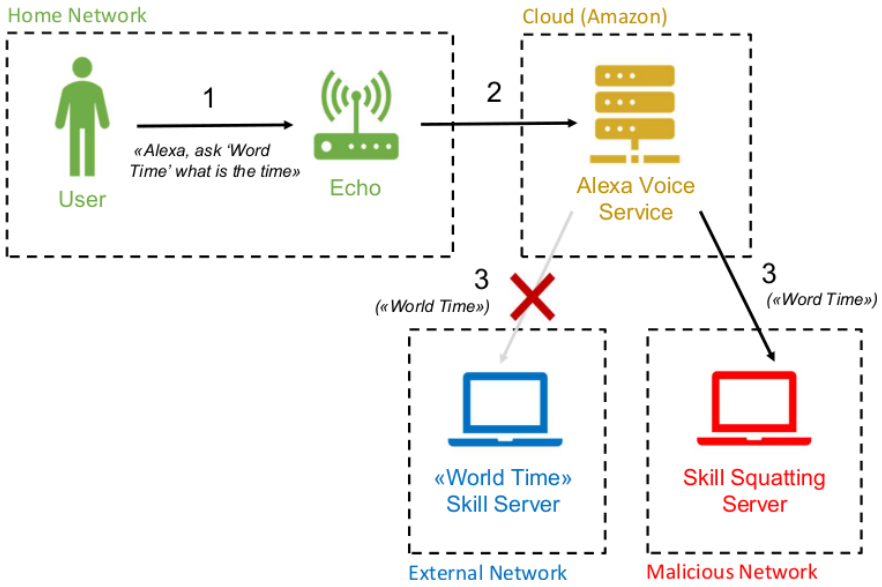

FIGURE 4.2: *An Example of Voice Squatting Attack*

but it is not interrupted: hence, this allows the device to hear the whole length of the self-issued voice command, satisfying the *non-exclusivity* condition and making this a valid attack vector for AVA. If the adversary establishes a connection with Echo using this vector we have that:

$$conn ::= remote.$$

$$[Eve]c2Server(EchoDot3, remote).$$

In fact, an advantage of this vector is that attackers can use the radio station as a C&C server to issue commands to multiple remote Echo devices at once. Radio stations can be deployed on the Alexa Skill Store by using the so-called Music and Radio skills.

Publishing a Malicious Skill

Anyone can deploy skills on the store, and skills do not need any special permission to run on the device or to play audio. Note that skills do not need to be installed, e.g. like apps on smartphones. There are documented cases of policy-violating skills successfully passing the validation [36]. Once a skill passes the certification process, further modifications to its code can be de-

ployed to the live skill without having to re-certify it again [146, 36], because the code resides on a server external to Alexa Voice Service (AVS). Hence, an attacker can certify a harmless skill and add malicious capabilities afterward. Assuming that the attacker publishes a skill called MyRadio, we have:

$$[Eve]deployApp(Alexa,\ MyRadio) \implies MyRadio \in Apps_{Alexa}.$$

4.4.2 Vector 2: Bluetooth Audio Streaming

If the adversary is in the proximity of the Echo device they want to attack, they can use another device, such as a smartphone or a computer, to connect via Bluetooth to the target Echo and make it act as a speaker for the adversary's device. This allows the attacker to play the audio files containing the malicious voice commands from the Echo's speaker. This operation does not require any PIN (hence, no brute-force or other similar attacks are required), and the whole pairing process requires approximately 25 seconds. Additionally, once paired, the Bluetooth device can connect and disconnect from Echo without any need to perform the pairing process again. Therefore, the actual attack may happen several days after the pairing (assuming the attacker uses the same paired Bluetooth device).

When the wake-word is captured while streaming audio from this vector, the volume of the streamed audio is turned down just like with Music and Radio skills — hence the *non-exclusivity* condition is met. More precisely:

$$conn ::= local.$$

$$[Eve]c2Server(EchoDot3, local).$$

Using this attack vector, the adversary does not need to host the voice commands online or to have a publicly reachable malicious radio station, because they can store the voice commands on the Bluetooth device. Another advantage of this approach is that the attacker can leverage the Full Volume Vulnerability described in Section 4.5 to increase the success rate of the self-issued commands. However, the adversary can only attack one Echo device at a time, and they need to be physically near the target for the Bluetooth

connection to work properly.

4.4.3 Vector 3: SSML `audio` Tag

As explained in Section 3.3, developers can use SSML within their skills to control how Alexa speaks. In particular, the `audio` SSML tag allows the skill developer to insert an .mp3 file, as they might want to use their own voice instead of Alexa's, or to play a short song at some point. However, if Echo hears a wake-word while reproducing an `audio` tag, such audio is paused instead of being turned down. This would interrupt the self-issue of any voice command: hence, the *non-exclusivity* condition is not met and `audio` tags cannot be used as attack vectors. It must be said that, at times, there can be a short delay between the wake-word being said and its recognition by the Echo device: this could allow a very short command, such as "Echo, hi" to be successfully self-issued with an `audio` tag. However, impactful commands are usually quite long and it is not always possible to shrink them to fit the very short self-issue window offered by the `audio` tag. Due to this, we consider only Radio skills and Bluetooth streaming as valid attack vectors.

4.4.4 Summary of Attack Vectors

Table 4.1 shows the valid attack vectors we found and their peculiarities. In particular, the Radio Station works remotely and can be used to control multiple devices at once, which allows the attacker to reach more targets than the Bluetooth vector. However, the adversary is not able to use the FVV and they need Social Engineering to make the user start the radio station. Additionally, if the user closes the radio station, the adversary would have to go over the whole process again to reconnect to the target Echo. By contrast, the Bluetooth vector works locally and with one device at a time, but it will not encounter all the other limitations. In particular, once the Bluetooth pairing between the target Echo and the adversary's device takes place, even if the latter is disconnected, they can always reconnect in a second moment without

repeating the pairing process.

TABLE 4.1: *Valid Attack Vectors*

Atk Vector	Remote	Multiple	FVV	Worldwide	SE Not Needed	Can Restart
Radio Station	✓	✓	✗	✗	✗	✗
Bluetooth	✗	✗	✓	✓	✓	✓

Remote: Works remotely — **Multiple:** Can control multiple Echo devices at once — **FVV:** Can be used with the Full Volume Vulnerability — **Worldwide:** Attack Vector is available anywhere in the world — **SE Not Needed:** Adversary does not need Social Engineering to start the attack — **Can Restart:** If the connection to the attack vector terminates, the adversary can reconnect without going through the initial steps.

4.5 Exploitation

After the attacker obtains the Initial Foothold, regardless of the used attack vector, they can finally proceed to self-issue voice commands to the attacked device. In fact, recall that, as a general rule:

$$\forall d, conn, cmd.wav.\ [Eve]c2Server(d, conn) \implies [Eve]giveCommand(p_d, cmd.wav).$$

Figure 4.3 shows the attack flow until this point: the adversary undergoes audio weaponization by generating the commands they want to self-issue and by storing them on the vectors they will use for the attack (steps 0.x), then they obtain the remote or local foothold (steps 1.x), after which a command can be self-issued (step 2) and correctly interpreted by AVS (step 3). If an external skill is requested by the command, AVS communicates with the related server (steps 4 and 5, indicated by a dotted line because this is an optional action), then it sends back the reply to Echo (step 6). As a result, the attacker can perform any action on the VPA (e.g. make phone calls, set alarms), on any skill (e.g., buy items), or they can control other smart appliances in the household (e.g., lights and door locks) (step 7).

Because the adversary can now self-issue any permissible command to

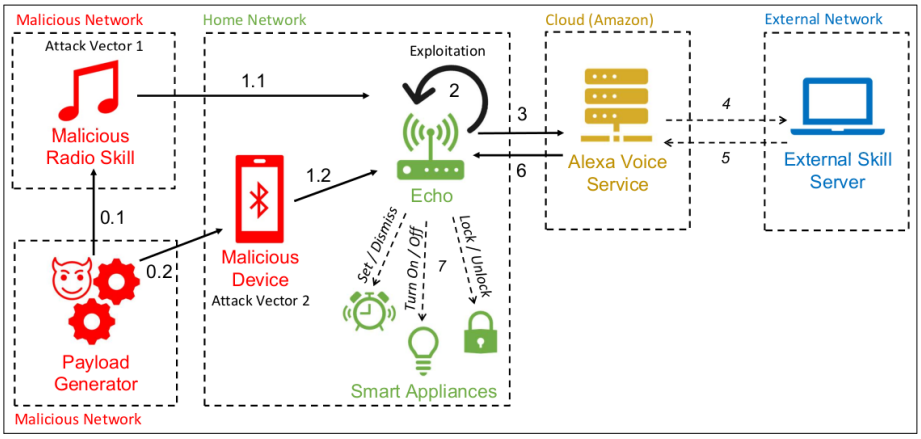

FIGURE 4.3: *AvA's Flow: Audio Weaponization to Exploitation*

the VPA, we get the following statement:

$$\forall cmd.wav.\ [Eve] giveCommand(Alexa, cmd.wav).$$

This is exactly the condition described in Section 3.4 for the attack's success. The self-issue vulnerability was confirmed by Amazon during the responsible disclosure process and we were subsequently able to confirm that it affects 4th Generation Echo devices as well. Other details on the vulnerability, including its Common Vulnerability Scoring System (CVSS) score, are presented in Table 4.2.

TABLE 4.2: *Details for the Self-Issue Vulnerability*

Vulnerability #1	**CVE-2022-25809**
Improper Neutralization of audio output from 3rd and 4th Generation Amazon Echo Dot devices allows arbitrary voice command execution on these devices via a malicious skill (in the case of remote attackers) or by pairing a malicious Bluetooth device (in the case of physically proximate attackers).	
CVSS Score: 9.8 (Critical)	CVSS 3.1 Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

We now describe the setup for our experiments and evaluate the results, to assess the practicality of the attack.

4.5.1 Setup of the Experiments

Figure 4.4 shows the placement of the 3rd Generation Echo Dot during our experiments for the Open, Wall, and Small scenarios. We streamed all payloads over Echo using two standard laptops running Windows 10 Pro 64-bit and Ubuntu 20.04, connecting them to Echo via Bluetooth (SBC codec).

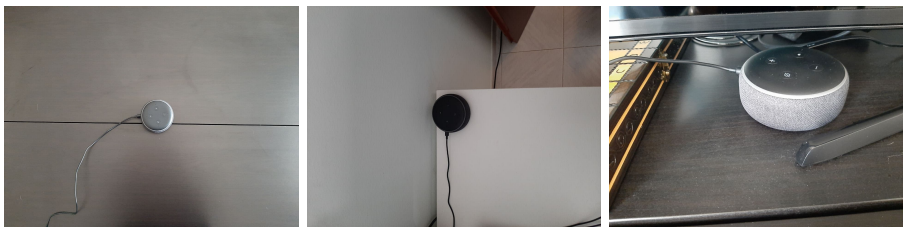

FIGURE 4.4: *Placement of the Echo Device in the Open (left), Wall (center), and Small (right) Scenarios.*

4.5.2 Evaluation

We evaluated AVA against a 3rd Generation Echo Dot. Because Music and Radio skills were available only in the US at the time of our tests, we focused on the Bluetooth attack vector to evaluate an attack scenario that is feasible anywhere. As stated before, we also verified that the self-issue vulnerability can be successfully exploited on 4th Generation Echo Dot devices as well, although the success rates would differ from the ones reported in this chapter, given this evaluation was performed on a 3rd Generation device. More generally, the attack is theoretically feasible against all devices (hence, all VPAs) that are vulnerable to the self-activation vulnerability, although the adversary would have to start the attack process all over again (i.e., from the Reconnaissance step) and gather information, craft malicious audio samples, and find a way to play an audio file on the specific voice-controllable device they want to attack. It also follows that the success rates shown in the upcoming tables may vary when performing the attack on a different device.

TTS Payload Performance

In the first test, we generated 70 audio payloads using Google TTS and measured their effectiveness when self-issued to Echo. The samples include 7 commands, each generated with 10 different Google TTS voice profiles, namely from "en-US-Wavenet-A" to "en-US-Wavenet-J". In all the tests, the volume of the Echo Dot was set to 5 (out of 10) and the language was set to "English (United States)". All commands were played in a room with 20dB background noise.

TABLE 4.3: *Self-Issued TTS Commands Reliability at Volume 5*

TTS Voice Command	Google TTS Voice Profile and Scenario								
	en-US-Wavenet-A			en-US-Wavenet-E			en-US-Wavenet-I		
	Open	Wall	Small	Open	Wall	Small	Open	Wall	Small
Wake-word	10	10	10	10	10	10	8	8	10
"Hello"	10	10	10	10	10	10	2	6	9
"What time is it?"	10	10	10	10	6	6	4	7	10
"Turn off the light"	4	8	9	6	8	10	2	6	10
"Open Mask Attack"	0	4	6	0	1	0	0	6	2
"Call mom"	2	8	8	0	4	6	1	6	8
"Call 1234567890"	0	0	0	0	0	0	0	0	0

Table 4.3 reports the results of the best-performing voice profiles. The table indicates the reliability of certain commands, which were selected taking into account the classification of the valid commands given in Section 4.2: (i) commands that use internal functionalities of Alexa or Echo, (ii) commands that control smart appliances, and (iii) commands that make use of third-party applications. More specifically, we tested four commands that are part of the first category, that is, "Hello", "What time is it", "Call mom", and "Call 1234567890". It can be noted that these commands have very diverse impact, as the first two have no impact on the user, the third could have some impact on the involved victims, while the last can severely undermine the user privacy; another command, "Turn off the light", is part of the second category, and it is there to demonstrate that smart appliances can be operated using self-issued commands; finally, another command, "Open Mask Attack", is part of the third category and proves the adversary can activate third-party applications with self-activations.

In the table, the reliability of selected commands is indicated with a score ranging from 0 to 10 according to the success rate of the command. For example, a command with a 23% success rate would score 3, while a command with a 97% success rate would score 10. Commands that never succeeded score 0. It can be observed that commands generated with the voice profile A perform better than the others in almost every scenario and that the "small" scenario is generally the one with higher command reliability, due to the reflection of the soundwaves caused by multiple nearby obstacles. We also find that only profiles "en-US-Wavenet-A" and "en-US-Wavenet-I" can reliably open Mask Attack,[2] while none of them can successfully dial a phone number.[3] We verified that this is due to the volume being turned down after the wake-word recognition, which does not enable Alexa to properly recognise longer commands. Nonetheless, because the wake-word recognition happens with a short delay, voice profiles that speak fast are able to pronounce more words before the volume turns down, hence they can issue longer commands more reliably than other profiles. As explained during the evaluation of the attack vectors, even in this case not all commands can be shortened enough to fit this window, such as *"Call 1234567890"*.

To reduce variance in the results of the tests with longer commands (i.e., "Turn off the light", "Open Mask Attack", "Call mom" and "Call 1234567890"), which seemed to yield more "random" results due to the inconsistency of the wake-word recognition described above, we performed 20 tests using those, and only 10 tests with the other commands, which had more consistent results. We report the results in the score form and not in percentage form for better visualisation of the results, given also that the precision loss is negligible for the results of longer commands (± 1 successful samples), and the exact number of succeeding samples is shown for all the other commands.

For each self-issue attempt, we distinguish between four cases:

[2] More details on this skill will be given in Section 4.6.
[3] Note that "1234567890" is a placeholder: a real phone number was used in the tests.

1. **The wake-word is not recognised.** The sample did not trigger Echo's wake-word recognition, so the command could not be self-issued.

2. **The wake-word was recognised but the command was not.** The sample could trigger Echo's wake-word recognition, but could not self-issue the command nonetheless.

3. **The wake-word was recognised and a command different than the self-issued one was executed.** The sample could self-issue a command, although Alexa made some errors in the transcription of the command, resulting in a slightly different command being executed.

4. **The wake-word was recognised and the self-issued command was executed.** The sample worked successfully, self-issuing the command.

In all of our results tables, a command was considered to be successfully self-issued only if it was correctly interpreted in all its length (i.e., case 4 in the list above).

Effectiveness Over Time

We observed that commands generated with certain voice profiles (e.g., "en-US-Wavenet-I") lose effectiveness over time: in fact, if they are used too many times in a short time span, the Echo device stops recognising them. We believe this is a defense against replay attacks. Because of this, during our tests, AVA slightly edited the voice pitch after each attempt to be able to re-issue commands. More precisely, instead of using pitch 0.00 for all the samples, its value varied from -2.00 to 2.00. In addition to this, we verified that the commands regain their effectiveness if the device has been moved or after a few minutes of inactivity: this last solution could also be adopted by an attacker as they would not need to issue a high-rate of commands. Additionally, the adversary can cycle through the commands they have generated, instead of re-using the same sample every time.

Effectiveness of Self-Triggering with Different Volume Levels

We tested TTS commands while no skill was running in the background, for different volume levels, and in different scenarios — results are shown in Table 4.4, where the score is computed on 10 tries for each sample. We observe that the success rate of the commands is not proportional to their volume (a higher volume does not mean a higher success rate). During the tests, we noted a degradation in the efficacy of the attack if the volume falls under 3. In these cases, we observed that the further volume reduction caused by the wake-word recognition, or by a skill running on the Echo device, rendered the audio completely inaudible, hence, it was not possible to reliably self-issue long commands anymore for most voice profiles.

Effectiveness in Presence of Another Audio Stream

We tested a scenario where the legitimate user tries to start another audio stream while the Echo device is connected to an attack vector, for example by saying "*Alexa, play Despacito on Spotify*", or by connecting a device via Bluetooth. Table 4.5 illustrates all possible situations: we observed there are three outcomes, namely, the attack vector is disconnected permanently (stop: ■), the attack vector is temporarily disconnected and is reconnected after the user has finished listening to their own music (pause: ▮▮), or the attack vector keeps the connection, and the user is not allowed to play their music (play: ▶). The favourable scenarios for AVA are those marked with the "play" and "pause" symbols, because the attack vector is not disconnected, while those marked with the "stop" symbol are not favourable as they disconnect Echo from the attack vector.

Full Volume Vulnerability

During our tests using the Bluetooth attack vector, we noted that sometimes the self-issued commands were played at full volume even after the wake-

TABLE 4.4: *Effectiveness of Self-Triggering ("Echo, what time is it?") When No Skill is Running in the Background*

Voice Profile		Volume Level and Scenario																	
Name	Gen.	6+			5			4			3			2*			1*		
		O	W	S	O	W	S	O	W	S	O	W	S	O	W	S	O	W	S
A		10	10	10	10	10	10	10	10	10	9	10	10	2	10	10	1	10	10
B		0	0	6	0	6	6	0	10	6	0	10	6	0	0	0	0	0	0
C		0	4	4	2	4	6	0	6	6	0	8	6	2	0	6	1	0	5
D		0	2	6	1	2	8	2	8	10	0	4	10	0	0	10	0	0	10
E		0	2	6	10	6	6	10	9	10	8	10	10	10	0	10	10	0	0
F		0	2	10	0	2	10	0	10	10	0	10	10	0	0	10	0	9	10
G		0	2	2	2	2	10	0	2	10	2	10	10	0	2	6	1	0	10
H		0	0	1	2	4	6	10	2	10	0	2	10	0	10	10	0	0	10
I		4	5	10	4	7	10	0	10	10	0	10	10	0	10	10	0	10	10
J		0	6	6	0	4	6	0	2	10	0	0	10	0	0	10	0	0	10

Name: The full name of the voice profile is "en-US-Wavenet-X", where X is the letter shown in the column. *When the volume is set at 1 or 2, the further volume turn down caused by Echo recognising the wake-word makes the played audio inaudible. Some commands succeed nonetheless due to the fact that the words "what time" manage to be played before the volume is muted, allowing the command to be interpreted correctly.

TABLE 4.5: *Behaviour of Echo When the Adversary is Already Streaming Commands and the User Asks Echo to Play Music*

Pre-conditions		User attempts to...	
Atk Vector	**VMA on?†**	***"Play Music"***	**Connect BT**
Radio	✗	■	▮▮
Radio	✓	▶	▮▮
Bluetooth	✗	▶*	■
Bluetooth	✓	▶	■

*In this case, both audio tracks are played at once. However, if the attacker stops their track and plays it again, they gain priority over the radio station, which gets muted. †For more information on our instance of VMA, see Section 4.6.

word recognition. Upon further inspection, we managed to reproduce this behaviour by self-issuing the command "Echo, turn off". In fact, subsequently, the affected Echo device did not turn down the volume anymore for the whole duration of the audio stream, allowing the attacker to self-issue commands at the current volume in their entirety. We call this the *Full Volume Vulnerability* (FVV).

We believe this is due to the fact that, when Echo is being used as a Bluetooth speaker and the "turn off" command is received, the audio stream should be stopped (as it happens with Music and Radio skills), but it does not. Hence, when another command is received, Echo does not turn down

the volume because it assumes the reproduction has already ended. Further details on this vulnerability are given in Table 4.6.

TABLE 4.6: *Details for the Full Volume Vulnerability*

Vulnerability #2	**No CVE**
Improper Resource Shutdown of Audio Output Channel from 3rd and 4th Generation Amazon Echo Dot devices disables the standard volume turn down of the audio currently in reproduction via Bluetooth after the wake-word recognition, enhancing the reliability of self-issued commands.	
CVSS Score: 6.5 (Medium)	CVSS 3.1 Vector: AV:A/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N

FVV has a great impact on the success rate of the self-issued commands: in Table 4.7 we re-evaluate the reliability of the commands that performed poorly in Table 4.3, this time issuing them after exploiting the FVV and doing 20 tests per sample. Comparing these results with those in Table 4.3, we observe that performance for some commands has been dramatically enhanced, for example, the number dial with profile A in the open and small space scenario. Figure 4.5 shows a direct comparison between success rates exhibited with and without the FVV, and we can see that performance of commands issued with FVV is always equal to or higher than the normal ones. Recall that the attacker can choose the samples to use during the attack, hence they can select the best ones and loop them during the AVA attack.

TABLE 4.7: *Enhancement of TTS Commands Exploiting FVV*

TTS Voice Command	**Google TTS Voice Profile and Scenario**								
	en-US-Wavenet-A			**en-US-Wavenet-E**			**en-US-Wavenet-I**		
	Open	**Wall**	**Small**	**Open**	**Wall**	**Small**	**Open**	**Wall**	**Small**
"Open Mask Attack"	8	4	10	0	6	4	6	6	10
"Call mom"	10	8	9	0	4	6	8	8	9
"Call 1234567890"	6	4	10	0	2	2	0	1	0

Adversarial Noise Payload Performance

We generated adversarial noise samples using Devil's Whisper DockerHub [34] and we tried to self-issue them using different parameters to measure the best-performing combination. In particular, we verified that the two key parameters to improve the success rate of the hidden commands are

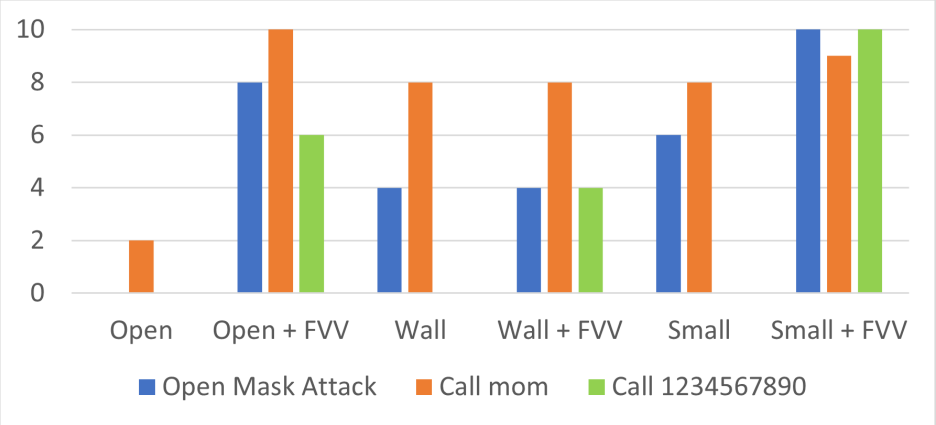

FIGURE 4.5: *Comparison Between Normal Self-Issue and Enhanced Self-Issue With FVV (en-US-Wavenet-A)*

`mini_noise_value` and `aspire_noise_value`, which alter the Signal-To-Noise ratio for the adversarial noise samples. We noted that, when using the default value of 5,000 for both variables, the Echo device could not be activated when the samples were self-issued, and only a small number (3%) succeeded in activating it when played in its close proximity. Increasing the noise value to 5,500, we observed a dramatic improvement in the activation rate when the samples were not self-issued: in fact, 83.5% of the samples were successful. We noted that the first self-activation of the Echo device happened at noise 7,500, and the first complete self-issue (both wake-word and command) of an adversarial noise input was at noise 8,000. We also verified that, by increasing the noise values past 11,500, the success rate of the samples decreased noticeably, due to clipping or excessive distortion of the audio.

In the tests, AVA was able to successfully self-activate the Echo device with adversarial noise samples in the small space scenario. The whole command could be recognised by exploiting the FVV or, alternatively, AVA exploited the adversarial noise samples to refresh the Mask Attack skill timer in a stealthy fashion. In fact, as we will explain in Section 4.6, only the wake-word needs to be recognised to achieve this goal. Table 4.8 reports the percentage of the adversarial `mini` samples that triggered at least one self-activation of the Echo

device or one self-execution of the command "Echo, turn off the light", over 100 commands per each noise value. It can be observed that the adversarial noise samples generated on top of the songs labeled *Song 1* and *Song 3* are the most effective in self-issuing complete commands. When these tracks are played again to check their reliability, we find that they replicate this behaviour 15% of the times on average.

TABLE 4.8: *Evaluation of Adversarial Noise Samples in the Small Space Scenario with Varying Noise and Background Songs*

Track	Noise Value and Behaviour							
	8000		9000		10000		11000	
	Ac	Ex	Ac	Ex	Ac	Ex	Ac	Ex
Song 1	3%	1%	16%	2%	15%	2%	13%	1%
Song 2	14%	0%	13%	0%	12%	0%	2%	0%
Song 3	0%	0%	0%	0%	3%	1%	8%	1%
Song 4	0%	0%	0%	0%	0%	0%	0%	0%
Song 5	1%	0%	1%	0%	0%	0%	0%	0%

Ac: % of `mini` samples that triggered self-activation of the Echo device
Ex: % of `mini` samples that were successfully self-executed

4.6 Persistence

We found another vulnerability on Amazon Echo, that is, the SSML `break` tag chain. When read by Alexa, SSML `break` tags add a pause of customisable length in her speech. For example, let's consider this string: `Hello <break time='5s'/> there`. When pronouncing this, Alexa will add a 5 seconds pause after the word "Hello". Amazon's policy regarding SSML `break` tags states that *"`break` tag silence cannot exceed 10 seconds, including scenarios with consecutive break tags. SSML with more than 10 seconds of silence isn't rendered to the user"* [12]. However, when chaining multiple SSML `break` tags of 10 seconds each, the total amount of silence is incorrectly calculated by the Amazon Skills Kit, and it is possible to exceed the 10 seconds limit. In fact, we found that the only limit for this chain of `break` tags is the maximum length allowed for the `outputSpeech` property of a skill response, set by Amazon to

8000 characters [13], that is, more than 400 `break` tags, which add up to over one hour of silence.

This is very interesting if we consider the standard execution flow of skills:[4] when they are started, Alexa reads some text and then it waits for a reply from the user to continue the interaction. The user has 8 seconds to reply, otherwise, the skill is closed. However, if the user (or a self-issued command) interrupts Alexa while she is speaking by saying the wake-word and issuing a command, the VPA will try to execute this command in the skill's context. This means that, if the initial text (and any other text) read by Alexa when the skill is launched contains those 400 `break` tags, the skill will keep running silently on the device for more than one hour, giving plenty of time to issue commands: any received command will refresh the timeout timer (i.e., it will make the skill output 400 more `break` tags) and will grant the skill owner, that is the attacker, the ability to capture every command given by the user and to decide the related replies. This is akin to Man-in-the-Middle attacks, and when affecting smart speakers they are known in the literature as Voice Masquerading Attacks (VMA) [177]. Details for the `break` tag chain vulnerability are given in Table 4.9.

TABLE 4.9: *Details for the Break Tag Chain Vulnerability*

Vulnerability #3	**No CVE**
Expected Behavior Violation in Amazon Alexa Skills Kit allows attackers to violate Alexa's SSML policy and create skills that remain silent for approximately one hour, by making the skill output a specially crafted sequence of SSML break tags.	
CVSS Score: 6.5 (Medium)	CVSS 3.1 Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:L

We implemented our VMA strategy described above on a skill that we called "Mask Attack", and we verified that it enables the adversary to:

- **Get a further layer of intrusion on the device,** because instead of being solely able to issue commands thanks to the self-issue vulnerability, the adversary can now listen to commands given by the legitimate

[4] Some special skills, like Music and Radio skills, do not follow this flow.

user and can tamper with the replies. This also allows the adversary to capture personal data that could be sent along with the voice commands, for example, the victim's home address. In fact, recall that $\forall p, cmd \in C.\ [Alice]giveCommand(p, cmd) \implies [[p]]cmd$, but in this case, Eve is listening to the commands instead of p. Hence, we get that:

$\forall cmd \in C.\ [Alice]giveCommand(Eve, cmd) \implies [[Eve]]cmd$.

- **Set up very realistic social engineering scenarios**, as the Mask Attack skill runs silently on the device, and the legitimate user will unknowingly interact with it, thinking they are talking with Alexa or with a certain skill. In fact, even commands that open other skills would be intercepted by Mast Attack instead. Additionally, as some sensitive actions require the user to say a PIN as a security measure, the adversary might attempt to steal it from the user by introducing a PIN request within a reply. Hence, this also means that a successful VMA attack can allow the adversary to know the secrets of the victim. In fact, recall that $\forall p, s.\ [Alice]shareSecret(p, s) \implies [[p]]s$, but because Eve is listening instead of p, we get that:

  $\forall s \in S.\ [Alice]shareSecret(Eve, s) \implies [[Eve]]s$

- **Get protection on the self-issue layer,** because any "Alexa, stop" command coming from the legitimate user would only terminate Mask Attack, but would not disconnect the Echo device from the attack vector, letting the attacker keep their foothold. In this way, the adversary gets Persistence for the self-issue.

Furthermore, it is not necessary that the user issues a command every hour to keep Mask Attack open: through AVA, the adversary can self-issue the *"Echo, go on"* command, which activates an Intent (that is, a sort of function within the skill) that just outputs another hour of `break` tags. We call this Intent the *ContinueIntent*. Additionally, we observed that the mere activation of the device with the wake-word is sufficient to refresh the skill timeout, as

the last executed Intent is triggered again. Hence, the Mask Attack skill can potentially keep running persistently on the device. The attacker can choose to close the skill by self-issuing the "Echo, quit" command: they might want to do so because any other command self-issued while the Mask Attack skill is open would be intercepted by the skill and not executed by the device. For example, if the adversary wanted to turn off all the lights within the victim's household, they cannot do it while Mask Attack is open, as the skill does not have access to the smart appliances.

In Figure 4.6 we summarise the main attack flow for AVA. In particular, after the exploitation phase, the adversary can switch from the *active state*, i.e. of self-issuing commands, to the *passive state*, in which Mask Attack is open. Table 4.10 gives some examples of actions that can be performed by the adversary in the active and passive states.

TABLE 4.10: *Examples of Malicious Actions to Perform in the Active and Passive States*

Active State *(Give Commands)*	**Passive State** *(Intercept Commands / VMA)*
Control smart appliances	Capture commands
Tamper with linked calendars	Ask for personal data
Reply/delete emails*	Ask for passwords/PINs
Call any phone number*	Tamper with replies
Buy items on Amazon	Infer user habits

*This feature of Alexa is available only in certain countries.

4.6.1 Retrieving Utterances and Realistic Replies

To pretend to be Alexa and remain unnoticed by the user, the Mask Attack skill performing the Voice Masquerading Attack (VMA) needs to retrieve a realistic response for any user command. To do so, we adopt a solution that is similar to the one adopted in Lyexa [120]. However, because AVA uses only the victim device, AVA does not need to timely inject the "use Mask Attack" command to every user query, because the user is already talking with that

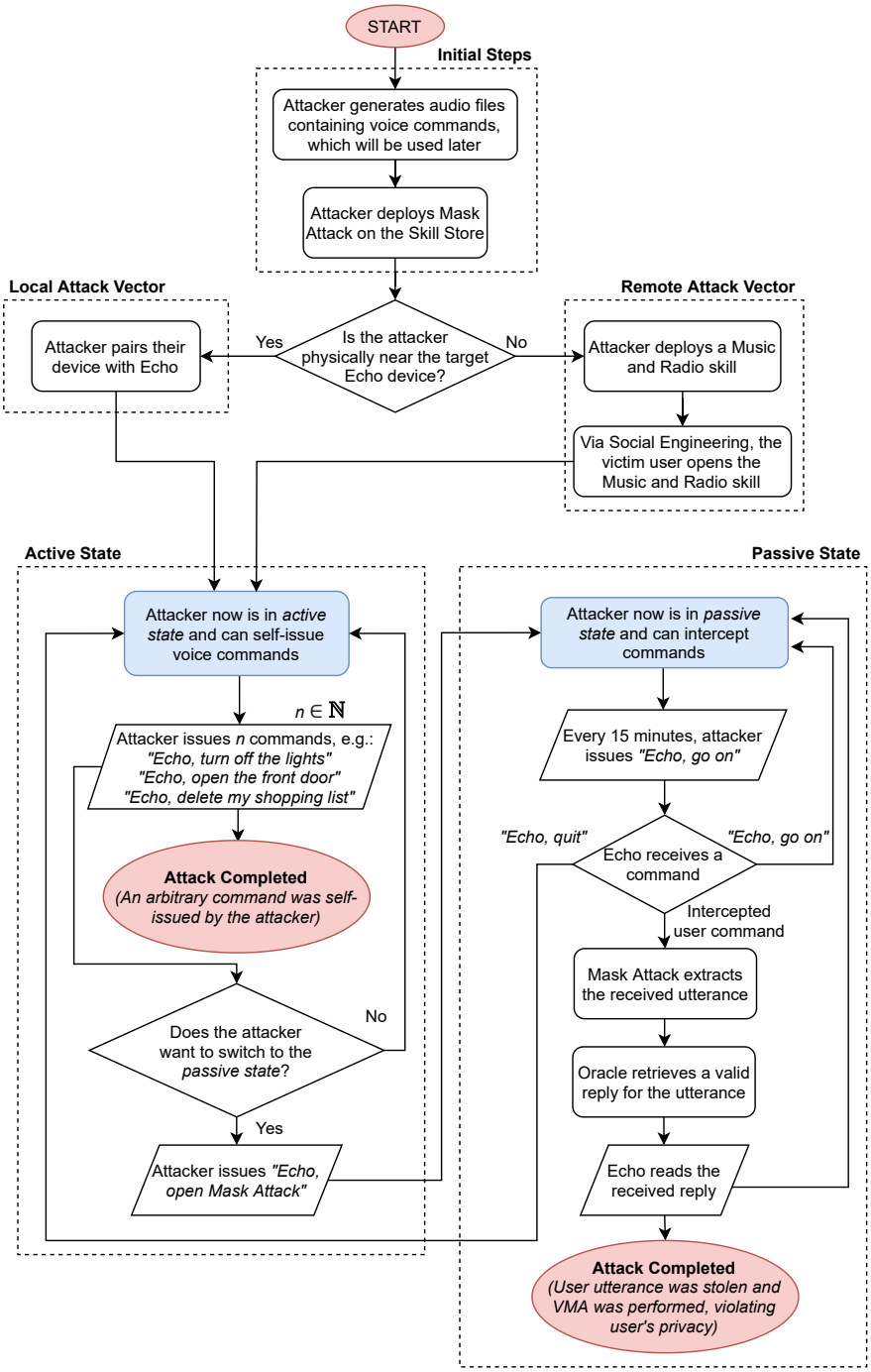

FIGURE 4.6: *Process Flow Diagram for AvA*

skill. Our VMA solution consists of two components that communicate via custom APIs: (i) the Mask Attack skill; and (ii) "The Oracle", an external script

that runs on another server controlled by the attacker. When the user, unknowingly, says a command to Mask Attack, the malicious skill has to retrieve the user utterance first, because AVS does not expose it directly. To this end, we train a custom Slot with some alphanumerical dummy values, so that any sentence can fit the Slot, and then we introduce a new custom Intent within the skill, whose only sample utterance is the Slot itself. We call this intent *InterceptIntent*. Since the only other Intent, *ContinueIntent* (described earlier in Section 4.6) only activates itself with the "Echo, go on" command, almost every other utterance[5] will activate *InterceptIntent*. This allows Mask Attack to retrieve the content of its Slot, that is, the utterance. Note that *InterceptIntent* activates even when the user tries to call another skill (e.g. "Echo, ask Big Sky what's the weather like in Paris").

The Mask Attack skill then needs to retrieve a realistic response for the user query. To this end, the Mask Attack skill sends the user utterance to the Oracle, which leverages the AVS Client package [10] to asynchronously query AVS and obtain the real reply. To communicate with AVS, the Oracle transforms the plaintext utterance into an audio file using Google TTS.[6] The Oracle then receives the reply to the query from AVS, within one or more .mp3 audio files. The Oracle transforms the reply in text using Google Speech-To-Text (STT) and sends it back to Mask Attack. Mask Attack now has a realistic reply for the user's query and reads it to the user. The whole process takes approximately 5 seconds.

To give a more natural user experience to the victim user, with the ultimate goal of avoiding detection, the adversary could also hardcode popular functions of Alexa (or hardcode standard replies) within the Mask Attack skill's source code. For example, they could insert a function for answering a common question such as "Alexa, what is the time?", and another one for "Add [item] to my shopping list". In this way, the Oracle is not contacted to get a

[5] Some utterances are reserved and they always trigger standard actions. For example, "close", "stop", "exit", and "quit" always close the active skill.

[6] Any other online Text-to-Speech service would work, as long as it can output `LINEAR16` [64] encoded audio files, which is the format accepted by AVS.

realistic reply (as the adversary already knows what are the possible replies and can hardcode them), and the user instantly gets their response, just as they were interacting with the real Alexa.

Implementation of Mask Attack

Mask Attack is written in Node.js and is hosted on Amazon Lambda. It uses the `ask-sdk-core` and `axios` packages. We also used a CustomUserAgent called `cookbookprogressive-responsev1`.

Implementation of the Oracle

The Oracle is written in Python 3 and leverages the Google Cloud `speech` and `texttospeech` packages, along with `AlexaClient`. The Oracle also connects to a set of APIs we coded in PHP to communicate with a MySQL database, where we store received user utterances and their replies.

4.7 Responsible Disclosure

We dutifully contacted Amazon via their Vulnerability Research Program, reporting the existence of all three vulnerabilities we have found, that is, the self-issue, the Full Volume, and the SSML `break` tag chain vulnerabilities. Our report was submitted on January 21st, 2021, and the Amazon VRP Team quickly reacted to it, asking for more details. On February 18th, 2021, our research team engaged in a call with the Amazon VRP Team to discuss details of the attack, its impact, and possible mitigation strategies. Our report was rated with Medium severity by the Amazon VRP team. Between December 2021 and January 2022, Amazon deployed some fixes for the remote self-issue vulnerability via Music & Radio skills and for the break tag chain, which are no longer possible in the manner demonstrated by our research. Disclosure of the vulnerabilities happened on February 17th, 2022, after the deadline that was agreed upon with Amazon. The self-issue vulnerability got a CVE entry

(CVE-2022-25809) and was rated with a 9.8 CVSS score by the NIST [57].

4.8 Related Work

First, we discuss self-issue attacks on other platforms, then we compare AVA with another known instance of Voice Masquerading attack and highlight the main differences between all these attacks.

4.8.1 Self-Issue Attacks

Jang et al. [82] leverage accessibility (a11y) tools made available on most OSes to identify two attacks that make use of voice commands self-issue. The first one works on Windows systems and exploits Windows Speech Recognition, which can be started by any process. By self-issuing some specific commands, any malware can escalate from low privileges to administrator and can control the system via the voice channel. The second attack works on Android smartphones, and allows an attacker to bypass the Voice Authentication mechanism by recording and then replaying the authentication phrase, which is usually "OK Google". Diao et al. [48] develop malware (VoicEmployer) that runs on Android and is able to self-issue voice commands using Google Voice Search when the user is not listening. Their attack (GVS-Attack) does not require any permission. Alepis and Patsakis [6] use Google Firebase as a C&C server and leverage sensors that require no permissions to infer if a smartphone is left unattended. If it is the case, the C&C is contacted and the device issues TTS commands to itself and to nearby devices, if possible. Differently from these works, AVA is the first attack that uses self-activation against smart speakers to gain prolonged control over a dedicated VPA device, and the first work to evaluate self-issued adversarial noise commands (see Section 4.5.2).

4.8.2 Voice Masquerading Attacks

The first work to successfully implement a working VMA was Lyexa [120]. Although the implementation of the VMA in Lyexa is similar to the one in AvA, there are some key differences: (i) AvA does not make use of rogue speakers; (ii) AvA can use different attack vectors; (iii) AvA achieves persistence of the malicious skill on the device in a different way; (iv) Lyexa uses ultrasonic, inaudible voice commands; (v) AvA does not need to dynamically append "use Mask Attack" to user commands because the Mask Attack skill is started beforehand by the attacker. Table 4.11 summarises the comparison among all discussed attacks and AvA.

TABLE 4.11: *Comparison Between* AvA *and Other Attacks with Similar Strategy or Goal*

Attack Details			Does not rely on...			Features			
Paper	**Target**	**Leverages**	**Soc. Eng.**	**2nd Speaker**	**Proximity**	**C&C**	**HVC**	**Deceive**	**Persist**
Adv. Music [106]	📢	-	✓	✗	✓	-	-	-	-
Monkey [6]	📱	SI	✗	✓	✓	✓	✗	-	✓
YVAIM [48]	📱	SI	✗	✓	✓	✗	✗	-	✓
A11y [82]	📱, 💻	SI	✗	✓	✓	✗	✗	-	✓
Lyexa [120]	📢	VMA	✓	✗	✗	✗	✓	✓	✓
AvA (Local)	📢	SI, VMA	✓	✓	✗	✗	✓	✓	✓
AvA (Radio)	📢	SI, SS, VMA	✗	✓	✓	✓	✓	✓	✓

📱: Android mobile phones. — 💻: Windows computers. — 📢: Smart speakers. — **SI:** Self-issue. — **SS:** Skill Squatting. — **Soc. Eng.:** Social Engineering, in this case, ✓ is assigned if the user does not have to inadvertently open malware (in form of applications, executables, or skills), ✗ otherwise. — **Proximity:** ✓ is assigned if the attack does not require the adversary (or a device controlled by them) to be near the target, ✗ otherwise. — **C&C:** ability, for the attacker, to issue arbitrary commands to multiple compromised devices at once, ✗ otherwise. — **HVC:** the ability to use adversarial noise inputs or inaudible ones as payload during the attack. — **Deceive:** the ability to interact with the user without being detected. We mark smartphone and computer attacks with "-" as they rely on malware that could hide the malicious behaviour. — **Persist:** the attack keeps running unless uncommon circumstances occur.

4.9 Summary

In this chapter, we introduced AvA, an attack that plays audio tracks containing voice commands over an Echo device, which executes them as if issued by the legitimate user. We evaluated the performance of many self-issued commands, generated with different Google TTS voice profiles, and we pre-

sented the Full Volume Vulnerability (FVV) to enhance the reliability of self-issued commands. We observed that, when exploiting the FVV, an attacker can self-issue any command to an Echo device with a 99% success rate in the small scenario, that is, when there are multiple obstacles close to Echo. In all other scenarios, we observed an above-50% success rate on average. We reported the vulnerabilities we found to Amazon, who rated them with Medium severity. AvA does not require rogue speakers to be in proximity of the target device, a constraint that many other attacks share, and does not have heavy computational requirements, as TTS samples can be easily generated and stored for later use. Additionally, the exploitation flow is rather simple, as it is sufficient to connect Echo to one of the attack vectors. During the tests, we were able to successfully perform a set of malicious actions that an attacker could issue by leveraging AvA, showing it has an impact on the physical safety and on privacy of the victim user. In fact, the self-issue vulnerability we discovered was later rated with Critical severity by the NIST.

Chapter 5

An Assessment of AvA in the Real World

In this chapter, we assess the feasibility, impact, and limitations of AvA in real scenarios. We do so by analysing the results of a field study on three different voluntary households (Section 5.2), and of a survey submitted to a study group of 18 Amazon Echo users (Section 5.4.1). Additionally, we detail a set of attacks that can be performed on the victim's Echo device via AvA, and we show their impact on the user's physical safety and privacy (Section 5.3).

Contents of this chapter have already been partially discussed in the following paper:

- Esposito S., Sgandurra D., Bella G. Alexa versus Alexa: Controlling Smart Speakers by Self-Issuing Voice Commands. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '22)*. 2022.

5.1 Introduction

In the last chapter, we evaluated the AvA attack and showed that the adversary can perform actions that can be very detrimental to the user's security, safety, and privacy. However, another important factor to assess is whether

the attack is feasible in the real world, or if it is only a theoretical attack that is unlikely to take place in real environments. In particular, as the VMA involves a malicious application interacting with victim users in an attempt to deceive them and make them believe they are talking with the VPA, we want to analyse how real people interact with such a malicious skill and if it is possible that they actually get deceived by it.

Hence, we first performed a field study with the participation of three voluntary households to assess the overall feasibility of AvA in real environments. In this field study, the users had to interact with a device under the AvA attack, both with and without an ongoing VMA. There were two users per household: in the first one, a user was in the 46-50 age gap, while the other was in the 71-75 one; in the second household, a user was in the 26-30 gap, while the other was in the 31-35 one; finally, in the third household a user was in the 18-20 gap, while the other was in the 26-30 one. All the users were from Italy.

We also performed an impact assessment of AvA, by actually performing malicious operations on a real device by self-issuing voice commands to it. Furthermore, we performed a limitation assessment using the results of a survey that was administered to 18 Amazon Echo users. In this survey, they answered questions that enabled us to understand what are the typical conditions in which an adversary would operate and to compare them with the optimal conditions for the AvA attack. Data collected from the aforementioned field study was useful for this purpose as well. Of the people that answered the survey, 2 were in the 18-20 age gap, 3 were in the 21-25 one, 9 were in the 26-30 one, 1 in the 31-35 one, 2 in the 46-50 one and 1 in the 71-75 one. Table 5.1 summarises the age demographics for both the field study and the survey. Of these users, 14 were from Italy, 2 from the UK, 1 from Norway and 1 from the Netherlands.

With these three assessments, we show that the AvA attack is feasible in all its declinations (i.e., the *active* and *passive* states) and that its limitations are mostly theoretical. At the end of this chapter we also briefly discuss the

mediatic impact that AvA had worldwide.

TABLE 5.1: *Age Demographics for the User Study and the Survey*

Age Gap	Field Study Users	Survey Users
18-20	1	2
21-25	0	3
26-30	2	9
31-35	1	1
46-50	1	2
71-75	1	1

5.2 Feasibility Assessment

To assess the feasibility of AVA in real scenarios, we performed a field study with three different households. All participants in the three households possessed an Amazon Echo device at various levels of expertise. For practical and ethical reasons, during these experiments, we asked the participants to use our own device, and we placed this device in the same position and with the same orientation as the participants' device. All the users in the three households were informed about the replacement of the device and they were told we would perform some usability tests by observing their interactions with our device. Before the tests, we did not specify to the users the exact goals of the tests to not introduce bias. For the ethics of this study, see Section 5.6.

Stealth Initial Foothold via Bluetooth The goal of this test is to check whether is it possible, for an attacker who temporarily has access to the household, to connect to Echo via Bluetooth without being detected, obtaining an Initial Foothold. In a real scenario, this could be a professional who has to repair something in the house or someone who was invited over by the legitimate user. In all three households, we were able to connect to Echo without anyone realising it: this can be trivially done if no one is in the room for a very short time frame. We set the volume of the Echo device to 1 (*"Echo,*

volume 1"), then we turned on the Bluetooth (*"Echo, turn on Bluetooth"*) and connected to it. We interrupted all Alexa's status messages by pressing the action button repeatedly. The whole process lasted 25 seconds on average.

Command Self-Issue Perception The goal of this test is to understand if people in the household are always able to hear the TTS commands, or if there are some conditions, such as their position in the house and the device's volume, that allow the attacker to self-issue commands even when the user is at home and awake. We set the device to the volume used by the different households (respectively 3, 5, and 6) and we told the users to go to different parts of the house. We then self-issued a command to the Echo device, let Echo play the normal response, and then asked the users if they did hear something. Regardless of the volume, all users in the same room and in adjacent rooms did hear both the command and the reply correctly. Only in the household with volume 3, in an adjacent room with the door closed, the user said they thought it was someone talking outside of the window, and they could not infer what was said. Finally, no user in any household was able to hear either the command or the reply, if they were in a non-adjacent room, although a user in the household with volume 6 reported they thought Alexa had said something.

User Behaviour After Bluetooth Connection When the adversary reconnects to an Echo device they have already paired their device with, they will skip the pairing process. However, the connection of the device makes Echo output a "successful connection" audio message that the user can hear — hence, the goal of this test is to assess what would the users do when hearing the message. We connected our device to Echo surreptitiously when the users were near the device. When asked separately if they could tell what happened, half of them could infer that we had connected a device to Echo via Bluetooth. However, they did not perceive it as a malicious action. Only one user tried to understand what happened by asking a series of questions to

Alexa, for example “what happened?” and “are you connected to something?”, however, the device did not reveal any ongoing Bluetooth connection.

Command Self-Issue from Outside The goal of this test is to check whether it is possible to issue commands from outside the houses after a device has been paired with Echo. We disconnected our paired device from Echo and we tried to reconnect once outside. We successfully managed to reconnect our device in all three cases, also succeeding in self-issuing voice commands. In two cases, the distance from the device was 8m, with two walls between us and the device. In another household, the distance was 3m, with only one wall separating us from the device.

Interaction with the Mask Attack Skill The goal of this test is to assess the behaviour of people interacting with the Mask Attack skill. In particular, we wanted to understand whether they were able to infer that an attack was in place. We told the users we wanted to observe their interactions with the Echo device, and that they had total control over it — they could change the volume, mute the microphone, issue any command, etc. They were then presented with the device with the Mask Attack skill already opened, unknowingly to them. We did not use the “Echo, go on” command to refresh the timeout timer during this test. All users noticed that Echo was slower in replying to the commands. When issuing commands with very long replies, the Oracle could not reply within the allocated five seconds, hence Mask Attack sent back the reply to the last issued command. However, when they issued the command again, they could hear the correct reply since the Oracle had the time to retrieve the answer, convert it to text, and store it in the database. All of them reported that they attributed this behaviour to a bug. One of the users noted the blue light blinking upon receiving an incorrect reply, due to an erroneous transcription of a date by the Speech-To-Text service. The user then restarted the device, reporting that they often get weird replies from Alexa (a feeling shared by the other participants as well), however, they thought it was

safer to restart the device because of the blinking light. All the other users participating in the study did not notice the blinking light or perceived it as normal. Additionally, all the other users kept interacting with the device until we stopped the experiment.

5.2.1 Feasibility Assessment Results

Table 5.2 summarises the results of the field study. As we can see, it is possible to quickly and stealthily set up a Bluetooth connection with Echo, and we also verified that it was possible to issue commands from outside the household in all cases. This confirms the dangerousness of the Bluetooth attack vector. As expected, the study confirms the limitation of the audible payloads, hence, the adversary needs to carefully plan when to self-issue commands to avoid alerting the users. Nonetheless, all other tests show that AVA is feasible in real scenarios, as it can successfully start and keep running, and most of the users did not perceive anomalies as malicious.

TABLE 5.2: *Field Study Results*

Test	**Observation Item**	**Result**
Stealth BT	Time Needed	25s on average
	Users Aware	0% of users
SI Perception	Same Room	100% of users could hear
	3.5m and Wall	100% of users could hear
	3.5m, Wall and Door	66% of users could hear
	Non-Adj. Room (7m+)	0% of users could hear
BT Connect	Perceived as Malicious	0% of users
	Inferred What Happened	50% of users
	Took Some Actions	16% of users
Outdoor Self-Issue	Successful Pairing	100% of households
	Successful SI	100% of households
Mask Attack	Noticed Slow Behaviour	100% of users
	Noticed Wrong Replies	100% of users
	Noticed Blinking Light Ring	16% of users
	Perceived as Malicious	16% of users
	Switched Off the Device	16% of users
	Intercepted Commands	100% (41/41)

5.3 Impact Assessment

We now detail the types of malicious actions that AVA allows an attacker to perform, and we assess their success rate (RG2).

Control Other Smart Appliances AVA allows an attacker to control other smart appliances that are connected to Echo. This action can undermine the physical safety of the user, for example, when turning off the lights during the evening or at night time, turning on a smart microwave oven, setting the heating at a very high temperature, or even unlocking the smart lock for the front door. We were able to turn off the lights in our house 93% of the times using the FVV. In these scenarios, sometimes Echo might repeat the name of the device the attacker wants to turn off as a form of double-check. In these cases, the adversary only has to always append a "yes" approximately six seconds after the request to be sure that the command will be successful.

Call Any Phone Number of Attacker's Choice The attacker can make the victim Echo device call a phone number controlled by them, effectively eavesdropping on what is being said in the proximity of the Echo device. While it is true that the user could see that the light on top of the device turns green (or on the bottom of the device in 4th Generation devices), it may take a while for the user to notice it, depending on the position of the device. Additionally, inexperienced users might not know what the green light means, hence ignoring it. In fact, among users who answered the survey discussed in the next section, only 27% knew that the green light is related to ongoing calls. During the test, we exploited AVA to call two phone numbers owned by us, by placing the device in different locations, and we succeeded 73% of the times when using the FVV. In the remaining 27%, some numbers were incorrectly interpreted by the Echo device.

Buy Items With the Victim's Amazon Account Although the user would be notified of the purchase via email, and be able to cancel the malicious order, this would be detrimental to the victim's user experience on Amazon, and they could lose trust in the company. The attacker can also delete items that the user had previously put in the shopping cart. We were able to perform both actions using the command self-issue, and we achieved a 100% success rate when using the FVV. Echo did not correctly understand what product we wanted to buy once (20%), however, another article was placed in the shopping cart nonetheless, and we were able to buy it.

Tamper with the User's Linked Calendar If the user had previously linked their online calendar, the attacker can exploit AVA to add, move or delete events from the user's calendar. We were able to perform all these actions, via self-issue, on a Google Calendar linked to Alexa. With FVV, we succeeded 88% of the times.

Impersonate Other Skills and the VPA The attacker can exploit AVA to start any skill of their choice, including Mask Attack to perform our instance of the VMA. The impact of VMA attacks is already thoroughly discussed in other works [177, 120], and they can seriously undermine the victim user's privacy. In fact, they could lead the user to disclose their passwords or security PINs, personal data (e.g., home address, name, and surname), or even sensitive data, such as health status, religious belief, and sexual preferences. During our tests with Echo users, no one noticed that the Mask Attack skill was running, although all users could understand that Echo was experiencing some problems due to the delay in the replies. In the tests, only one user noticed the blinking light and thought it was safer to shut down the Echo device. All the other users kept interacting normally with Echo.

Retrieve User Utterances As Mask Attack stores all intercepted commands in a database, this would allow the adversary to extract private information,

gather information on used skills, and infer user habits, e.g., to calculate when it is safer to self-issue commands. During the field study, we were able to retrieve all utterances made by the users — as explained in Section 5.2.

5.3.1 Impact Assessment Results

Table 5.3 summarises our findings: green indicates an optimal success rate (we set the threshold at ≥80%) and yellow indicates a medium success rate. Note that, as explained in Section 5.6, during all the tests we have only used accounts belonging to us.

TABLE 5.3: *Impact Assessment Results*

Threat	Success Rate		Confirm?†	Notes
Control Smart Dev.	14/15	*(93.3%)	Sometimes	Tested with lights only
Call Phone Number	11/15	*(73.3%)	Yes	14 digits including prefix
Buy Items	5/5	*(100%)	Yes	80% (4/5) items recognised
Tamper Calendar	8/9	*(88.8%)	Sometimes	Did not ask when moving event
VMA	-	-	-	See Table 5.2
Retrieve Utterances	41/41	(100%)	-	See Table 5.2

***** Indicates that the success rate was achieved using the FVV. — † indicates if the command requires the attacker to self-issue a "Yes" after a few seconds to answer the confirmation request.

With these tests, we demonstrated that AVA is very impactful as it can be used to give arbitrary commands of any type and length, with optimal results — in particular, an attacker can control smart lights with a 93% success rate, successfully buy unwanted items on Amazon 100% of the times, and tamper with a linked calendar with 88% success rate. Complex commands that have to be recognised correctly in their entirety to succeed, such as calling a phone number, have an almost optimal success rate, in this case, 73%. Additionally, results shown in Table 5.2 demonstrate the attacker can successfully set up a Voice Masquerading Attack, via our Mask Attack skill, without being detected and all issued utterances can be retrieved and stored in the attacker's database, namely 41 in our case.

5.4 Limitations Assessment

In this section, we first discuss the limitations of AVA, and then we report the results of a survey submitted to a study group of 18 Amazon Echo users, to evaluate the limitations' effectiveness (Section 5.4.1). In fact, while AVA can issue arbitrary commands for a prolonged time on Echo, some events can terminate the attack. We list and compare them with the results of the already discussed field study (Section 5.2) and of a user survey (Section 5.4.1), showing that the likelihood for them to take place is minimal (RG1).

The Echo Device is Unplugged from Power Radio stations, and in general played audio tracks, are not automatically played again when the Echo device is restarted, and this would disconnect Echo from the attack vector. However, only 27% of the users of the survey ever restarted their Echo device, and only 6% do it systematically.

The User Says "Alexa, Stop" The command "Alexa, stop" would close the radio station, disconnecting Echo from the attack vector. However, if the Mask Attack skill is running, such a command would close the skill instead, so the user would have to say the command two times in a row to terminate the radio station and the AVA attack. This does not apply when using the Bluetooth attack vector, as this command does not stop the stream. Realistically, the user does not say this command without a specific reason: during our experiments, no users tried to issue this command.

Headphones Connected to Echo AVA would not succeed if headphones are connected to Echo, because the malicious commands would be played through them instead. In our study group, no user tried to connect them.

Echo's Microphone is Turned Off Echo Dot includes a button to turn off its microphone. If pressed, payloads cannot exploit the self-issue vulnerability

because the microphone would not capture anything. Nearly 89% of the users never turn off Echo's microphone, or they do it rarely. The remaining 11% claimed to do it sometimes.

Payloads are Audible Although adversarial noise commands can be used with AVA, they are not very reliable when self-issued (recall the evaluation in Section 4.5.2). To deploy a real attack, the adversary has to use TTS commands, which can be heard by nearby users up to 4.5 metres on average, as shown in Section 5.2. Because the attacker does not know if the legitimate user is nearby, they might want to issue arbitrary commands during the night, or after they gained insight into user habits with the aid of Mask Attack.

Volume Buttons While the Mask Attack skill is on, the only way for the user to change the volume is by using the physical buttons on the Echo device, because the skill is intercepting voice commands: this could make the user suspicious. However, this would not affect 27% of the users in our study group, who claimed to use manual commands to change the Echo's volume.

Echo's Light Ring Turns Green During a Call When the attacker makes Echo call a phone number they control to eavesdrop on the user, the user could see the light ring become green. Depending on the position of the device, this could take some time. Additionally, the survey reports that only 27% of the users know what the green light means.

Echo's Light Ring Blinks when Reading While Alexa is talking, Echo's light ring slowly blinks. When the Mask Attack skill is open, the light ring keeps on blinking as Alexa is reading the `break` tags issued by the skill. Similarly to what has been said for the green light, depending on the position of the device, the users might not notice it. In fact, during our field study, only one user noticed the blinking light.

Inaccuracy of Reply Retrieval Some skills use pre-recorded audio tracks, such as the voice of a journalist. Mask Attack will read these tracks with Alexa's voice. Additionally, *by default*, skills cannot access specific data on the user account, e.g. shopping lists, unless the user grants them the appropriate permissions via the Alexa companion app. For such commands, Mask Attack will send a feasible reply anyway, however, it will not be the *correct* reply.

Delay Before the Reply The infrastructure behind Mask Attack needs some time to convert the user utterance into audio and, vice-versa, the reply into text. All users noted this delay, however, none of them perceived it as malicious. Recall that the adversary can hardcode replies to the most common queries made to Alexa so that the delay is added only when the correct response is not known in advance and the Oracle must be contacted.

5.4.1 Evaluation of the Limitations

We wanted to evaluate how realistically the limitations discussed in Section 5.4 would reduce the effectiveness of AVA. To this end, we created a detailed survey composed of twelve questions, clustered into three categories: Usage Information, Scenario Recognition, and Limitations Assessment. Firstly, Usage Information is related to how the users interact with their devices. This information was used to determine the overall confidence of the user with the device and to understand the quality of their answers. Secondly, Scenario Recognition questions are needed to collect information on the different placements and volumes of real Echo devices, to understand the distribution of the different scenarios. Finally, Limitations Assessment questions allow us to estimate the likelihood of the limitations actually taking place in real scenarios. To avoid bias, questions were presented to the users with no categorisation and with uniform IDs (Q1, Q2, Q3...). Table 5.4 lists all questions answered by the study group.

As already stated before, we submitted the survey to a study group of 18

TABLE 5.4: *Survey for the Study Group*

ID	Question
Usage Information	
U1	From 1 to 10, how would you score your skill in interacting with Amazon Echo and Alexa?
U2	How often do you use your Echo device (e.g., once a day)?
U3	What do you usually use your Echo device for? If you want, you can answer with some of the most common commands you use.
Scenario Recognition	
S1	Can you send a photo that shows the placement of your Echo device? Please make sure that the photo does not show people or personal data. If this is not possible, can you estimate the distance (in cm) of your Echo device from the nearby walls and obstacles?
S2	What is the average volume of your Echo device? If you do not know, the current volume will be fine (ask "Alexa, what is the current volume?").
Limitations Assessment	
L1	Have you ever unplugged your Echo device from the power source? Why?
L2	Have you ever connected headphones to your Echo device? How often do you do so?
L3	How do you usually change Alexa's volume?
L4	How often do you turn off Echo's microphone?
L5	Do you know what the green light ring around Echo means? If you do not know and want to learn what it means, please search on the Internet AFTER answering this question.
L6	Have you ever used Echo as a Bluetooth speaker (i.e. connecting another device via Bluetooth to play music or other audio files)?
L7	Do you follow any security best practices (e.g., you turn off Echo's microphone after 9pm)?

Echo owners, with different levels of ability in using their Echo devices. Some users owned more than one device, and they were asked to answer the survey choosing one of their devices randomly. Table 5.5 summarises the results of the survey, where we have used the green colour to indicate results that are favourable for AVA, red for unfavourable scenarios, and orange for

mostly neutral results. As we can see from the table, the average volume of the devices is set at 4.7, which is ideal for AVA's success rate, given that AVA commands at volumes 4 and 5 perform well as shown in Section 4.5.2. Additionally, most of the users place their Echo device in the Small Space scenario, which allows AVA to achieve the best performance.

Only 27% of users ever rebooted their device voluntarily, mostly to move the device to another room, or for cleaning purposes. Only 11% of the users ever switched off their Echo due to a malfunctioning or to perform a hard reboot, and only 6% turned off the device because they did not recognise the color of the light ring. This is favourable for AVA because restarting the device is a way to disconnect Echo from an attack vector, taking away the Initial Foothold from the attacker. No one ever connected headphones to Echo — their connection would not allow the self-issued commands to be captured by Echo's microphone. However, because 0% of users ever used them, we can assume this behaviour is very unusual among Echo users, and this limitation remains theoretical.

Only 27% of participants knew that the green light indicates an ongoing call, hence the remaining 73% would not realise that there is an ongoing rogue call. Only 11% of users claimed to mute Echo's microphone sometimes — all the other participants reported they do it rarely, very rarely, or never. Hence, the Echo device will keep listening to self-issued commands in most cases. Finally, only 6% of users claimed to systematically follow security procedures, such as turning off Echo's microphone during the night — this means that most devices would be vulnerable during nighttime or when the user is away. Hence, the results of this survey clearly show that most of the limitations do not impact the feasibility and success rate of AVA.

5.4.2 Limitations Assessment Results

Table 5.6 summarises the evaluation of the limitations, based on the results of both the survey and the field study. We can see that all limitations that might

TABLE 5.5: *Survey Results*

Test		**Result**
	Open Scenario	16.67%
Device Placement	Wall Scenario	27.78%
	Small Scenario	55.55%
Average Volume		4.7
Users who voluntarily ever switched off Echo		27%
Users who systematically switch off Echo		6%
Users who ever connected headphones		0%
Users who change volume via manual commands*		27%
	Never	66.67%
Users who turn off Echo's microphone	Rarely / Very Rarely	22.22%
	Sometimes	11.11%
Users who use Echo as a Bluetooth speaker		44.44%
Users who know the meaning of the green light ring		27%
Users who follow security procedures		6%

***** Including those who stated to use both voice and manual commands.

stop the attack have a very low likelihood, hence they are not a real threat for AVA. Among events that might alert users, results show that it is not very likely that users realise that a phone call is taking place, so an adversary would probably be able to eavesdrop on the users. However, the fact that users can hear the self-issued payloads is confirmed to be the main limitation of the attack, hence the adversary needs to carefully choose when to send commands to Echo. All other events in the table are mostly ignored by Echo users, and only expert ones would be possibly alerted by their presence. Hence, we argue that most limitations for the AVA attack remain theoretical.

TABLE 5.6: *Evaluation of the Limitations*

Limitation	**Likelihood**		**Impact**
Echo's microphone is muted	Low	(11%)	●●●●
Echo is unplugged from power	Low	(6%)	●●●●
User says "Alexa, stop"	Low	(0%)	●●●●
Headphones are connected to Echo	Low	(0%)	●●●●
User hears payload (same room)	High	(100%)	●●●
User hears payload (adj. room)	Medium	(66%)	●●●
User notices ongoing call (green light)	Medium	(27%)	●●●
User hears payload (non-adj. room)	Low	(0%)	●●●
User notices inaccuracies in replies	High	(100%)	●●
User cannot change volume by voice	Medium	(73%)	●●
User realises light ring is blinking	Low	(16%)	●●
User notices delay before the reply	High	(100%)	●

●●●●: It might stop the attack. — ●●●: It may alert any user. — ●●: It is ignored by most users, but can still alert an expert user — ●: It is ignored by all users.

5.5 Perceived Impact by the Population

During the two months following the responsible disclosure, AVA received a lot of attention from worldwide media. Notably, in the UK, AVA was featured by popular news websites such as *The Register*[1] and by the BBC's podcast *Digital Planet*.[2] The work received media coverage from important American websites as well, such as *Ars Technica*,[3] and from other news websites in different languages through the world. The attack was mostly perceived as very impactful by the readers and viewers of the published contents, and a demonstrative video showing how an attacker can use AVA to exploit the three vulnerabilities we found was viewed more than 20,000 times.[4]

5.6 Ethics of the Tests

Regarding the field study, we requested all users in the three households permission to place our Echo device in a specific room, and to collect usage data useful for the experiment, such as the given commands or the actions they performed during the different tests, and to use these data in an anonymised form only for the purpose of this research work. After the tests were performed, we explained to the users the tests we did and informed them of the results: all the users were again asked to either confirm or withdraw their permission to use the results in a scientific study. At no point were any of the user's private data accessed by third parties or by us. In addition, all user utterances saved on the Mask Attack database and on the Amazon Cloud (including recordings of the commands that are stored on the Amazon Cloud by default) were deleted after summarising in an anonymised format the results of the tests. Note that all the tests performed on Echo only involved

[1] https://www.theregister.com/2022/03/03/amazon_alexa_speaker_vuln/

[2] https://www.bbc.co.uk/programmes/w3ct31y5

[3] https://arstechnica.com/information-technology/2022/03/attackers-can-force-amazon-echos-to-hack-themselves-with-self-issued-commands/

[4] https://www.youtube.com/watch?v=t-203SV_Eg8

accounts, emails, and phone numbers owned by us, and no user nor third-party data were accessed during the experiments. Regarding the user survey, all the questionnaires were administered online, and users were asked for their explicit consent to use the collected information for research work. Participation in the survey was voluntary, and users were informed they could withdraw their consent at any time. Finally, the permission forms of the field study and of the survey were extensively analysed through an internal formal assessment, as mandated by our institutional policy, and these activities did not identify any danger or ethical concern.

5.7 Limitations of the Field Study and Survey

The field study was performed in June 2021, during the coronavirus pandemic, hence, finding people that were complete strangers and that wanted to help in a scientific study was a very challenging task. Therefore, we asked people we already knew, who already owned an Amazon Echo device, if they could have helped us in the study. While they did not know any detail of the experiments we were going to perform, there was some sort of implicit bias in their behaviour, as they already knew the researcher that was going to perform the study in their households. This leads to the following questions tied to the generalisability of our results:

1. Would the users leave a *complete stranger* in a room with their Echo device for 25 seconds (i.e., enough to make the adversary perform the Bluetooth connection)?

2. Is the answer to the previous question affected by the cultural background (e.g., nationality) of the users, or by their age?

Unfortunately, there is no way to answer these questions without making another field study with a larger number of households, such as ten, twenty, or more. Even then, the tests must be performed manually by a researcher

that has to be physically in the household: hence, this quickly becomes very time-consuming and a lot of financial resources are needed, if the research needs to take into account the behaviour of people with very different cultures, such as households from different countries and continents.

Furthermore, to not introduce bias during the tests in which the users had to say whether they had heard the self-issued commands at 3.5m or 7m, we had to exploit the moments in which they were not in the room with us, or we had to make them move to another room with an excuse. However, the users in the household might not want to leave the room while there is a complete stranger in it, even if they are a researcher, creating additional problems in the field study process. Hence, for future research, we hypothesise that having a team made of different researchers around the world, who perform tests with people they know, would be the best way to perform this study at a bigger scale. To check with more precision if users are willing to leave a complete stranger in the room with their Echo device for 25 seconds, another group could be created, in which only this experiment is performed, so that the other results are not affected.

While in the survey there is a bit more diversity, both in nationalities and age gaps, it would still be useful to perform the survey again with a larger user group, such as one hundred users or more, to confirm the generalisability of the results we obtained. This improvement should be easier than the one on field studies, as the surveys can be administered online.

5.8 Related Work

We now explore other works that analyse the feasibility of attacks in the cyber-physical domain. Some of these works leverage user studies, surveys, or practical experiments to support their results and assess limitations.

Nair et al. [125] investigate the feasibility of an attack that violates the privacy of users in the metaverse with a user study featuring 30 participants. In

this study, users had to interact with a Virtual Reality game while a hidden malicious feature in the game extracted 25 private data attributes from the users' interaction, without any permission available on the device it was running on. These data attributes allow the adversary to infer the demographics of the user playing, including sensible information such as health status (e.g., color blindness, eyesight, disabilities). Phan et al. [130] evaluate their so-called "Universal Adversarial Perturbation" on neural networks for image classification. To do so, they assess the adversarial samples' stealthiness, their efficiency compared to samples generated with other techniques, and the generalisation of their approach. Interestingly, differently from us, they did not perform a user study to double-check their claims regarding stealthiness, although an image that compares genuine images and adversarial images generated using their method is available within the paper to support the claim that the perturbation is indeed unnoticeable.

Beavers et al. [19] assess the feasibility of simple attacks to real pacemakers. In this case, recruiting real users for the study was unfeasible because of the strong ethical implications and the risks involved. Hence, only pacemakers of deceased patients were used for the study, after deleting all confidential data within the devices. The authors do not disclose the technical details of their tests, however, they report to have tested radio frequency attacks primarily, such as replay attacks and signal jamming. Finally, Alotaibi et al. [8] investigate whether AI can be used in conjunction with thermal cameras to analyse residual heat traces on keyboards or ATMs to infer user PINs. They perform two user studies: the first is performed to assess the overall impact on the attack of (i) the age of the heat trace, (ii) the user's typing method, and (iii) their password length, while the second is performed to check whether the material of which the keyboard is made has an influence or not on the heat trace and on the attack itself. Their user study featured 21 users and was able to highlight both strengths and limitations of their attack.

5.9 Summary

In this chapter, we discussed the feasibility, impact, and limitations of AVA when deployed in real-world scenarios, with the aid of three voluntary households who took part in a field study, and of 18 Echo users who answered a survey. We discovered that AVA is feasible in the real world, that the adversary can reliably execute critical commands that can undermine the victims' safety and privacy, and that most limitations for AVA remain theoretical.

Chapter 6

A Taxonomy of Measures Against Voice Spoofing Attacks

In this chapter, we look at security features of commercial voice-controllable devices. Subsequently, we show that current countermeasures against voice spoofing attacks (hence, against self-issue attacks as well) do not match the necessities of the users, and to address this we present a taxonomy of possible security settings to be applied to voice-controllable devices. Contents of this chapter have already been partially discussed in the following paper:

- Esposito S., Sgandurra D., Bella G., Protecting Voice-Controllable Devices Against Self-Issued Voice Commands. Accepted to the *8th IEEE European Symposium on Security and Privacy*. 2023.

6.1 Introduction

Identifying proper countermeasures against an attack is a vital step in the offensive security process. Within the previous chapters, we have seen how self-activation is a pervasive problem that is present on smart speakers and on other platforms as well, e.g., Windows and Android. Liveness detection, that is, understanding whether the command given to a device comes from a real user or from a speaker, is currently the state-of-the-art solu-

tion against voice spoofing attacks [113, 24, 176], and command self-issue falls within this category of attacks. However, in this chapter, we show that liveness detection-based countermeasures cannot be applied blindly to all voice-controllable devices in all real-world scenarios. The main reason is that users with severe speech impairments may rely on Augmentative and Alternative Communication (AAC), and specifically on Speech Generating Devices (SGDs), to communicate with other people and to operate their smart devices [134, 83, 155, 39, 137]. SGDs emit artificially generated voices, hence enabling liveness detection pervasively would hamper these people's interaction with their voice-controllable devices. At the same time, the few existing countermeasures specifically developed against self-issues (e.g., the solution proposed by Pogue and Hilmes [131]) do not adequately protect all devices.[1] For example, the AvA attack has a 9.8 CVSS score and is classified as Critical, however, its exploitation via Bluetooth has not been properly addressed yet [127]. Therefore, the threat posed by self-issue attacks to users and VCDs is usually high.

Countermeasures against voice spoofing commands can be set up to work against self-issued commands, however, we will show that implementing them pervasively would not benefit all users. To demonstrate this, we first provide a security analysis of existing VCDs, showing how authentication in voice channels is their main weakness. We then present a security-usability taxonomy that manufacturers can implement to allow users to choose the desired level of security for their device, based on their necessities.

6.2 Security Features and Weaknesses in VCDs

We analysed the security features of most commercial VCDs available worldwide. In the following, we report our findings on their basic security and privacy features and weaknesses.

[1] `https://www.ava-attack.org/`

6.2.1 Features

Our analysis shows that most devices provide basic authentication, encryption, and automatic updates, which we summarise in the following.

- **Backend Authentication**: the user is usually required to log in to the service provider's server to access the features of the VPA via, e.g., an Amazon, Google, or Apple account to complete the device setup. This authentication process usually requires a strong password, and the user can also enable two-factor authentication.
- **In-Transit Data Encryption**: data from the device to the companion app or to the provider's servers (and vice-versa) is usually encrypted with TLS.[2][3][4]
- **Automatic Updates**: the vast majority of commercial smart speakers checks for updates on a regular basis and installs them automatically. By contrast, this is not common for other voice-controllable devices, such as personal computers and smartphones, as the user usually has to manually install updates: this is because they might require a reboot of the device, and flexibility on when to install them is given to the user to avoid disruptions.

6.2.2 Weaknesses

Our analysis also shows that most VCDs share some security and privacy problems, which are reported in the following.

- **Weak Voice Authentication and Authorisation**: although authentication is usually strong on the backend, as discussed above, the same robustness is not guaranteed on the voice channel, which is the main input vector for VCDs. While most devices can be trained to recognise the

[2] https://privacy.commonsense.org/privacy-report/Microsoft-Cortana
[3] https://d1.awsstatic.com/whitepapers/White%20Paper-Alexa%20Confidentiality%20and%20Data%20Handling%20Overview%20Dec%202019.pdf
[4] https://support.google.com/product-documentation/answer/9293126

different users in a multi-user environment, the device owner cannot assign different permissions to each of them. Hence, the device will accept any command from any user, including payments [55]. Although there is the possibility to set up PIN numbers in some devices to protect such critical functionalities, the PIN must be pronounced out loud, potentially disclosing it to anyone in physical proximity of the interaction.

- **Always Listening**: although the majority of the devices features a light or a message on-screen that notifies the user that the wake-word has been recognised and a command is being recorded, the device is always waiting for such a wake-word, so it is always listening. Hence, it may activate at the wrong time and record private conversations [54, 161].

- **Bluetooth Connection with no PIN**: smart speakers can act as Bluetooth speakers for other devices. The pairing process, however, does not require any credential (e.g., PIN number) and can be initialised with a voice command. Because of the weak voice authentication discussed above, an attacker in the proximity of the device can connect via Bluetooth to smart speakers that have this functionality, allowing the adversary to exploit the self-issue vulnerability [56].

6.2.3 Addressing the Voice Channel Insecurity

The main takeaway point of our security analysis of commercial VCDs is the poor protection of the voice channel. We want to emphasise that the voice channel is the main input vector for these devices: therefore, poor protection of this channel may actually jeopardise the overall security of VCDs as an adversary can execute self-issued voice commands by improperly using the (unauthenticated) voice channel. To address the current insecurity of voice channels, we start with the understanding that blocking self-issued commands is a necessary security requirement in all scenarios and for all users, as self-issued commands are inherently malicious. Concerning the rest of the

synthesised voice commands, such as real-voice recorded commands or TTS commands, we reckon that some categories of users may need to block only some types of synthesised voices instead. For instance, they might want to whitelist only one synthetic voice if they trust it to be unique to their SGD, or they might want to allow all artificial voices in scenarios where a variety of such voices interact legitimately with the voice-controllable device. For this reason, we next present a taxonomy of usability and security settings for blocking or allowing synthetic voices (including self-issued commands) that takes into account different categories of users and realistic scenarios, by outlining the pros and cons of each of these settings.

6.2.4 Synthesised Voices for Issuing Commands to VPAs

Recent work in the field shows that disabled individuals actually benefit from being able to issue commands to their VCD using artificial voices. Kane et al. [83] present a study of users affected by Amyotrophic Lateral Sclerosis who can send text messages to their contacts by making Siri and Echo recognise their synthesised voices. Similarly, Pradhan et al. [134] analyse online reviews for Amazon Echo that show that disabled individuals experienced an improvement in their quality of life, e.g., thanks to the use of synthesised voice to issue commands to Echo. Velasco-Álvarez et al. [155] developed a Brain-Computer Interface which allows disabled individuals to command Google Assistant through synthesised speech. Corso [39] evaluates the improvements in the quality of life of SGD users when options to control smart speakers are added to the SGD they already use, while the development of a tailored solution is discussed by Ryu et al. [137].

6.3 Classifying Security vs. Usability

We present a taxonomy of security settings that progressively reduces the classes of synthetic voice commands that are allowed to issue valid voice

commands. In this taxonomy, we rank the levels from the most usable (and least secure) to the most secure (and least usable).

- **Level 0 — Accept all synthesised commands**: the minimum level of security available, where any command is executed. This tolerance level should never be selected unless there are critical usability problems that prevent the user to interact with the voice-controllable device.

- **Level 1 — Accept synthesised commands, but detect and discard self-issued commands**: a good compromise between usability and security. While synthesised commands may be useful for some users, self-issued commands are malicious in nature and can be blocked without further analysis. This setting should be enabled on devices that receive legitimate commands from a variety of synthetic voices, or if their user does not want to (or cannot) undergo the training procedure to make the device learn their synthetic voice, which is explained in Level 2.

- **Level 2 — Accept synthesised commands from known synthetic voices. Detect and discard all other synthesised commands including self-issue**: the best compromise between usability and security in this taxonomy. People who use synthetic voices to communicate could undergo a brief training session with the voice-controllable device, so it can learn to recognise their synthetic voices.[5] All received commands are sent to a generic solution against voice spoofing attacks for analysis: if the command is classified as synthesised, the device checks if it knows the voice issuing the command. If it does, the command is executed as coming from a trusted synthesised voice. This grants protection against non-targeted attacks,[6] while allowing a good user experience.

[5] Theoretically speaking, a user who has recordings of their own voice before the illness (i.e. before they became speech impaired) could also use deepfake technologies to create a clone of their voice to issue commands. However, we were not able to confirm if someone was already able to do this or not.

[6] If the voice profile used by the SGD is publicly available, such as Google TTS, an adversary could use it to record voice commands that will be then issued to the device, bypassing the security Level 2 as the voice would be recognised as trusted.

- **Level 3 — Discard all synthesised commands**: the maximum level of security available, where all detected synthesised commands are discarded, independently from their origin. This tolerance level is extremely good for those users who do not need synthetic voices to be recognised, as it protects against most of the attacks we know that make use of the voice channel. However, people who communicate via SGDs will be negatively affected by this setting, as they will not be able to interact with their device as they did before.

TABLE 6.1: *Security-Usability Taxonomy Summary*

Setting	Self-Issue	Synth Commands	Security	Usability
Level 0	✓	✓	●	● ● ●
Level 1	⊘	✓	● ●	● ● ●
Level 2	⊘	!	● ● ●	● ●
Level 3	⊘	⊘	● ● ● ●	●

Self-Issue: ✓ if self-issued voice commands are allowed in the current setting, ⊘ if they are blocked. — **Synth Commands:** ✓ if synthesised voice commands are allowed in the current setting, ⊘ if they are blocked. ! indicates that not all synthesised commands are blocked, i.e. some artificial voices are allowed to issue commands.

Table 6.1 summarises the pros and cons of the four levels in our taxonomy, in terms of their usability and security. Currently, most commercial voice-controllable devices only implement Level 0 [56, 82, 48, 6, 171, 35, 120]. Some of these devices implement countermeasures against replay attacks and self-activation, i.e., theoretically at Level 1 — however, they are not currently capable of preventing exploitation from self-issued commands [56]. Most of the research is currently focused on Level 3, namely research on liveness detection and ASV, which will be better discussed in Section 7.9. Instead, currently, no existing solution fully implements Level 1 and Level 2.

6.4 Implementation Considerations

To create the taxonomy we presented in the previous section, we first assessed the categories of commands that were relevant for our study, that is, self-

issued commands and synthesised commands generated with any solution. Note that all of these theoretically fall in the voice spoofing attack category, however, we have already mentioned that not all synthesised commands are malicious in nature, while self-issued commands always are. From this, we get the intuition that self-issued commands *must* be blocked, while synthesised commands *might* be blocked, depending on the circumstances.

Therefore, when it comes to the implementation of the taxonomy, it is easy to understand that Level 0, the least secure one, should allow all commands to be executed. After that, the next step in increasing the device security is blocking the only kind of commands that we surely know we must block, that is, self-issued commands. Hence, we get that in Level 1 we block self-activations, but not other synthesised commands. Then, the next step in increasing the device security could be either (i) blocking all synthesised commands, not allowing SGD users to interact with the VCD, or (ii) blocking only synthesised commands that come from unknown voices — as the latter reduces the overall security of the device when compared to the former, because the already mentioned targeted attacks become possible, we choose it as Level 2. Hence, the former becomes Level 3, in which all synthesised commands are blocked and users are protected from all synthetic voices.

6.5 Assessing Security vs. Usability at Present

We believe that a proper balance between usability and security needs to be found for voice-controllable devices, as countermeasures discussed in the literature against voice spoofing attacks cannot be applied pervasively without taking into account the necessities of all users. As already discussed, while countermeasures, such as liveness detection, are currently the state-of-the-art against this kind of attack, they can disrupt the user experience for users interacting with voice-controllable devices by means of SGDs. At the same time, VPA devices should *always* be protected against self-issue attacks,

which are a class of synthetic voice commands specifically performed by bad actors to perform voice spoofing attacks. For these reasons, in the next chapter, we present, implement, and evaluate a Level 1 countermeasure, i.e., a countermeasure against self-issue attacks that at the same time accepts synthesised commands.

6.6 Related Work

In this section, we examine other relevant taxonomies on countermeasures against voice spoofing attacks and on other related fields.

Khan et al. [88] thoroughly analyse voice spoofing attacks and their countermeasures. Their taxonomy for the latter includes two categories: *conventional* and *deep learned*. The conventional category includes methods that employ handcrafted features such as Mel Frequency Cepstral Coefficients and prescribes the analysis of front-end features (e.g. spectrum analysis) or backend classification, which can also involve deep learning. Instead, in the deep learned category find place all solutions that learn to recognise key features from audio files, in an attempt to extract the most relevant ones for the task. Our solution against self-activation, that will be presented in the next chapter, falls in this category. This taxonomy also includes the already mentioned End-to-End models which are proving themselves to be more effective than classic deep learning in several fields.

Hewitt and Cunningham [72] create a taxonomy for voice assistant software and hardware. While their analysis of voice assistant software is limited to the most used commercial ones, their analysis of hardware, i.e. Voice User Interfaces and especially edge devices — sensors and appliances essentially — is thorough and precise.

Karim et al. [86] thoroughly classifies all attacks and known countermeasures to Internet of Vehicles (IoV) attacks. Interestingly, in their taxonomy of possible attacks we find *"attacks on voice controllable systems"* within the

"Attacks on In-Vehicle Systems" category, as we have previously seen in Section 3.3.1 that car manufacturers as BMW developed and embedded their own voice personal assistant within their cars. Hence, they outline possible defensive measures to be applied to thwart these attacks, although at a lower level of detail than the previously cited work, as attacks on the voice personal assistants embedded within cars are just one of the many attacks on the IoV.

6.7 Summary

In this chapter, we assessed the basic security and usability features of voice-controllable devices, confirming that their main weakness is the lack of authentication and authorisation on the voice channel. After, we analysed the security requirements for users of such devices, and we introduced a taxonomy for synthesised voice command tolerance. Finally, we showed that existing countermeasures against voice spoofing attacks cannot be pervasively applied as they would cut out users who require a synthesised voice to communicate, such as people with severe speech impairments, and that Level 1 and 2 of our taxonomy can be implemented to allow them to use their device while being protected from self-activations. Unfortunately, as of today, there are no reliable implementations of Level 1 and 2 countermeasures.

Chapter 7

An Intelligent Measure Against Self-Activation

In this chapter, we introduce our solution to protect voice-controllable devices from self-issued voice commands, implementing the already mentioned security Level 1. We also describe current voice spoofing countermeasures, which can be used to implement security Level 3 (RG3). Contents of this chapter have already been partially discussed in the following paper:

- Esposito S., Sgandurra D., Bella G., Protecting Voice-Controllable Devices Against Self-Issued Voice Commands. Accepted to the *8th IEEE European Symposium on Security and Privacy.* 2023.

7.1 Introduction

We now discuss the implementation of security Level 1 by introducing a novel anti-self-activation solution to detect whether commands are self-issued or not. The reference scenario considers a device playing an audio file and, at the same time, capturing a command. As the device can access both audio files, it can compare and check them for similarities. The core novel idea underlying our solution is that, if the captured audio has substantial differences from the played one, then a real user's voice is partially overlapping what is

being played — hence, the command comes from a live user and can be executed. Vice versa, if the two files are very similar, it is likely that the voice command was pre-embedded within the played audio file — hence, the command needs to be discarded as self-issued.

As the main research challenge revolves around similarity checks, we fine-tune a Twin Network [47, 178, 122, 111] structure to detect differences in audio. Twin Networks have several applications in different security scenarios: for instance, Dey et al. [47] present SigNet, a framework to verify whether a signature is genuine or not. The work uses a Twin Convolutional Neural Network and outperforms the state-of-the-art. Another application of Twin Convolutional Networks is facial similarity, namely, detecting whether two photos feature the same person or not [69]. From a user safety perspective, Droghini et al. [53] apply Twin Neural Networks to human fall detection, which is considered a hard problem as there are not many samples and datasets featuring people falling to train detection solutions. To this end, their proposed approach uses few-shot learning and achieves an 80% F1 score.

In our solution to the self-activation problem, we extract the Mel-Spectrogram from every captured and played audio file, and we exploit a Twin Network architecture to classify the audio. Our solution correctly classifies 97% of voice commands after being trained on a dataset we created with 35 pairs of captured/played audio files and their augmentations. With our solution, we aim to make a substantial step towards a good balance between usability and security in voice-controllable devices.

7.2 Refining the Threat Model

Our solution implements the Level 1 security of our taxonomy, and, as such, is designed to protect voice-controllable devices against self-issued commands. Hence, the VOCODES Threat Model discussed in Chapter 3.4 still applies. However, a few more details are due to clarify the reference scenario.

Applications for smart devices (such as a malicious Music & Radio skill or Mask Attack) run on the cloud, and VPAs only send queries and read replies aloud. However, Windows and Android malware run on the attacked device: in this case, depending on the privileges available to the malicious applications, they may be able to disable protections in place on the attacked machine, including our solution, before self-issuing any command. Nonetheless, in our threat model, we do not consider the attacker to have the ability to disable security mechanisms on the target. Furthermore, as security Level 1 of our taxonomy allows external synthesised voices to issue commands to the device for usability purposes, we only consider self-issue attacks as malicious, while all other voice spoofing attacks are out-of-scope.

7.3 Proposed Solution

The main idea underlying our solution to address the self-activation problem is to consider both the input and output audio channels when the voice-controllable device is receiving a command. Our solution compares the audio file that is being played (the *played* audio file, henceforth) with the audio file obtained while recording the command coming, supposedly, from the user (the *recorded* audio file, henceforth). If a user effectively issues a command, the *recorded* audio file will contain both (i) what is being played by the smart speaker and (ii) the user command. Therefore, the *recorded* audio file will be sufficiently different from the *played* one. Conversely, if the *played* audio file contains a command that triggers a self-activation, the *recorded* audio file will not be different from the *played* audio file as the audio captured by the smart speaker's microphone is exactly the one of the *played* audio file.

Figure 7.1 gives a visual representation of both scenarios: in the benign case, the *recorded* audio is different from the *played* audio, as the voice of the user overlaps the backing track (top-right figure); by contrast, in the malicious case, the *played* audio and the *recorded* one match as the evil command is

embedded in the original audio (bottom-left and bottom-right figures).

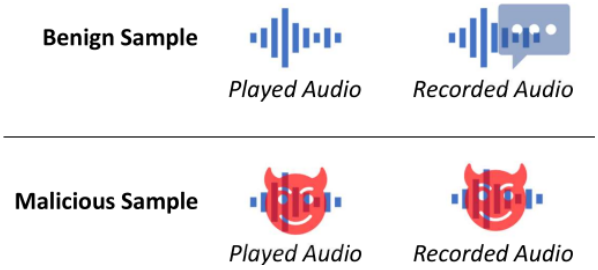

FIGURE 7.1: *Benign and Malicious Samples Compared*

7.3.1 Introducing Twin Networks

Unfortunately, a direct comparison between the soundwaves of the *played* and the *recorded* audio files cannot be performed to detect self-issues attacks, because distortions, reflections, background noise, and other artifacts that significantly alter the *recorded* audio from the *played* one. Hence, we needed more precise means to compute this difference. This is why we chose to use neural networks. In particular, we note that Twin Neural Networks (TNNs) are better suited to detect audio differences by learning relevant discriminating features, and therefore can generalise well our problem. Hence, we decided to use TNNs in our proposed solution as: (i) TNNs are well-suited for problems of outlier detection [47, 178, 122, 111], (ii) TNNs can work well with small datasets, (iii) TNNs are natively designed to compare pairs of similar inputs, (iv) other euristic approaches shown in different patents do not seem to work [3, 131, 100].

The choice of TNNs is particularly important in this context to provide precise predictions (RG7) with only a small dataset for training (RG5) as there are currently no datasets available that feature self-issued commands, or datasets featuring both played and recorded audio samples. For this reason, we also created and released a dataset for training a TNN, which will be introduced in Section 7.4. Additionally, we will see that our solution performs better than other simpler machine learning models running in the same con-

ditions, such as One-Class SVM, as they are not able to generalise the problem on their own (i.e., without a neural network helping in extracting features).

The structure of our Twin Neural Network is shown in Figure 7.2. As we can see from the figure, the network consists of a single Convolutional Neural Network (CNN). Note that during training, its weights and biases are updated after both the *played* and the *recorded* samples have been fed into it. The vectors output by the CNN are then compared by means of the pairwise distance using the 2-norm. In the training phase, this value is used to compute the Contrastive Loss, which will be the criterion for the network's training. Instead, in the validation phase, if the pairwise distance exceeds a certain threshold, the current sample is classified as benign, meaning the differences between the *recorded* and *played* audios are large enough, so it is likely that the command was not embedded in the *played* audio but was pronounced by a legitimate user. The final value of the threshold was selected by exploring different values, and by analysing the resulting confusion matrix for different training instances of our solution. In the end, we selected 0.4 as it achieves the best performance overall (see Section 7.5 for performance details).

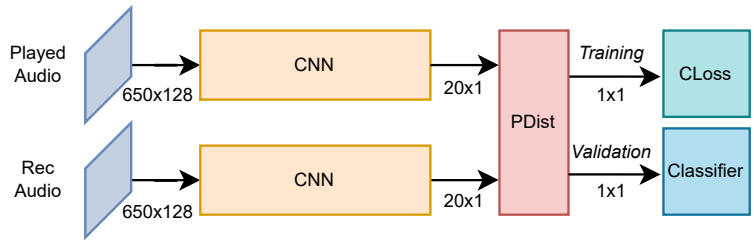

FIGURE 7.2: *Structure of the Twin Network*

Figure 7.3 details how we fine-tuned our CNN for the self-issue detection problem. In particular, the network takes a single Mel-Spectrogram as input, with dimensions 650x128 (83,200 pixels), which goes through a convolutional layer with the ReLU activation function. A 1px zero-padding is applied to the input image, to address the border effect [80]. After convolution, a MaxPool layer extracts relevant information within each 2x2 area. It then performs

Batch Normalization [79] and Dropout [144] to regularise features. This process is repeated five times, reducing progressively the number of used channels and the number of features per channel by means of convolution and pooling. After the fifth dropout, only 456 features are left. These features are then fed to three consecutive dense layers, and a final vector of 20 features is output by the network. Table 7.1 summarises the implementation of the network and its layers.

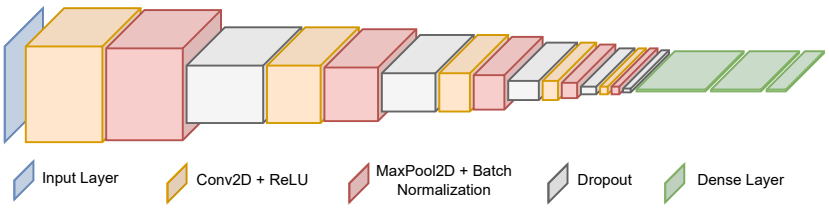

FIGURE 7.3: *Visual Representation of the CNN in Our Twin Network*

TABLE 7.1: *Convolutional Network Structure*

Layer	**Output Size**	**Channels**	**Notes**
Input	650x128	1	-
Conv2D + ReLU	646x124	60	Kernel 7x7, Padding 1
MaxPool + BatchNorm	323x62	60	Size 2x2, Stride 2
Dropout	323x62	60	p=0.25
Conv2D + ReLU	319x58	48	Kernel 7x7, Padding 1
MaxPool + BatchNorm	159x29	48	Size 2x2, Stride 2
Dropout	159x29	48	p=0.25
Conv2D + ReLU	157x27	36	Kernel 5x5, Padding 1
MaxPool + BatchNorm	78x13	36	Size 2x2, Stride 2
Dropout	78x13	36	p=0.25
Conv2D + ReLU	76x11	24	Kernel 5x5, Padding 1
MaxPool + BatchNorm	38x5	24	Size 2x2, Stride 2
Dropout	38x5	24	p=0.25
Conv2D + ReLU	38x5	12	Kernel 3x3, Padding 1
MaxPool + BatchNorm	19x2	12	Size 2x2, Stride 2
Dropout	19x2	12	p=0.25, 19x2x12 = 456 features
Dense + ReLU	300	-	-
Dense + ReLU	100	-	-
Dense + ReLU	20	-	Output Layer

7.4 Realistic Commands Dataset Creation

In this section, we describe the equipment we leveraged to create the voice-commands dataset (Section 7.4.1) that we use to train and evaluate our solution. This was necessary, as we are not aware of any open-source dataset that features both the *played* and the *recorded* audio files. Hence, we describe the process for its recording and the different categories of samples within it, and also the data augmentation techniques we implemented to enhance the training process (Section 7.4.2).

7.4.1 Equipment

We used a Seeed Respeaker 4-Mic Microphone Array v1.1 [141], connected to a Raspberry Pi 4 Model B,[1] to record the audio dataset. The device was placed on a table with other objects nearby to simulate a real-world environment where the VCD is placed near other appliances. The *played* audio files are various extracts of songs, podcasts, and audiobooks. To generate the *recorded* samples, we placed the Raspberry with the Respeaker microphone array on top of a 3rd Generation Echo device, while Echo streamed the *played* audio files and the Respeaker array was recording. A script coordinated the *played* audio files and their recording. To train the neural network we used a virtual machine on Google Colab, with a Tesla P100-PCIE-16GB GPU, an Intel Xeon CPU @ 2.20GHz, and 26 GB RAM. Figure 7.4 shows the placement of these devices during the recording of the dataset.

7.4.2 Building the Dataset

First, we identify the possible categories of audio files that can be legitimately played by the user with their smart speaker (*benign samples*), and then the categories of rogue audio files containing voice commands that can be issued by the attacker (*malicious samples*).

[1] `https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/`

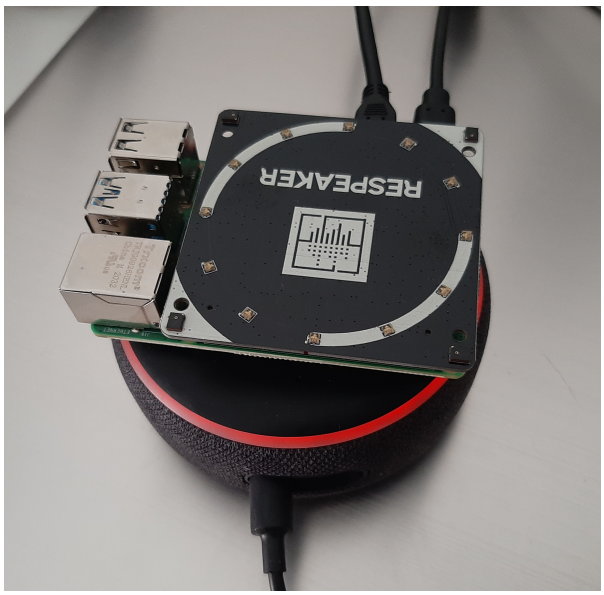

FIGURE 7.4: *Placement of Echo Dot and Respeaker 4-Mic Microphone Array During the Recording of The Dataset*

Benign Category

Using the classification in Chapter 4.3 as a starting point for possible kinds of audio files that the user could be listening to while issuing a command, the benign category includes:

- **Text-To-Speech Samples**: artificially generated speeches that do not contain malicious commands. For example, the user could be listening to an audiobook read by an artificial voice.
- **Real Voice Samples**: real people speaking. For instance, the user could be listening to a podcast or to a radio show.
- **Music Samples**: any audio containing music. For example, the user could be listening to a song, or they could be listening to the radio.
- **White Noise Samples**: artificially-generated noise with a certain frequency. These can be sleeping or relaxing sounds to facilitate relaxing, studying, or having a baby sleep.

This list is not meant to be exhaustive, as more audio types or combinations of these audio types are not part of our dataset (e.g., rain or bus noise instead of simple white noise, TTS voice with music in background, real voice with music in background, *a cappella* music).

Malicious Category

This category, coherently with the classification in Chapter 4.3, includes:

- **Text-To-Speech Samples**: synthesised speeches that, contrary to the ones in the benign category, contain malicious commands. The attacker could have included them, for example, within the audiobook that is being read. We generated all of our samples using Google TTS, with varying Wavenet profiles. There was only one restriction on which commands could be generated, that is, they had to be in the classic wakeword + command form.

- **Real Voice Samples**: real people speaking, and pronouncing malicious commands. The attacker would record them within a podcast, or suddenly issue them during a live show. We had a real user recording these samples, who read the command from a prebuilt list. Even in this case, the only restriction for the selection of commands to be added to the list was that they had to be in the classic wakeword + command form.

- **Adversarial Noise Audio Samples**: audio files containing adversarial noise commands. These could be virtually included in every media, although they are usually hidden in music [35, 171]. We generated our samples using the Devil's Whisper Dockerhub.[2] Because we did not retrain the solution on other commands and we did not introduce other songs, we were restricted to use the songs and the commands already present in the Dockerhub.

[2] `https://hub.docker.com/repository/docker/neeze/devilwhisper`

Audio Generation and Recording Process

We generated 10 audio samples for each category, with varying sample rates, using the Waveform Audio File (WAV) format. The *played* audio dataset is composed of 70 samples, with an average length of 6 seconds. As our goal is to check whether our solution is able to correctly classify commands with a limited number of samples (recall RG5 introduced in Section 1.2.1), 70 is a very low number of samples when compared to other datasets, such as the ASVSpoof 2019 *Logical Access* and *Physical Access* datasets, which are made of 50,224 and 83,700 samples, respectively.[3]

To create the *recorded* audio dataset, we reproduced the *played* audio files using the setup described in Section 7.4.1. The audio files of the malicious samples already contain the malicious command. To simulate real benign commands being issued to the smart speaker, a real user was involved in pronouncing commands to record benign samples. To this end, a script records all the samples and prompts a real user to give a command when the *played* audio starts. Since one *recorded* audio file is generated for each *played* audio file, the *recorded* audio dataset is composed of 70 samples as well.

The whole process of generating a played-recorded audio file pair requires four steps:

1. **Download**: we downloaded audio files for *played* samples among podcasts, songs, and shows, to introduce diversity among the samples.

2. **Split**: we split these files in short samples, as audio files captured by real voice-controllable devices when a command is issued are only a few seconds long.

3. **Select**: we built the *played* dataset by identifying suitable candidates among these samples. For example, we discarded samples with too much silence or samples that are similar to previously selected ones.

[3] `https://datashare.ed.ac.uk/bitstream/handle/10283/3336/asvspoof2019_evaluation_plan.pdf`

4. **Record**: we generated a list of different and valid commands for a real user to read while recording the voice command on top of the *played* audio file. A script then aligned the *recorded* audio with the *played* one: this required a left-shift of ∼90ms (mostly hardware dependant). The part of the audio that was left-shifted to the end of the file was trimmed.

The recordings were made with the real user staying at a 60 cm distance from the device, in a room with 20 dB of background noise and the volume of Echo Dot set to 5.

Dataset Augmentation

All the samples in our dataset undergo a data augmentation process to allow better training results for our Twin Network. In fact, this process generates other samples from the existing ones and expands the dataset in an efficient way. We implemented five techniques:

- **Speed Up**: increasing the playback rate of the audio sample by 20%, while keeping the same pitch.
- **Speed Down**: decreasing the playback rate of the audio sample by 20%, while keeping the same pitch.
- **Pitch Up**: shifting up the pitch of the sample by 2 semitones.
- **Pitch Down**: shifting down the pitch of the sample by 2 semitones.
- **Frequency Masking**: deleting all information within two random frequency ranges. Because the chosen frequency range is random, every sample will likely have different frequencies missing. Unlike the other data augmentation techniques, which are performed on the raw audio file, this technique can be applied to Mel-Spectrograms, by applying horizontal white bars to mask the chosen frequencies. We applied two masks with dimensions of 650x6 pixels for each sample.

By adding up the original samples to the augmented ones, the dataset included a total of 420 samples. Note that a pair made of a *played* audio file and a *recorded* audio file only counts as one sample, despite them being two separate files. We never use our samples all at once: this is because the augmented samples are only used during training, and not for the validation process. Hence, the augmented samples belonging to audio files selected for validation are discarded for that specific instance.

Table 7.2 summarises the different datasets we used for each of the tests we have performed. To train our Twin Network, we make a homogeneous split selecting half of the samples and their augmentation for training. In this way, we use 20 benign samples and 15 malicious ones: with their augmentations, we get to 120 benign samples and 90 malicious samples, adding up to 210 training samples. We use the remainder for validation purposes, excluding the augmented samples, that is, we have 20 benign samples and 15 malicious ones in our validation dataset. These are identified with DNN (training) and DNN (testing) in Table 7.2.

As we are not aware of any other solutions to directly compare our work with, for benchmarking purposes, we used state-of-the-art Anomaly Detection (AD) techniques to address the same problem. Because AD algorithms are typically trained on benign samples only, we had to rearrange the sample distribution within the datasets, to make a realistic yet balanced comparison with our solution. In the AD1 dataset, we try to keep the same samples as the DNN dataset, however, because the malicious samples are excluded from the training, the result is that AD1 (training) contains 57% of the samples when compared to DNN (training), that is, it only contains the 120 benign samples. In the AD2 dataset, we try to keep the DNN dataset's total number of training samples, regardless of the balance. Again, because the training dataset does not contain malicious samples, this leads to the AD2 *testing* dataset having less benign samples: in fact, it is rather imbalanced, as 6 samples out of 7 are malicious. Recall that the AD datasets only contain the *recorded* audio sam-

ples, as anomaly detection solutions have no means to compare them with the *played* samples.

TABLE 7.2: *Dataset Structure*

Dataset	Benign	Malicious	Total	Type
Original Dataset	40	30	70	▶ ●
Augmented Dataset	40+200	30+150	420	▶ ●
DNN (training)	20+100	15+75	210	▶ ●
DNN (testing)	20	15	35	▶ ●
AD1 (training)	20+100	0	120	●
AD1 (testing)	20	15	35	●
AD2 (training)	35+175	0	210	●
AD2 (testing)	5	30	35	●

▶: includes *played* audio files. — ●: includes *recorded* audio files. — The number after the + sign indicates the number of augmented samples.

FIGURE 7.5: *Dataset Generation Flow*

Mel-Spectrograms Extraction

After performing the data augmentation, we extracted Mel-Spectrograms for each sample. In fact, even if it is possible to feed a raw audio file into a neural network, several works show that feeding Mel-Spectrograms yields better results for different tasks that involve time series [37, 30]. As Mel-Spectrograms focus on the human-hearable sounds, they give less relevance to infrasounds and ultrasounds by allocating them fewer pixels compared to the human-hearable frequency ranges. This allows the comparison of audio files with different frequency ranges, as long as human-hearable frequencies are present in both. All spectrograms were then resized to 650x128 pixels: this dimension was chosen as it was close to the average size of all spectrograms. Finally, we organised the audio files and the spectrograms in different directories, depending on their category and benign/malicious classification, and ulti-

mately labelled the whole dataset. The entire process used to create a sample is summarised in Figure 7.5.

7.5 Evaluation

In this section, we describe how we train the Twin Network (Section 7.5.1), we evaluate its performance during validation (Section 7.5.2), and we compare it with state-of-the-art anomaly detection (Section 7.5.3).

7.5.1 Training

The training is performed by letting both the *played* and *recorded* samples go through the network, one after another. The Contrastive Loss is used as the criterion for the training, and is calculated as [70]:

$$CLoss = (1-Y)\frac{1}{2}(D_w)^2 + Y\frac{1}{2}\{max(0, m - D_w)\}^2 \tag{7.1}$$

In this formula, Y is the correct label for the sample, D_w is the pairwise distance (using the 2-norm) between the two predicted vectors output by the Twin Network, and m is the margin, which was set to 1. The pairwise distance of two vectors v_0 and v_1 of the same dimension using the p-norm is [135]:

$$PDist_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}} \tag{7.2}$$

In this formula, n is the dimension of such vectors, x_i is the arithmetic difference between the i-th element of v_0 and v_1, and p is the norm. After these values are calculated, backpropagation allows the network's weights and biases to be updated. We train the network for 100 epochs using the Adam optimizer [91] and $\alpha = 5 \cdot 10^{-5}$ as the learning rate. For this task, we used the DNN training dataset, which contains 120 benign samples (20 + 100 augmented) and 90 malicious ones (15 + 75 augmented), for a total of 210 samples to be used for training purposes.

In the next section, we report the average performance of our models.

7.5.2 Validation

After each training epoch, we run a validation round against the DNN testing dataset to measure the network's performance. This dataset contains 20 benign samples and 15 malicious ones, for a total of 35 samples. During the validation phase, the pairwise distance between the two feature vectors output by the Twin Network — one for the *played* audio and one for the *recorded* one — is compared with the threshold (0.4 in our case, as explained in Section 7.3) and all samples are classified accordingly. We observed that the network usually reaches its best performance between the first 10-50 epochs, after which there is a small degradation of the accuracy — this may mean that the number of training epochs is too high compared to the number of samples available, and the Twin Network slightly overfits the training dataset.

The neural network achieves similar results in most training instances. Table 7.3 presents the confusion matrix for the best results of ten different instances of training: in five instances, the network manages to correctly classify all samples, except for one false negative (instances 1, 5, 6, 8, and 9). In three instances, there is one false positive instead (instances 2, 7, and 10). Finally, instance 3 is the least reliable, with 2 misclassifications, while instance 4 is the best as it correctly classifies all the samples.

Hence, our solution is capable of reliably detecting when a command is being self-issued (RG4) and it has been trained with only 35 original samples and their augmentation (RG5).

7.5.3 Comparison with Anomaly Detection

We compare our results with state-of-the-art techniques for anomaly detection as a baseline, as currently there are no other works against self-issue to compare our solution with. We performed four tests:

TABLE 7.3: *Confusion Matrices for 10 Training Instances*

#	TP	TN	FP	FN	Acc.	BA	F1
1	14	20	0	1	0.97	0.97	0.97
2	15	19	1	0	0.97	0.98	0.97
3	14	19	1	1	0.94	0.94	0.93
4	**15**	**20**	**0**	**0**	**1.00**	**1.00**	**1.00**
5	14	20	0	1	0.97	0.97	0.97
6	14	20	0	1	0.97	0.97	0.97
7	15	19	1	0	0.97	0.98	0.97
8	14	20	0	1	0.97	0.97	0.97
9	14	20	0	1	0.97	0.97	0.97
10	15	19	1	0	0.97	0.98	0.97
Avg	**14.4**	**19.6**	**0.4**	**0.6**	**0.97**	**0.97**	**0.97**

#: ID of the instance — **TP:** True Positives — **TN:** True Negatives — **FP:** False Positives — **FN:** False Negatives — **Acc.:** Accuracy — **BA:** Balanced Accuracy — **F1:** F1 Score — **Avg:** Average values

- **Test 1 — Anomaly Detection Baseline**: we trained two state-of-the-art anomaly detection techniques, namely, One-Class Support Vector Machine (OCSVM) [140] and Isolation Forest (iForest) [107] on the 83,200 features. Training and testing were performed using the AD1 dataset first, and then the AD2 dataset in a separate training instance. Recall that these datasets only include benign *recorded* audio samples.

- **Test 2 — Anomaly Detection with Resnet-18**: we used a pre-trained Resnet-18 to extract relevant features from the benign *recorded* audio files. OCSVM and iForest were then trained on the features extracted by the said neural network, which are only 512. The AD1 and AD2 datasets were used separately in this scenario as well.

- **Test 3 — Anomaly Detection with Resnet-152**: it works similarly to Test 2, this time with a pre-trained Resnet-152. This network outputs 2048 features. AD1 and AD2 were used in separate training instances.

- **Test 4 — Anomaly Detection with our Convolutional Network**: it works similarly to both Tests 2 and 3, this time with the anomaly detection models trained with the 20 features extracted by our Convolutional Network. While such a network was trained using the DNN dataset, the anomaly detection models were trained and tested with AD1 and AD2

separately, so that results are comparable with other tests.

Table 7.4 shows the results of the tests and compares the different solutions. As we can see, Test 1 gives rather poor results, as the anomaly detection models cannot figure out relevant features from 83,200 pixels per sample. As shown by the TPR and TNR values, the OCSVM always predicts that a sample is malicious, while the iForest always predicts that it is benign. This results in a balanced accuracy of 50% for both solutions.

The anomaly detection models start to correctly classify our samples only when a neural network is introduced to extract relevant features. In detail, OCSVM slightly improves its performance during Test 2, while iForest keeps classifying all samples as benign, except for one in the AD2 testing dataset, which is correctly classified as malicious. Isolation Forest's performance does not increase in Test 3. However, OCSVM is capable to achieve 80% balanced accuracy and a 0.75 F1 score, substantially improving its performance. These are improved also for the AD1 testing dataset when compared to those obtained in Test 2.

Test 4 shows that our network outperforms the generic Resnet in feature extraction for this specific task: in fact, with only 20 features, both anomaly detection techniques manage to substantially improve their performance with all testing datasets. In particular, we note that, in this case, Isolation Forests get really close in performance to our solution, but our solution is still more reliable in the average case (RG7), as we see from its balanced accuracy.

7.6 Real Use-Case Experiments

We now assess the performance of our solution when operating in different, realistic, conditions than the ones in which the dataset was recorded. To this end, we deployed an instance of our solution on a Raspberry Pi 4 Model B with a Seeed Respeaker 4-Mic Microphone Array v1.2. This is a newer version of the microphone that was used to record the dataset to introduce subtle differ-

TABLE 7.4: *Evaluation of Our Solution Against the Anomaly Detection Baseline*

Solution	Dataset	TPR	TNR	P	R	Acc.	BA	F1
Test 1 - OCSVM	AD1, AD2	**1.00**	0.00	0.43	**1.00**	0.43	0.50	0.60
Test 1 - iForest	AD1, AD2	0.00	**1.00**	-	0.00	0.57	0.50	-
Test 2 - OCSVM	AD1	0.33	0.80	0.56	0.33	0.60	0.57	0.42
Test 2 - iForest	AD1	0.00	**1.00**	-	0.00	0.57	0.50	-
Test 2 - OCSVM	AD2	0.27	**1.00**	**1.00**	0.27	0.37	0.63	0.42
Test 2 - iForest	AD2	0.03	**1.00**	**1.00**	0.03	0.17	0.52	0.06
Test 3 - OCSVM	AD1	0.73	0.45	0.50	0.73	0.57	0.59	0.59
Test 3 - iForest	AD1	0.00	**1.00**	-	0.00	0.57	0.50	-
Test 3 - OCSVM	AD2	0.60	**1.00**	**1.00**	0.60	0.66	0.80	0.75
Test 3 - iForest	AD2	0.00	**1.00**	-	0.00	0.14	0.50	-
Test 4 - OCSVM	DNN + AD1	**1.00**	0.20	0.48	**1.00**	0.54	0.60	0.65
Test 4 - iForest	DNN + AD1	**1.00**	0.80	0.79	**1.00**	0.86	**0.90**	0.88
Test 4 - OCSVM	DNN + AD2	**1.00**	0.60	0.94	**1.00**	0.94	0.80	0.97
Test 4 - iForest	DNN + AD2	**1.00**	0.80	0.97	**1.00**	**0.97**	**0.90**	**0.98**
Our Solution (Avg)	**DNN**	0.96	0.98	0.97	0.96	**0.97**	**0.97**	0.97
Our Solution (Best)	**DNN**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

TPR: True Positive Rate — **TNR:** True Negative Rate — **P:** Precision — **R:** Recall — **Acc.:** Accuracy — **BA:** Balanced Accuracy — **F1:** F1 Score. — P and F1 could not be calculated for solutions that predict that every sample is benign, as it would require a division by zero.

ences between these recordings and the ones in the original dataset.[4] In this scenario, four real users[5] provided commands to our solution through this device, by setting up different environmental and recording scenarios that differ from the ones in which the dataset was recorded. Finally, we assessed our solution's resilience under various settings.

We identified five variables that describe the conditions of the recordings:

1. **Background noise**: the intensity (in dB) of the background noise in the room while issuing the command.

2. **Voice**: the voice of the user that issues the benign command, or of the adversary when recording real-voice commands to be self-issued.

3. **Position of the user giving a command**: the physical location of the user within the room while issuing a command. We only take into account users who are standing still while talking, and they must be inside

[4] Devices of the same family may be built with different components within certain tolerance limits. However, note that if the device manufacturer decides to change the microphone array with another that is substantially different, our solution can be retrained by the device manufacturer to preserve optimal performance.

[5] The user who recorded the dataset and three additional users.

the room where the device is placed. This only applies to benign commands as there is no user near the device during the exploitation of the voice command self-issue.

4. **Volume of the device**: the current volume of the device while the command is being issued.

5. **Position of the device**: the physical location in which the device was placed when recording the command (e.g., on a table in a certain room).

To avoid using samples that the neural network already knew, we repeated the process of generating new samples, that is, we identified songs and podcasts to be played for benign samples while users were issuing commands, and we generated further malicious samples in which a voice command is embedded. In the end, we performed the following tests:

- **Test 1 — Variation of background noise**: we increased the background noise in the room from 20 dB (as when the dataset was recorded) to 40 dB. Commands were issued by the original user (the one who recorded the dataset). This test applies to both benign and malicious samples.

- **Test 2 — New users issuing commands**: to generate benign samples, we instructed three new users to stand in the same position that was used to record the original dataset and to issue a command while an audio file was being played. To generate malicious samples, we asked the same three users to send us some audio files featuring them issuing a command, so we could use them to generate a command self-issue. An advantage of this approach is that we can test our solution with audio files recorded from different devices, as every user would record them with different hardware (e.g., a smartphone). However, this test cannot be executed with malicious TTS samples as it would not involve different real users.

- **Test 3 — Different position of the user**: the original user was asked to issue a command when standing in a different position than the one used to record the dataset. This time they were standing at a 15 cm of distance from the device. This test applies to benign samples only.

- **Test 4 — Different volume of the device**: we changed the playback volume of the device to 3 or 7. Commands were issued by the original user. This test applies to both benign and malicious samples.

- **Test 5 — Different position of the device**: the device was moved into another room while keeping all the other original conditions (i.e., 20 dB background noise, original user issuing a command while standing at 60 cm from the device, playback volume 5). For malicious samples, we reproduced the malicious audio file with the device in the new room. Hence, this test applies to benign and malicious samples.

- **Test 6 — Combinations of the above tests**: the device was moved into another room, having a 40 dB background noise, and three different users were asked to issue commands during an audio playback at volume 3 while standing at 2.5 m from the device. We also reproduced malicious audio files in these new conditions. Hence, this test applies to both benign and malicious samples.

- **Test 7 — Commands with synthesised voices from external speakers**: we performed this set of tests to assess our solution's performance against synthesised voice commands issued by external speakers. In a real scenario, this represents a legitimate user trying to issue a command to a voice-controllable device via their SGD. The audio files containing voice commands were played from a speaker that was at a 50 cm of distance from the recording device during the first half of the tests, and that was at a 90 cm of distance for the second half of the tests. There was 40 dB of background noise in the room and the playback volume of the device was set at 3. Recall that, as our solution implements the Level

1 security of our taxonomy, malicious external synthesised commands are *out-of-scope*. Hence, this test applies to benign samples only.

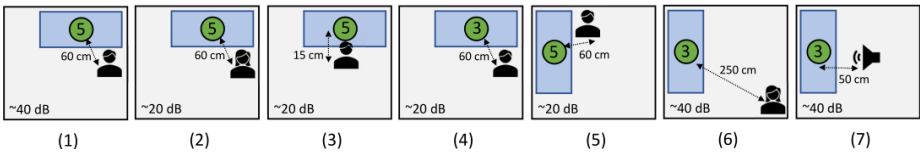

FIGURE 7.6: *Visual Representation of the Test Scenarios*
Images are not in scale. The number in the bottom-left corner indicates the background noise in the room. The circle indicates the device used to reproduce the *played* audio files and to create the *recorded* audio files. The number within the circle is the playback volume. The rectangle is a surface (a desk, a small cupboard) that has different objects on it, including the device. The icons of the male/female users only indicate that there is a change of user and they do not necessarily represent the actual genders of the people involved in that experiment. For simplicity, this image features only one setting for each test, even if the test has multiple scenarios (e.g., Test 4 is actually performed with 2 different volumes).

Figure 7.6 provides a visual representation of the conditions described above for the different scenarios. For these tests, we used an instance of our solution that correctly classified all of the samples in the testing dataset. However, as our model was never tested under these new conditions, a drop in the success rate was expected. For each test, we had 10 benign samples (4 featuring music, 3 podcasts, and 3 white noise) and 10 malicious ones (5 featuring TTS voice commands and 5 real-voice commands). As in Test 2 we cannot use malicious TTS voice commands, we only have 15 samples for that test. Additionally, as in Tests 3 and 7 there are no malicious samples, we only have 10 samples.

Table 7.5 summarises the obtained results. We can see that our solution still correctly classifies 90% of the benign samples on average: this indicates that, independently from the identity of the user speaking, our solution is able to reliably classify benign commands in all the tested environments. This also means that the user does not need to do an enrolment procedure if they want to activate this countermeasure, that is, they do not need to issue some commands to show the model the conditions in which it will operate.

Regarding the malicious samples, we can see that the feature that mostly affects a correct classification is the volume change: when all conditions were

TABLE 7.5: *Real Use-Case Tests Results*

Test	Ben.	Mal.	TPR	TNR	P	R	Acc.	BA	F1
0 - Test Dataset	✓	✓	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1 - Noise +	✓	✓	0.70	0.90	0.88	0.70	0.80	0.80	0.78
2 - User Change	✓	!	0.60	0.90	0.75	0.60	0.80	0.75	0.67
3 - User Pos.	✓	✗	-	0.90	-	-	0.90	-	-
4.1 - Volume -	✓	✓	0.40	0.90	0.80	0.40	0.65	0.65	0.53
4.2 - Volume +	✓	✓	0.40	0.90	0.80	0.40	0.65	0.65	0.53
5 - Device Pos.	✓	✓	0.70	0.90	0.88	0.70	0.80	0.80	0.78
6 - Combination	✓	✓	0.60	0.90	0.86	0.60	0.75	0.75	0.71
7 - Synth Voice	✓	✗	-	0.90	-	-	0.90	-	-

Ben.: Test applicable to benign samples. — **Mal.:** Test applicable to malicious samples. — ! in Test 2 means that the change of user is only feasible when using real-voice commands for the self-issue. — As there are no malicious samples for Tests 3 and 7, TPR, P, R, BA, and F1 are not calculated.

unaltered and the playback volume changed (Tests 4.1 and 4.2), our solution correctly classified only 40% of the malicious samples. However, we see that 60% of the malicious samples are correctly classified in Test 6. This is most likely because, even if the playback volume was set as 3, the users were also issuing commands from a farther distance than the one used to record the dataset — our intuition is that this allows the playback/user voice volume ratio to be closer to the one of the testing dataset. Hence, we hypothesise that with more extreme changes to the environmental conditions (e.g., user way too far or close to the VCD, volume of the VCD way too high or low, very high background noise such as 80 dB or more), a further drop in success rates is to be expected.

7.6.1 Performance Analysis

To evaluate the feasibility of running our solution on different scenarios, we analysed its performance while running on three different devices:

- **Smart Speaker (Raspberry Pi 4 Model B)**: this evaluates the implementation of our solution on a real smart speaker, as the performance of these devices is comparable to the Raspberry Pi's one. In fact, 3rd generation Echo Dot devices run a Mediatek MT8516BAA processor, a slightly

less performant processor than the Broadcom BCM2711 processor embedded on Raspberry Pi 4 Model B. Conversely, 4th generation Echo Dot devices run a Mediatek MT8512BAAV with an AZ1 neural edge processor, which is slightly better than the Raspberry's processor, especially when it comes to machine learning tasks. Hence, Raspberry offers a fair average of these devices' performance when running our solution. Our Raspberry runs Raspbian as OS and has 4 GB RAM available.

- **Laptop (ASUS X580VD)**: this evaluates the performance of our solution when running on a laptop. Our device runs an Intel Core i7-7700HQ, 16 GB RAM, and Windows 10 Home 64-bit as the operating system.

- **Cloud Server (Google Colab)**: this evaluates the performance of our solution when running on the cloud. We used a virtual machine on Google Colab, with an Intel Xeon processor, 26 GB RAM available, and the Linux Kernel 5.10.133.

Overhead Measurement

To ensure a fair comparison, we ran our solution using the devices' CPUs. Only on Google Colab, we ran a test using CPU and another using GPU, to check the magnitude of the improvement. Table 7.6 shows the average time taken by each device to execute the core functions of our solution.

TABLE 7.6: *Overhead Measurement on Different Devices*

Task	**Rasp**	**Laptop**	**Cloud CPU**	**Cloud GPU**
Network Initialisation	0.04s	**0.01s**	0.05s	2.08s
Sample Preprocessing	0.56s	**0.07s**	0.15s	0.10s
Category Prediction	3.40s	3.52s	0.46s*	**0.15s***

*: The first prediction on both Cloud CPU and GPU takes longer: on CPU it takes 0.60s on average while on GPU it takes 1.18s on average.

The network initialisation function is executed *una tantum*, for example when the device is booting up. We see that, in the worst-case scenario, it only takes slightly more than two seconds, a negligible time during the device

startup. Conversely, preprocessing of the audio files must be executed every time a command is given to the device. We can see that, in the worst-case scenario, our solution takes half a second to do so, which is acceptable as this process happens unknowingly to the user: in fact, this operation is done after the user has issued a command (in the benign scenario, as there would be no user in the malicious one), and at this point, the user is just waiting for the device to execute it. Hence, what is perceived by the user is the time elapsed from their query to the answer of the device, which is the preprocessing time plus the prediction time.[6]

Regarding the measurement of the prediction time, the Raspberry Pi and the laptop take more than three seconds to output whether the command is benign or not. This time, added to the time taken for the preprocessing, makes almost four seconds of overhead before the user can hear their reply. On the cloud, the preprocessing is slightly slower than the one performed by the laptop, but predictions are sensibly faster: even when adding the preprocessing time, our solution has less than one second of total overhead (0.61s when using CPU and 0.25s with a GPU).

Memory Usage

We calculate the primary and secondary memory usage of our solution by analysing the used RAM during a prediction and the total space on secondary memory taken by our Python 3.10 installation on the aforementioned Windows laptop, after installing all requirements for our solution to run. Disk space used to store the trained model and the source code of our solution is negligible (less than 2 MB). We find that our solution takes up to 400 MB of RAM to perform all operations. Additionally, it uses 1.26 GB of secondary memory. While these requirements are easily satisfied by modern laptops, smartphones, and cloud servers, it is not the case for smart speakers: 3rd Generation Echo Dot devices are supported by only 2Gb RAM (approximately

[6] Plus the time taken by the VPA to retrieve the answer to the query, which is not evaluated here as it is not part of our solution and it constitutes a constant overhead from the device.

256 MB) and 8Gb Flash Memory (approximately 1 GB) [118]. Even if 4th Generation Echo Dot devices have more RAM, the secondary memory requirement is not satisfied [52]. Google Home Mini runs similar hardware [9].

Analysis of the Results

Overhead measurement results indicate that user devices could run our solution and deliver a prediction in approximately four seconds. Although this adds significant overhead to the received replies, it is not uncommon for smart speakers and voice-controllable devices in general to have some delay between a command and its reply, and users tend to perceive this behaviour as normal [56]. However, memory usage results clearly indicate that smart speakers cannot run our solution yet, due to lack of RAM, secondary memory, or both [118, 52, 9]. Note that to build and test our solution we used standard Python and its neural network libraries, i.e., not natively built to run on small devices, hence the primary and secondary memory usage can be improved further. Nonetheless, as the speech recognition already happens on the cloud, overhead measurements reported in Table 7.6 show that it is feasible, and recommended, to implement our solution in the cloud as well, alongside the speech recognition service: in this way, the voice-controllable device sends both the *played* and the *recorded* audio to the cloud server for analysis (instead of just the *recorded* one, as it happens now for speech recognition), and the command is executed only if classified as benign by our solution. This approach has the advantage of being deployable to existing devices with only a software update (RG6) and runs with minimal overhead (0.25s to 0.60s, depending on the used cloud solution). Similarly to speech recognition, an instance of our solution in the cloud would serve several VCDs.

7.7 Discussion

In this section, we assess the limitations, possible misuse, and ethics and privacy considerations for our solution.

7.7.1 Limitations

Our solution is a countermeasure against command self-issue attacks (Level 1 in our taxonomy), hence it does not address other voice spoofing attacks, as done by solutions based on liveness detection [113, 24, 59, 176, 166, 158] or Automatic Speaker Verification [98, 151, 103, 31] (Level 3).

We also do not consider adversarial attacks against deep learning, which could bypass our solution by carefully crafting adversarial noise commands to be self-issued. In fact, past research [93] has shown that creating adversarial noise samples against deep-learning based classification of audio files based on spectrograms can be done with two techniques: the Fast Gradient Sign Method (FGSM) [63] and the Basic Iterative Method [97]. The work by Koerich et al. [93] analyses adversarial perturbations generated with these techniques, and discover that the unnoticeable perturbations on the spectrograms are indeed noticeable to the human ear when the spectrograms are converted to sound. However, this is not a problem in our context, as we are already aware that perceptible audio files are the main limitation of self-activation attacks, hence the adversary wants to attack the target device when the victim is not around. Although these techniques are relatively simple to execute, the adversarial samples they generate were not designed to work *over-the-air*, hence, the *played audio* would accurately reflect them, but more perturbations would be added to the *recorded audio* if the adversary tries to use such adversarial noise samples in a real scenario, which could make the samples harmless. Hence, even if an adversary can theoretically execute an adversarial attack against our solution using BIM and FGSM, they would need to be modified to bypass our solution in a real-world scenario.

7.7.2 Possible Misuses

Self-issued commands are inherently malicious (and their detection is desirable), hence, no relevant misuses of our solution can be anticipated at this stage. A purely theoretical misuse scenario may see an adversary place a rogue device physically near the target voice-controllable device. The rogue device would capture the audio in the room through a microphone, and at the same time, it would stream it on the target device, e.g. via Bluetooth. Hence, any legitimate command given to the target device would be instantly played by the same device, making our solution detect it as malicious. This is clearly an edge-case scenario — however, as stated before, to allow the best usability-security compromise, the user should always be able to disable or enable security measures. Intuitively, misuse of our solution would only increase its false positive rate.

7.7.3 Ethics and Privacy Considerations

We analysed potential risks stemming from the involvement of people in our tests via the risk assessment form provided by our institution. As a result, no risks were identified for our participants. We also provided each participant with an information sheet explaining the purpose of the study, their rights (e.g., withdrawing their consent), and how their data would have been used, along with a consent form to explicitly confirm they understood everything and to express their will to participate. The information sheet also informed them about the nature of the tests, as well as how files were recorded, used, and deleted. All users explicitly granted their permission to record the audio commands. It is worth noting that we did not retain the audio files containing the voices of the users involved in these tests.

As far as regards privacy considerations during the design of the solution, we designed it to be able to only process, but not store nor share with unauthorised third parties, only data that is already used by the voice controllable

device or by the speech recognition cloud. When our solution is implemented on the cloud as suggested in Section 7.6.1, the cloud providers do not receive more information than they already do without our solution: in fact, the audio file currently being played on a voice-controllable device is always captured in the background when the user issues a command. As no other data is processed by our solution, it does not introduce any additional privacy risks. Note that the data flow of the device is controlled by its manufacturer, hence, outside of the proposed solution's control.

7.8 Data Availability

The source code of our solution and the dataset we used to perform the training and validation tasks can be found at: https://github.com/Vereos/Protecting-Against-Self-Issue. Datasets are stored with Git LFS, so they cannot be downloaded via the Github website. To download them, install Git LFS and then `git clone` the repository using your command line interface. The voice that is featured within the samples is the candidate's voice.

7.9 Related Work

We now illustrate some countermeasures that can be applied to voice-controllable devices to mitigate the self-activation threat regardless of the used payload.

7.9.1 Self-Generated Wake-Word Suppression

Similar to the process of web input sanitisation, VCDs should ignore commands coming from their own speakers. This could be done by implementing a wake-word detection system for the audio output by the VCD: if a wake-word is detected within the played audio, it is not considered valid and no

command is executed [3, 100]. This approach has several limitations: as it is not possible to apply hardware modifications to already sold devices, such mitigation should be implemented via software. While a software update that introduces this feature might be possible on some voice-controllable devices, smart speakers have very limited resources [51, 52] and the additional recogniser could exceed them [3, 100]. Additionally, even if something similar to the playback device [100] is implemented on the cloud to not burden the voice-controllable device, this would mean that the *totality* of streamed audio files would transit from the cloud (i.e., not just while the user is issuing a voice command), including those that are reproduced via Bluetooth. This could have some privacy implications related to what the user is listening to and to their willingness to share it with the VPA service provider.

Alternatively, directional audio signals might be analysed, to check whether the wake-word comes from single or multiple directions, the latter indicating that the wake-word is being self-issued [131]. It is possible that Echo Dot already implements a similar technology, as an array consisting of four microphones can be found within the device [51], however, we have not been able to confirm this.

7.9.2 Liveness Detection

Countermeasures in this area detect whether the given command has been uttered by a real user or has been artificially created, e.g., a spoofed command coming from a nearby speaker. Meng et al. [113] propose ArrayID, a lightweight passive liveness detection solution that leverages the *array fingerprint*, which is based upon the design of circular microphone arrays already embedded on smart speakers, achieving good results even with environmental changes or with user movement. Blue et al. [24] found that soundwaves generated by electronic speakers differ from those generated by organic speakers, as the former have more energy in the sub-bass region (20-60 Hz), and use spectrum analysis to classify commands as invalid if too

much energy is detected in that region. Similarly, Void [4] is a lightweight solution that uses only 97 features to analyse spectrum power in voice commands to determine whether they are spoofed or not. Feng et al. [59] present VAuth, a system that provides continuous speaker authentication by means of a wearable device that features an accelerometer and an embedded VPA. When the user says a command, VAuth tries to authenticate the user via a previously-trained SVM. VoiceGesture [176] is a liveness detection system that leverages articulatory gesture detection, that is, detection of the articulatory movements that human beings perform when producing speech sound, for example, lip closure and jaw angle, and classifies a command to be valid if extracted features from the doppler shifts can match the pronounced phonemes. CaField [166] achieves a 99% success rate in detecting spoofed audios by training a Gaussian Mixture Model to recognise fieldprints, constructed using the acoustic biometrics embedded in sound fields. It is also possible to make a distinction between the airflow generated by a real user from the one generated by an electronic speaker if an airflow sensor is embedded within a device [158].

Recently, the work by Ahmed et al. [5] showed how some Liveness Detection techniques, such as Void [4], can be bypassed if the malicious actor speaks through a tube, raising concerning questions on whether other solutions might be vulnerable to similar simple attacks.

7.9.3 Automatic Speaker Verification

Works in this field detect if the given command comes from the user they have learned to recognise. ASVspoof [128, 165] is a competition to detect spoofed audios, where researchers can submit ASV solutions to be evaluated on a fixed dataset. One of the best solutions in ASVspoof 2019 was ASSERT [98], which leverages Squeeze and Excitation Networks and Residual Networks, while the best solution in the 2021 edition was developed by Tomilov et al. [151] and makes use of Residual Networks as well, in combination with a 9-layer Light

CNN (LCNN9) and RawNet2. Other recent work leverages Gaussian Mixture Models trained on genuine speech to output log-probabilities fed to a 1-D Convolutional Network, or a Twin Network, to detect whether the given command is spoofed or not, with good results [103]. Chaiwongyen et al. [31] developed a solution that is also able to reliably detect replay attacks to the same ASV system by feeding Gammatone cepstral coefficientss (GTCCs) to a model based on Residual Networks.

7.10 Summary

In this chapter, we proposed a countermeasure against self-issued voice commands to voice-controllable devices, to balance the security and usability requirements. In the tests, our solution correctly classifies commands 97% of the times on average, that is, it reliably predicts if the given commands are self-issued or if they come from a real user. We compared our results with state-of-the-art anomaly detection techniques and we showed that our solution outperforms them. Additionally, we deployed and measured the performance of our solution on three different classes of devices, showing that the overhead is negligible. Finally, we performed a test with four real users interacting with a device running our solution, showing that it is resilient to environmental changes such as the identity of the user issuing a command or the room in which the device is placed.

Chapter 8

Conclusion

This thesis has shown that self-activation poses a significant threat to users and voice-controllable devices, addressing RQ1 presented in Section 1.2.2, and that deep learning can be used to mitigate this risk. The PhD study began by formalising self-activation attacks, which are attacks where an adversary generates, sends, and executes self-issued commands, gaining persistence on the target device. The VOCODES Framework was developed, consisting of a tailored kill chain covering the entire self-activation procedure, and of a threat model that modelled the capabilities and knowledge of both the attacker and victim.

Next, the PhD study introduced the Alexa versus Alexa attack, which exploits a self-issue vulnerability found in 3rd and 4th generation Echo Dot devices to give them abritrary commands. This answers RQ2. Using AvA and exploiting two other vulnerabilities, the adversary can gain complete control of the target device, allowing them to (self-)issue arbitrary commands, eavesdrop on the user, tamper with replies, and more. Therefore, the impact of the attack is critical, and this addresses RQ3. The study also evaluated AvA in the real world with the help of 18 Echo users who completed a survey and three households who took part in a field study. While limitations exist for the AvA attack, they are mostly theoretical.

Finally, the PhD study analysed existing countermeasures against self-

activation attacks and found that the literature lacks specific protections against this type of attack. The few countermeasures that exist are either not yet implemented or not effective enough, hence answering RQ4. To address this gap and RQ5, the PhD study proposes a specific countermeasure against self-issued commands and a security taxonomy that enables users to select the desired level of security for their VCD. This solution allows disabled individuals who use synthesised voices to continue operating their VCDs, while providing a higher level of security against self-activation attacks. The study evaluated the proposed countermeasure against state-of-the-art anomaly detection techniques and found that it outperformed them, achieving 97% accuracy on average in classifying commands into benign and malicious.

In the next sections, we discuss about the key steps, the technical difficulties and the implications of the contributions made and of the technologies we decided to use.

8.1 Discussion

Our research on self-activation began in 2020, with the aim of investigating the feasibility of self-issued voice commands to Amazon Echo devices. As the self-activation attack leverages the victim device to play an audio file containing a voice command, which will be executed by the device itself, this attack allowed us to eliminate the need for an unauthorised speaker in proximity to the target device — a constraint shared by most attacks against smart speakers — and we were able to broaden the scope of potential attacks. To verify that such attacks were possible, we undertook two key steps: (i) finding an initial foothold to play the audio files and (ii) assessing the optimal way to generate voice commands to be self-issued.

In particular, for (i) we found it necessary to investigate multiple potential methods for playing audio files on the Amazon Echo device to determine which ones could be used for the self-activation attack. This process involved

a significant amount of effort, including searching for ways to play audio files and implementing them. One of the most challenging aspects was developing a Music & Radio skill, as this type of skill was not available in the countries where we conducted our research (i.e., the UK and Italy) and required multiple workarounds to set up. Ultimately, we discovered that not all methods for playing audio files were suitable for self-issuing commands. For instance, we found that playing audio files using an SSML audio tag would cause the device to stop audio playback when it heard the wake-word, which would interrupt the command self-issue. However, we confirmed that this issue did not occur with other methods for playing audio files, and were, therefore, able to establish the feasibility of self-activation on Amazon Echo.

Point (ii) also required a significant amount of effort, as we conducted several experiments to understand how an attacker could consistently carry out the self-activation attack. We found that the position of the Echo device within a room had a significant impact on how voice commands were self-issued due to soundwave reflections. To account for this, we identified three general scenarios: Open (with no obstacles near the Echo device resulting in fewer soundwave reflections), Wall (with only one obstacle near the Echo device), and Small (with multiple obstacles around the Echo device). We assessed the performance of ten different Wavenet voice profiles in each scenario to determine optimal conditions for the attack. We also evaluated the performance of the attack in specific conditions, such as when the device was set to a very high or very low volume, when the user was already listening to music, when the adversary kept using the same voice command, and while exploiting other vulnerabilities we had discovered, such as the Full Volume Vulnerability. These experiments were critical to the success of the attack and played a significant role in the media coverage that our work received worldwide. The self-activation attack raises serious concerns about the safety, security, and privacy of users: NIST rated the resulting AvA attack as “Critical”, giving it a 9.8 CVSS score out of 10. This was a significant upgrade from the

initial “Medium” severity that Amazon had assigned when we first reported the vulnerabilities, which had an average CVSS score of 5.5.

As we looked for ways to mitigate AvA, we discovered that there were existing countermeasures, including patents, against self-activation [3, 131, 100]. However, we observed that devices were still susceptible to self-issue attacks. We hypothesised that either these mitigations had not been implemented yet or were not as effective as advertised. Our brainstorming led us to the idea that similar *recorded* and *played* audio files could mean that a command was being self-issued, but the implementation of this idea was challenging. We attempted to use different technologies, for example, audio fingerprinting, which turned out to be ineffective, sometimes after weeks of study. After deciding to use Twin Networks, we did not find any suitable dataset consisting of *played* and *recorded* voice commands. Therefore, we set up our recording device with Seeed Respeaker and Raspberry Pi to create one. Finding the correct structure and hyperparameters for our network was challenging: when we achieved 100% accuracy on the validation dataset for the first time, although we knew we were on the right track, the performance of our network was still strongly influenced by the random initial values of weights and biases — additionally, a “bad” training epoch could easily undermine the 20-30 subsequent ones, potentially leading to “unlucky” scenarios in which the network could not be trained effectively. Fine-tuning the hyperparameters to obtain a stable training that did not overfit the dataset was a long and meticulous process. However, it was a crucial step in the network design, as it led us to a training process that, under our observations, always produced a network that has at least 94% accuracy in the benign/malicious classification of audio samples, has 97% accuracy on average and is able to hit 100% accuracy.

8.2 Implications of Contributions

Summarising the contributions described in Section 1.3, this thesis achieved the goal of:

(i) formally modelling,

(ii) practically implementing,

(iii) efficiently mitigating

self-activation attacks, which were mostly overlooked.

In regards to (i), the VOCODES Framework consisting of the kill chain and threat model provides a formal procedure and standard scenario for self-activation attacks. Other researchers can use the framework to compare their work with ours. For example, they can repeat the steps of the VOCODES Kill Chain on another smart speaker and compare their results with AvA, or they can use the scenarios described in the VOCODES Threat Model as a starting point for their own research on self-activation attacks.

Regarding (ii), our experiments demonstrated that self-issue attacks cannot be overlooked as they pose a critical threat to the security, safety, and privacy of legitimate users. Data collected from our experiments can serve as a baseline to assess the security of other devices against self-activation attacks, such as Google Nest, Apple Homepod, and Macbook Air. Additionally, repeating the user study and survey in the future can help understand whether user behavior toward this technology has changed. For example, more users may choose to mute their smart speakers' microphones during the night in the future. Through our work, we also raised awareness of self-activation attacks among users through media coverage, an informative website,[1] and a demonstrative video that has garnered over 21,000 views as of April 2023.[2] Finally, our efforts had a tangible impact on the security of the Echo device as Amazon deployed fixes to mitigate our attack.

[1] `https://www.ava-attack.org/#media`
[2] `https://www.youtube.com/watch?v=t-203SV_Eg8`

In relation to (iii), we conducted a thorough review of existing literature on mitigating voice-spoofing attacks and discovered that while liveness detection and ASV are effective against self-activation, no research addresses the needs of disabled users who require an artificial voice to communicate. This is a crucial opportunity to address and resolve this issue, especially as countermeasures against voice-spoofing attacks are not yet widely implemented on commercial devices, providing device manufacturers with ample time to effectively plan for the introduction of security measures that balance security and usability requirements, without hindering the ability of disabled individuals to interact with their voice-controlled devices. Our solution against self-activation serves as the initial milestone toward more secure voice-controlled devices while retaining their usability. The source code and performance data of our solution can be used as a baseline for future research endeavors aimed at defending against self-issued commands. Furthermore, the dataset we released is a critical milestone as other researchers can train or test their solutions against self-issued commands, even if they were developed to resolve general voice-spoofing command problems.

The methodological implications of our research are closely tied to the limitations of the methods and technologies we used. In regards to the VOCODES Kill Chain, we initially recognised the limitations of the Lockheed-Martin Kill Chain [77], including its lack of depth and cyclicality. We then analysed other kill chains, including the detailed Unified Kill Chain [132], which is currently considered the state-of-the-art in addressing these limitations. It may seem counterintuitive that we chose to build our VOCODES Kill Chain on the same Lockheed-Martin Kill Chain that other works attempted to improve. However, we emphasise that the six steps outlined in the VOCODES Kill Chain were sufficient to describe in detail the various phases of a self-activation attack, to the point where additional declinations and sub-steps may be redundant. Additionally, the VOCODES Kill Chain does not suffer from the depth problem, as self-activation targets are typically not layered,

so the external/internal attack distinction is unnecessary. We also addressed the cyclicality issue within the Lockheed-Martin Kill Chain by clearly outlining the links from one phase to another, including previous ones. That being said, the VOCODES Kill Chain may not be the most suitable option for attacks that include self-issued commands but have a broader infrastructure scope (which may only happen to include a voice-controllable device). For these scenarios, a more fine-grained kill chain could be useful to analyse the attack, such as linking the Unified Kill Chain to the VOCODES Kill Chain or developing a new kill chain.

Regarding AvA, our research identified two main limitations in the methods and technologies used. Firstly, the development of Music & Radio skills was only possible within the US, which limits the generalisability of their use to other regions. Addressing this limitation would require the manufacturer (Amazon) to provide a way to access this functionality outside of the US. Secondly, there was a lack of automated processes to perform the audio weaponization task. However, this limitation can be easily addressed by implementing a tool that creates many weaponized audio files with varying combinations of voice characteristics. It is worth noting that while our work focused on the Echo device, other voice-controllable devices such as computers or smart speakers may have different methods for playing audio files.

Regarding our solution against self-activation, we utilised twin networks, which are the state-of-the-art for comparing inputs with deep learning. Because no public dataset featuring both *played* and *recorded* audio files existed, we had to build a recording device with the Seeed Respeaker array while using an Echo device to play audio. To improve the dataset, an optimal way would be to use a smart speaker that communicates with a computer to capture both the *played* and *recorded* audio in real-time. This approach would provide samples coming from the actual hardware of the real device. Laptops are also voice-controllable devices and can be used to extend the dataset, but separate datasets may be necessary for each class of voice-controllable de-

vice, given the differences in shape and design.

8.3 Future Work

Building upon the methodological implications discussed earlier, this section suggests areas that could be explored in future research.

For AvA, we propose exploring the reliability of self-issued adversarial commands that are semantically meaningful, such as the adversarial samples proposed by Yu et al. [170], as they could significantly increase the stealthiness of the attack. This could open up new attack scenarios where the user is in close proximity of the device during the self-issued command. Furthermore, other smart speakers and voice-controllable devices might be vulnerable to self-issue attacks. Investigating such vulnerabilities would aid in generalising the optimal characteristics of the payload used for self-activation attacks, which can be used for ethical purposes to develop effective countermeasures that consider these characteristics.

Furthermore, although our proposed Level 1 countermeasure is reliable, having a fully-operational Level 2 countermeasure would offer more options to the user and ensure a better balance between usability and security. However, a Level 2 countermeasure would still be susceptible to targeted attacks if the synthesised voice used by the legitimate user is publicly available online. Embedding an audio fingerprint in the authorised artificial speaker that emits the artificial voice on behalf of the user can help establish whether the synthesised voice is coming from an authorised speaker or not, but it may still be vulnerable to replay attacks. To mitigate this, a combination of audio fingerprints and nonces could be used to verify if the command is coming from an authorised speaker and if it is fresh enough. WakeGuard [108], a solution against voice spoofing attacks that leverages audio watermarks to identify nearby speakers, could be a good starting point to implement a robust authentication system for external speakers.

Glossary

AAC	Augmentative and Alternative Communication
AD	Anomaly Detection
APT	Advanced Persistent Threat
ASR	Automatic Speech Recognition
ASV	Automatic Speaker Verification
ATT	MITRE ATT&CK®
AvA	Alexa versus Alexa
AVS	Alexa Voice Service
CKC	Cyber Kill Chain®
CNN	Convolutional Neural Network
CVSS	Common Vulnerability Scoring System
DNN	Deep Neural Network
DoS	Denial of Service
E2E	End-to-End
EKC	Expanded Kill Chain
FVV	Full Volume Vulnerability

GMM	Gaussian Mixture Model
GPT	Generative Pre-Trained Transformer
GTCC	Gammatone cepstral coefficients
HMM	Hidden Markov Model
IoT	Internet of Things
IoV	Internet of Vehicles
LCNN9	9-layer Light CNN
LSTM	Long Short-Term Memory
MKC	Modified Kill Chain
NLP	Natural Language Processing
NLU	Natural Language Understanding
OCSVM	One-Class Support Vector Machine
PII	Personal Identifiable Information
PZT	Piezoelectric Transducer
RG	Research Goal
RNN	Recurrent Neural Network
RNN-T	Recurrent Neural Network Transducer
RQ	Research Question
SGD	Speech Generating Device
SLU	Spoken Language Understanding
SSML	Speech Synthesis Markup Language

STT	Speech-To-Text
TNN	Twin Neural Network
TTS	Text-To-Speech
UKC	Unified Kill Chain
VCD	Voice-Controllable Device
VMA	Voice Masquerading Attack
VOCODES	VOice-COntrollable DEvice Self-Issue
VPA	Voice Personal Assistant
WAV	Waveform Audio File

Bibliography

[1] Noura Abdi, Kopo M. Ramokapane, and Jose M. Such. More than Smart Speakers: Security and Privacy Perceptions of Smart Home Personal Assistants. In *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*, pages 451–466, Santa Clara, CA, August 2019. USENIX Association. ISBN 978-1-939133-05-2. URL https://www.usenix.org/conference/soups2019/presentation/abdi.

[2] Hadi Abdullah, Kevin Warren, Vincent Bindschaedler, Nicolas Papernot, and Patrick Traynor. SoK: The Faults in our ASRs: An Overview of Attacks against Automatic Speech Recognition and Speaker Identification Systems. *CoRR*, abs/2007.06622, 2020. URL https://arxiv.org/abs/2007.06622.

[3] Erich Adams. Avoiding Wake-Word Self-Triggering. https://patents.google.com/patent/US20190311719A1/en, 2018. Accessed: 2020-12-04.

[4] Muhammad Ejaz Ahmed, Il-Youp Kwak, Jun Ho Huh, Iljoo Kim, Taekkyung Oh, and Hyoungshick Kim. Void: A fast and light voice liveness detection system. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2685–2702, 2020.

[5] Shimaa Ahmed, Yash Wani, Ali Shahin Shamsabadi, Mohammad Yaghini, Ilia Shumailov, Nicolas Papernot, and Kassem Fawaz. Tubes among us: Analog attack on automatic speaker identification. 2023.

[6] Efthimios Alepis and Constantinos Patsakis. Monkey Says, Monkey Does: Security and Privacy on Voice Assistants. *IEEE Access*, 5:17841–17851, 2017. doi: 10.1109/ACCESS.2017.2747626.

[7] Abdulaziz Alhadlaq, Jun Tang, Marwan Almaymoni, and Aleksandra Korolova. Privacy in the Amazon Alexa skills ecosystem. *Star*, 217(11), 1902.

[8] Norah Alotaibi, John Williamson, and Mohamed Khamis. Thermosecure: Investigating the effectiveness of ai-driven thermal attacks on commonly used computer keyboards. *ACM Transactions on Privacy and Security*, 26(2):1–24, 2023.

[9] Justin Alvey. Google Home Mini teardown, comparison to Echo Dot, and giving technology a voice. https://justlv.medium.com/google-home-mini-teardown-comparison-to-echo-dot-and-giving-technology-a-voice-c59a23724a26, 2017. Accessed: 2022-09-21.

[10] Amazon.com Inc. Github - Alexa Voice Service Client: Python Client for Alexa Voice Service (AVS). https://github.com/richtier/alexa-voice-service-client, 2017. Accessed: 2021-01-25.

[11] Amazon.com Inc. Request Customer Contact Information for Use in Your Skill. https://developer.amazon.com/en-US/docs/alexa/custom-skills/request-customer-contact-information-for-use-in-your-skill.html, 2019. Accessed: 2022-09-21.

[12] Amazon.com Inc. Speech Synthesis Markup Language (SSML) Reference - Alexa Skills Kit. https://developer.amazon.com/en-US/docs/alexa/custom-skills/speech-synthesis-markup-language-ssml-reference.html, 2019. Accessed: 2021-01-25.

[13] Amazon.com Inc. Request and Response JSON Reference - Alexa Skills Kit. https://developer.amazon.com/en-GB/docs/alexa/custom-skills/request-and-response-json-reference.html, 2020. Accessed: 2021-01-25.

[14] Amazon.com Inc. Amazon Echo & Alexa Devices. https://www.amazon.com/smart-home-devices/b?node=9818047011, 2022. Accessed: 2022-08-11.

[15] Deeksha Anniappa and Yoohwan Kim. Security and Privacy Issues with Virtual Private Voice Assistants. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0702–0708, 2021. doi: 10.1109/CCWC51732.2021.9375964.

[16] Wihem Arsac, Giampaolo Bella, Xavier Chantry, and Luca Compagna. Validating security protocols under the general attacker. *ARSPA-WITS*, 5511:34–51, 2009.

[17] Alexandru Baltag and Sonja Smets. Probabilistic Dynamic Belief Revision. *Synthese*, 165(2):179–202, 2008.

[18] Wilson Bautista. *Practical cyber intelligence: how action-based intelligence can be an effective response to incidents*. Packt Publishing Ltd, 2018.

[19] Jake L Beavers, Michael Faulks, and Jims Marchang. Hacking nhs pacemakers: a feasibility study. In *2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3)*, pages 206–212. IEEE, 2019.

[20] Giampaolo Bella, Rosario Giustolisi, and Carsten Schürmann. Modelling Human Threats in Security Ceremonies. *Journal of Computer Security*, 2022.

[21] Frank Bentley, Chris Luvogt, Max Silverman, Rushani Wirasinghe, Brooke White, and Danielle Lottridge. Understanding the long-term use of smart speaker assistants. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2 (3), sep 2018. doi: 10.1145/3264901. URL `https://doi.org/10.1145/3264901`.

[22] Battista Biggio and Fabio Roli. Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning. *Pattern Recognition*, 84:317–331, dec 2018. doi: 10.1016/j.patcog.2018.07.023.

[23] Mary K. Bispham, Ioannis Agrafiotis, and Michael Goldsmith. Nonsense Attacks on Google Assistant and Missense Attacks on Amazon Alexa. In Paolo Mori, Steven Furnell, and Olivier Camp, editors, *Proceedings of the 5th International Conference on Information Systems Security and Privacy, ICISSP 2019, Prague, Czech Republic, February 23-25, 2019*, pages 75–87. SciTePress, 2019. doi: 10.5220/0007309500750087.

[24] Logan Blue, Luis Vargas, and Patrick Traynor. Hello, Is It Me You're Looking For? Differentiating Between Human and Electronic Speakers for Voice Interface Security. In *Proceedings of the 11th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '18, page 123–133, New York, NY,

USA, 2018. Association for Computing Machinery. ISBN 9781450357319. doi: 10.1145/3212480.3212505.

[25] BMW (UK) Limited. BMW Online Genius - What is Intelligent Personal Assistant? `https://discover.bmw.co.uk/help/technology/what-is-ipa`, 2021. Accessed: 2022-12-05.

[26] Tero Bodström and Timo Hämäläinen. A novel method for detecting apt attacks by using ooda loop and black swan theory. In *Computational Data and Social Networks: 7th International Conference, CSoNet 2018, Shanghai, China, December 18–20, 2018, Proceedings 7*, pages 498–509. Springer, 2018.

[27] John R Boyd. The essence of winning and losing. *Unpublished lecture notes*, 12 (23):123–125, 1996.

[28] Michael Burrows, Martin Abadi, and Roger Needham. A Logic of Authentication. *ACM Transactions on Computer Systems (TOCS)*, 8(1):18–36, 1990.

[29] Nicholas Carlini and David A. Wagner. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. In *2018 IEEE Security and Privacy Workshops, SP Workshops 2018, San Francisco, CA, USA, May 24, 2018*, pages 1–7. IEEE Computer Society, 2018. doi: 10.1109/SPW.2018.00009.

[30] Sachin Chachada and C.-C. Jay Kuo. Environmental Sound Recognition: A Survey. *APSIPA Transactions on Signal and Information Processing*, 3:e14, 2014. doi: 10.1017/ATSIP.2014.12.

[31] Anuwat Chaiwongyen, Suradej Duangpummet, Jessada Karnjana, Waree Kongprawechnon, and Masashi Unoki. Replay Attack Detection in Automatic Speaker Verification Using Gammatone Cepstral Coefficients and ResNet-Based Model. *Journal of Signal Processing*, 26(6):171–175, 2022.

[32] Anuwat Chaiwongyen, Norranat Songsriboonsit, Suradej Duangpummet, Jessada Karnjana, Waree Kongprawechnon, and Masashi Unoki. Contribution of Timbre and Shimmer Features to Deepfake Speech Detection. In *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 97–103, 2022. doi: 10.23919/APSIPAASC55919.2022.9980281.

[33] Guangke Chen, Sen Chen, Lingling Fan, Xiaoning Du, Zhe Zhao, Fu Song, and Yang Liu. Who is Real Bob? Adversarial Attacks on Speaker Recognition Systems. *CoRR*, abs/1911.01840, 2019. URL http://arxiv.org/abs/1911.01840.

[34] Yuxuan Chen, Xuejing Yuan, Jiangshan Zhang, Yue Zhao, Shengzhi Zhang, Kai Chen, and XiaoFeng Wang. Devil's Whisper Docker Hub. https://hub.docker.com/repository/docker/neeze/devilwhisper, 2019. Accessed: 2021-01-25.

[35] Yuxuan Chen, Xuejing Yuan, Jiangshan Zhang, Yue Zhao, Shengzhi Zhang, Kai Chen, and XiaoFeng 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)*, pages 2667–2684. USENIX Association, August 2020. ISBN 978-1-939133-17-5. URL https://www.usenix.org/conference/usenixsecurity20/presentation/chen-yuxuan.

[36] Long Cheng, Christin Wilson, Song Liao, Jeffrey Young, Daniel Dong, and Hongxin Hu. Dangerous Skills Got Certified: Measuring the Trustworthiness of Skill Certification in Voice Personal Assistant Platforms. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 1699–1716, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370899. doi: 10.1145/3372297.3423339.

[37] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. A comparison of audio signal preprocessing methods for deep neural networks on music tagging. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1870–1874. IEEE, 2018.

[38] Vijay Choubey. Understanding Recurrent Neural Network (RNN) and Long Short Term Memory(LSTM), Jul 2020. URL https://medium.com/analytics-vidhya/undestanding-recurrent-neural-network-rnn-and-long-short-term-memory-lstm-30bc1221e80d.

[39] Christina L Corso. *The Impact of Smart Home Technology on Independence for Individuals Who Use Augmentative and Alternative Communication.* Ohio University, 2021.

[40] Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*, 2018.

[41] Marc Dacier and Yves Deswarte. Privilege graph: an extension to the typed access matrix model. In *Computer Security—ESORICS 94: Third European Symposium on Research in Computer Security Brighton, United Kingdom, November 7–9, 1994 Proceedings 3*, pages 319–334. Springer, 1994.

[42] Poorna Banerjee Dasgupta. Detection and Analysis of Human Emotions Through Voice and Speech Pattern Processing. *arXiv preprint arXiv:1710.10198*, 2017.

[43] Allan de Barcelos Silva, Marcio Miguel Gomes, Cristiano André da Costa, Rodrigo da Rosa Righi, Jorge Luis Victoria Barbosa, Gustavo Pessin, Geert De Doncker, and Gustavo Federizzi. Intelligent personal assistants: A systematic literature review. *Expert Systems with Applications*, 147:113193, 2020.

[44] Alberto De Campo. Toward a data sonification design space map. In *Proceedings of the International Conference on Auditory Display*, pages 342–347, 2007.

[45] Francien Dechesne and Yanjing Wang. To Know or Not to Know: Epistemic Approaches to Security Protocol Verification. *Synthese*, 177(1):51–76, 2010.

[46] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019.

[47] Sounak Dey, Anjan Dutta, J Ignacio Toledo, Suman K Ghosh, Josep Lladós, and Umapada Pal. Signet: Convolutional Siamese Network for Writer Independent Offline Signature Verification. *arXiv preprint arXiv:1707.02131*, 2017.

[48] Wenrui Diao, Xiangyu Liu, Zhe Zhou, and Kehuan Zhang. Your Voice Assistant is Mine: How to Abuse Speakers to Steal Information and Control Your Phone. In Cliff Wang, Dijiang Huang, Kapil Singh, and Zhenkai Liang, editors, *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile*

Devices, SPSM@CCS 2014, Scottsdale, AZ, USA, November 03 - 07, 2014, pages 63–74. ACM, 2014. doi: 10.1145/2666620.2666623.

[49] Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208, 1983.

[50] Chenhe Dong, Yinghui Li, Haifan Gong, Miaoxin Chen, Junxin Li, Ying Shen, and Min Yang. A Survey of Natural Language Generation. *ACM Comput. Surv.*, 55(8), dec 2022. ISSN 0360-0300. doi: 10.1145/3554727. URL https://doi.org/10.1145/3554727.

[51] Brian Dorey. Echo Dot 3rd Gen Smart Speaker Teardown. https://www.briandorey.com/post/echo-dot-3rd-gen-smart-speaker-teardown, 2019. Accessed: 2021-02-03.

[52] Brian Dorey. Amazon Echo Dot 4th Gen Smart Speaker Teardown. https://www.briandorey.com/post/echo-dot-4th-gen-smart-speaker-teardown, 2020. Accessed: 2022-09-21.

[53] Diego Droghini, Fabio Vesperini, Emanuele Principi, Stefano Squartini, and Francesco Piazza. Few-Shot Siamese Neural Networks Employing Audio Features for Human-Fall Detection. In *Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence*, PRAI 2018, page 63–69, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450364829. doi: 10.1145/3243250.3243268.

[54] Daniel J Dubois, Roman Kolcun, Anna Maria Mandalari, Muhammad Talha Paracha, David Choffnes, and Hamed Haddadi. When Speakers Are All Ears: Characterizing Misactivations of IoT Smart Speakers. *Proceedings on Privacy Enhancing Technologies*, 2020(4):255–276, 2020.

[55] Jide S. Edu, Jose M. Such, and Guillermo Suarez-Tangil. Smart Home Personal Assistants: A Security and Privacy Review. *ACM Comput. Surv.*, 53(6), dec 2020. ISSN 0360-0300. doi: 10.1145/3412383.

[56] Sergio Esposito, Daniele Sgandurra, and Giampaolo Bella. Alexa versus Alexa: Controlling Smart Speakers by Self-Issuing Voice Commands. In *Proceedings of*

the 2022 ACM on Asia Conference on Computer and Communications Security, pages 1064–1078, 2022.

[57] Sergio Esposito, Daniele Sgandurra, and Giampaolo Bella. Alexa vs Alexa (AvA). https://www.ava-attack.org/, 2022. Accessed: 2022-08-16.

[58] Ronald Fagin and Joseph Y. Halpern. Reasoning about Knowledge and Probability. *J. ACM*, 41(2):340–367, mar 1994. ISSN 0004-5411. doi: 10.1145/174652.174658.

[59] Huan Feng, Kassem Fawaz, and Kang G. Shin. Continuous Authentication for Voice Assistants. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, MobiCom '17, page 343–355, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349161. doi: 10.1145/3117811.3117823.

[60] Fortinet, Inc. Exploit Definition. https://www.fortinet.com/resources/cyberglossary/exploit, 2021. Accessed: 2023-08-30.

[61] Forum of Incident Response and Security Teams (FIRST). Common Vulnerability Scoring System v3.1: Specification Document. https://www.first.org/cvss/v3.1/specification-document, 2019. Accessed: 2023-08-30.

[62] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[63] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[64] Google LLC. Introduction to Audio Encoding - Cloud Speech-to-Text. https://cloud.google.com/speech-to-text/docs/encoding, 2013. Accessed: 2021-01-25.

[65] Google LLC. Compare the Google Nest family. https://store.google.com/gb/magazine/compare_speakers, 2022. Accessed: 2022-08-11.

[66] Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

[67] Louis Grenard. Leon - Your Open-Source Personal Assistant. https://getleon.ai/, 2019. Accessed: 2022-12-05.

[68] Houjian Guo, Chaoran Liu, Carlos Toshinori Ishi, and Hiroshi Ishiguro. QuickVC: Any-to-many Voice Conversion Using Inverse Short-time Fourier Transform for Faster Conversion, 2023.

[69] Harshvardhan Gupta. Facial Similarity with Siamese Networks in PyTorch. https://hackernoon.com/facial-similarity-with-siamese-networks-in-pytorch-9642aa9db2f7, 2017. Accessed: 2022-09-14.

[70] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality Reduction by Learning an Invariant Mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[71] Rosa González Hautamäki, Tomi Kinnunen, Ville Hautamäki, Timo Leino, and Anne-Maria Laukkanen. I-vectors meet imitators: on vulnerability of speaker verification systems against voice mimicry. In *Interspeech*, pages 930–934. Citeseer, 2013.

[72] Mary Hewitt and Hamish Cunningham. Taxonomic classification of iot smart home voice control. *arXiv preprint arXiv:2210.15656*, 2022.

[73] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[74] John D Howard and Thomas A Longstaff. A common language for computer security incidents. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States); Sandia . . . , 1998.

[75] Matthew Hoy. Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants. *Medical Reference Services Quarterly*, 37:81–88, 01 2018. doi: 10.1080/02763869.2018.1404391.

[76] Aaron Hunter, James P Delgrande, et al. Belief Change and Cryptographic Protocol Verification. In *Formal Models of Belief Change in Rational Agents*, 2007.

[77] Eric M Hutchins, Michael J Cloppert, Rohan M Amin, et al. Intelligence-driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80, 2011.

[78] IBM Corporation. Protect Data in Transit or at Rest. `https://www.ibm.com/cloud/architecture/architecture/practices/data-security/`, 2022. Accessed: 2023-08-30.

[79] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[80] Md Amirul Islam, Matthew Kowal, Sen Jia, Konstantinos G Derpanis, and Neil DB Bruce. Position, Padding and Predictions: A Deeper Look at Position Information in CNNs. *arXiv preprint arXiv:2101.12322*, 2021.

[81] Mahaveer Jain, Gil Keren, Jay Mahadeokar, Geoffrey Zweig, Florian Metze, and Yatharth Saraf. Contextual rnn-t for open domain asr, 2020.

[82] Yeongjin Jang, Chengyu Song, Simon P. Chung, Tielei Wang, and Wenke Lee. A11y Attacks: Exploiting Accessibility in Operating Systems. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, page 103–115, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450329576. doi: 10.1145/2660267.2660295.

[83] Shaun K. Kane, Meredith Ringel Morris, Ann Paradiso, and Jon Campbell. "At times avuncular and cantankerous, with the reflexes of a mongoose": Understanding Self-Expression through Augmentative and Alternative Communication Devices. *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, 2017.

[84] Takuhiro Kaneko and Hirokazu Kameoka. CycleGAN-VC: Non-parallel Voice Conversion Using Cycle-Consistent Adversarial Networks. In *2018 26th Eu-*

ropean Signal Processing Conference (EUSIPCO), pages 2100–2104, 2018. doi: 10.23919/EUSIPCO.2018.8553236.

[85] Takuhiro Kaneko, Hirokazu Kameoka, Kou Tanaka, and Nobukatsu Hojo. Cyclegan-VC2: Improved Cyclegan-based Non-parallel Voice Conversion. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6820–6824, 2019. doi: 10.1109/ICASSP.2019.8682897.

[86] Sulaiman M Karim, Adib Habbal, Shehzad Ashraf Chaudhry, Azeem Irshad, et al. Architecture, protocols, and security in iov: Taxonomy, analysis, challenges, and solutions. *Security and Communication Networks*, 2022, 2022.

[87] Sean Kennedy, Haipeng Li, Chenggang Wang, Hao Liu, Boyang Wang, and Wenhai Sun. I Can Hear Your Alexa: Voice Command Fingerprinting on Smart Home Speakers. In *2019 IEEE Conference on Communications and Network Security (CNS)*, pages 232–240, 2019. doi: 10.1109/CNS.2019.8802686.

[88] Awais Khan, Khalid Mahmood Malik, James Ryan, and Mikul Saravanan. Voice spoofing countermeasures: Taxonomy, state-of-the-art, experimental analysis of generalizability, open challenges, and the way forward. *arXiv preprint arXiv:2210.00417*, 2022.

[89] Hyeob Kim, HyukJun Kwon, and Kyung Kyu Kim. Modified Cyber Kill Chain Model for Multimedia Service Environments. *Multimedia Tools and Applications*, 78(3):3153–3170, 2019.

[90] Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, and Ye-Yi Wang. Intent detection using semantically enriched word embeddings. In *2016 IEEE spoken language technology workshop (SLT)*, pages 414–419. IEEE, 2016.

[91] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[92] Robin Knote, Andreas Janson, Matthias Söllner, and Jan Marco Leimeister. Classifying smart personal assistants: An empirical cluster analysis. 2019.

[93] Karl Michel Koerich, Mohammad Esmailpour, Sajjad Abdoli, Alceu de S Britto, and Alessandro L Koerich. Cross-representation transferability of adversarial attacks: From spectrograms to audio waveforms. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

[94] 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)*, pages 1962–1966, 2018. doi: 10.1109/ICASSP.2018.8462693.

[95] Deepak Kumar, Riccardo Paccagnella, Paul Murley, Eric Hennenfent, Joshua Mason, Adam Bates, and Michael Bailey. Skill Squatting Attacks on Amazon Alexa. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 33–47, Baltimore, MD, August 2018. USENIX Association. ISBN 978-1-939133-04-5. URL `https://www.usenix.org/conference/usenixsecurity18/presentation/kumar`.

[96] Deepak Kumar, Kelly Shen, Benton Case, Deepali Garg, Galina Alperovich, Dmitry Kuznetsov, Rajarshi Gupta, and Zakir Durumeric. All things considered: An analysis of IoT devices on home networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1169–1185, Santa Clara, CA, August 2019. USENIX Association. ISBN 978-1-939133-06-9. URL `https://www.usenix.org/conference/usenixsecurity19/presentation/kumar-deepak`.

[97] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

[98] Cheng-I Lai, Nanxin Chen, Jesús Villalba, and Najim Dehak. ASSERT: Anti-Spoofing with Squeeze-Excitation and Residual Networks. In Gernot Kubin and Zdravko Kacic, editors, *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 1013–1017. ISCA, 2019. doi: 10.21437/Interspeech.2019-1794.

[99] Harjinder Singh Lallie, Kurt Debattista, and Jay Bal. A review of attack graph and attack tree visual syntax in cyber security. *Computer Science Review*, 35:100219, 2020.

[100] Jonathan P. Lang. Wake-Word Detection Suppression. https://patents.google.com/patent/US10475449B2/en, 2017. Accessed: 2020-12-04.

[101] Yann LeCun, Koray Kavukcuoglu, and Clement Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256, 2010. doi: 10.1109/ISCAS.2010.5537907.

[102] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436–444, 2015.

[103] Zhenchun Lei, Yingen Yang, Changhong Liu, and Jihua Ye. Siamese Convolutional Neural Network Using Gaussian Probability Feature for Spoofing Speech Detection. In *INTERSPEECH*, pages 1116–1120, 2020.

[104] Christopher Lentzsch, Sheel Jayesh Shah, Benjamin Andow, Martin Degeling, Anupam Das, and William Enck. Hey Alexa, is this Skill Safe?: Taking a Closer Look at the Alexa Skill Ecosystem. In *Proceedings of the 28th ISOC Annual Network and Distributed Systems Symposium (NDSS)*, 2021.

[105] Jinyu Li et al. Recent advances in end-to-end automatic speech recognition. *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.

[106] Juncheng Li, Shuhui Qu, Xinjian Li, Joseph Szurley, J. Zico Kolter, and Florian Metze. Adversarial Music: Real world Audio Adversary against Wake-word Detection System. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11908–11918, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/ebbdfea212e3a756a1fded7b35578525-Abstract.html.

[107] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation Forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[108] Israel J Lopez-Toledo. *Wake guard: Protecting smart speakers' wake words with audio watermarking*. PhD thesis, 2022.

[109] Loren Lugosch. Sequence-to-sequence learning with Transducers, Nov 2020. URL https://lorenlugosch.github.io/posts/2020/11/transducer/.

[110] Sean Malone. The Expanded Cyber Kill Chain Model. https://www.seantmalone.com/docs/us-16-Malone-Using-an-Expanded-Cyber-Kill-Chain-Model-to-Increase-Attack-Resiliency.pdf, 2016.

[111] Brian Margolis, Madhav Ghei, and Bryan Pardo. Applying Triplet Loss to Siamese-Style Networks for Audio Similarity Ranking. In *DCASE*, pages 128–132, 2018.

[112] Johnny Mariéthoz and Samy Bengio. Can a professional imitator fool a GMM-based speaker verification system? Technical report, IDIAP, 2005.

[113] Yan Meng, Jiachun Li, Matthew Pillari, Arjun Deopujari, Liam Brennan, Hafsah Shamsie, Haojin Zhu, and Yuan Tian. Your Microphone Array Retains Your Identity: A Robust Voice Liveness Detection System for Smart Speakers. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1077–1094, Boston, MA, August 2022. USENIX Association. ISBN 978-1-939133-31-1. URL https://www.usenix.org/conference/usenixsecurity22/presentation/meng.

[114] Abraham Mhaidli, Manikandan Kandadai Venkatesh, Yixin Zou, and Florian Schaub. Listen Only When Spoken To: Interpersonal Communication Cues as Smart Speaker Privacy Controls. *Proceedings on Privacy Enhancing Technologies*, 2020(2):251–270, 2020.

[115] Microsoft Corporation. The STRIDE Threat Model, 2009. URL https://learn.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20).

[116] Microsoft Corporation. Text to Speech - Realistic AI Voice Generator — Microsoft Azure. https://azure.microsoft.com/en-us/products/cognitive-services/text-to-speech/, 2022. Accessed: 2022-12-06.

[117] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[118] Robin Mitchell. Teardown - Components Breakdown of an Amazon Echo Dot. https://www.electropages.com/blog/2020/10/teardown-amazon-echo-dot, 2020. Accessed: 2022-09-14.

[119] Tom M Mitchell. Machine learning, 1997.

[120] Richard Mitev, Markus Miettinen, and Ahmad-Reza Sadeghi. Alexa Lied to Me: Skill-Based Man-in-the-Middle Attacks on Virtual Assistants. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, Asia CCS '19, page 465–478, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367523. doi: 10.1145/3321705.3329842.

[121] MITRE Corporation. Common Attack Pattern Enumeration and Classification, 2007. URL https://capec.mitre.org/about/index.html.

[122] Robert Müller, Fabian Ritz, Steffen Illium, and Claudia Linnhoff-Popien. Acoustic Anomaly Detection for Machine Sounds Based on Image Transfer Learning. *arXiv preprint arXiv:2006.03429*, 2020.

[123] Mycroft AI, Inc. Mark II - Mycroft. https://mycroft.ai/product/mark-ii/, 2021. Accessed: 2022-12-05.

[124] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551, 2011.

[125] Vivek Nair, Gonzalo Munilla Garrido, and Dawn Song. Exploring the unprecedented privacy risks of the metaverse. *arXiv preprint arXiv:2207.13176*, 2022.

[126] National Institute of Standards and Technology. NIST Special Publication 1800-25A. https://www.nccoe.nist.gov/publication/1800-25/VolA/index.html, 2020. Accessed: 2023-08-25.

[127] National Institute of Standards and Technology. NVD - CVE-2022-25809. https://nvd.nist.gov/vuln/detail/CVE-2022-25809, 2022. Accessed: 2023-02-07.

[128] Andreas Nautsch, Xin Wang, Nicholas W. D. Evans, Tomi Kinnunen, Ville Vestman, Massimiliano Todisco, Héctor Delgado, Md. Sahidullah, Junichi Yamagishi, and Kong Aik Lee. ASVspoof 2019: Spoofing Countermeasures for the De-

tection of Synthesized, Converted and Replayed Speech. *CoRR*, abs/2102.05889, 2021. URL https://arxiv.org/abs/2102.05889.

[129] Santiago Pascual, Antonio Bonafonte, and Joan Serrà. SEGAN: Speech Enhancement Generative Adversarial Network, 2017.

[130] Huy Phan, Yi Xie, Jian Liu, Yingying Chen, and Bo Yuan. Invisible and efficient backdoor attacks for compressed deep neural networks. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 96–100. IEEE, 2022.

[131] Michael Alan Pogue and Philip Ryan Hilmes. Detecting Self-Generated Wake Expressions. https://patents.google.com/patent/US9747899B2/en, 2013. Accessed: 2020-12-04.

[132] Paul Pols and Jan van den Berg. The Unified Kill Chain. *CSA Thesis, Hague*, pages 1–104, 2017.

[133] Alexander Ponticello. *Towards secure and usable authentication for voice-controlled smart home assistants.* PhD thesis, Wien, 2020.

[134] Alisha Pradhan, Kanika Mehta, and Leah Findlater. "Accessibility Came by Accident": Use of Voice-Controlled Intelligent Personal Assistants by People with Disabilities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–13, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356206. doi: 10.1145/3173574.3174033.

[135] PyTorch. PairwiseDistance – PyTorch 1.12 documentation. https://pytorch.org/docs/stable/generated/torch.nn.PairwiseDistance.html, 2022. Accessed: 2022-08-18.

[136] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[137] Se-Hui Ryu, Ki-Hyung Hong, Soojung Chae, and Seok-Jeong Yeon. Augmentative and Alternative Interaction Service with AI Speakers to Access Home IoT Devices and Internet Services for People with Multiple Disabilities. In Klaus

Miesenberger, Georgios Kouroupetroglou, Katerina Mavrou, Roberto Manduchi, Mario Covarrubias Rodriguez, and Petr Penáz, editors, *Computers Helping People with Special Needs*, pages 496–502, Cham, 2022. Springer International Publishing. ISBN 978-3-031-08648-9.

[138] Maria Schmidt and Patricia Braunger. A survey on different means of personalized dialog output for an adaptive personal assistant. In *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*, pages 75–81, 2018.

[139] Bruce Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.

[140] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the Support of a High-Dimensional Distribution. *Neural computation*, 13(7):1443–1471, 2001.

[141] Seeed Technology Co., Ltd. ReSpeaker 4-Mic Array for Raspberry Pi. `https://wiki.seeedstudio.com/ReSpeaker_4_Mic_Array_for_Raspberry_Pi/`, 2018. Accessed: 2022-10-12.

[142] Paul Semaan. Natural language generation: an overview. *J Comput Sci Res*, 1(3): 50–57, 2012.

[143] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pages 1015–1021. Springer, 2006.

[144] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, jan 2014. ISSN 1532-4435.

[145] Statista Inc. Smart home - Statistics & Facts. `https://www.statista.com/topics/2430/smart-homes/`, 2022. Accessed: 2022-08-11.

[146] Dan Su, Jiqiang Liu, Sencun Zhu, Xiaoyang Wang, and Wei Wang. "Are you home alone?" "Yes" Disclosing Security and Privacy Vulnerabilities in Alexa Skills. *CoRR*, abs/2010.10788, 2020. URL `https://arxiv.org/abs/2010.10788`.

[147] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. Light Commands: Laser-Based Audio Injection Attacks on Voice-Controllable Systems. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2631–2648. USENIX Association, August 2020. ISBN 978-1-939133-17-5. URL https://www.usenix.org/conference/usenixsecurity20/presentation/sugawara.

[148] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. Light Commands. https://lightcommands.com/, 2020. Accessed: 2022-11-15.

[149] The MITRE Corporation. CWE Glossary. https://cwe.mitre.org/documents/glossary/index.html, 2008. Accessed: 2023-08-30.

[150] The MITRE Corporation. MITRE ATT&CK. https://attack.mitre.org/, 2013. Accessed: 2023-01-03.

[151] Anton Tomilov, Aleksei Svishchev, Marina Volkova, Artem Chirkovskiy, Alexander Kondratev, and Galina Lavrentyeva. STC Antispoofing Systems for the ASVspoof2021 Challenge. In *Proc. ASVspoof 2021 Workshop*, pages 61–67, 2021.

[152] U.S. Army. *A Military Guide to Terrorism in the Twenty-first Century*. Cosimo reports. Cosimo, Incorporated, 2010. ISBN 9781616401931. URL https://books.google.it/books?id=vmUjcAAACAAJ.

[153] Anton V. Uzunov and Eduardo B. Fernandez. An extensible pattern-based library and taxonomy of security threats for distributed systems. *Computer Standards & Interfaces*, 36(4):734–747, 2014. ISSN 0920-5489. doi: https://doi.org/10.1016/j.csi.2013.12.008. URL https://www.sciencedirect.com/science/article/pii/S0920548913001827. Security in Information Systems: Advances and new Challenges.

[154] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio, 2016.

[155] Francisco Velasco-Álvarez, Álvaro Fernández-Rodríguez, Francisco-Javier Vizcaíno-Martín, Antonio Díaz-Estrella, and Ricardo Ron-Angevin.

Brain–Computer Interface (BCI) Control of a Virtual Assistant in a Smartphone to Manage Messaging Applications. *Sensors*, 21(11), 2021. ISSN 1424-8220. doi: 10.3390/s21113716. URL https://www.mdpi.com/1424-8220/21/11/3716.

[156] Vocera Communications Inc. Vocera Skill for Alexa. https://www.vocera.com/sites/default/files/2022-04/Alexa.3326.SB_.202203.pdf, 2022. Accessed: 2022-11-15.

[157] Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, Lei He, Sheng Zhao, and Furu Wei. Neural Codec Language Models are Zero-Shot Text to Speech Synthesizers, 2023.

[158] Yao Wang, Wandong Cai, Tao Gu, Wei Shao, Yannan Li, and Yong Yu. Secure Your Voice: An Oral Airflow-Based Continuous Liveness Detection for Voice Assistants. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 3(4), December 2019. doi: 10.1145/3369811.

[159] Robert Willison and Mikko Siponen. Overcoming the Insider: Reducing Employee Computer Crime Through Situational Crime Prevention. *Communications of the ACM*, 52(9):133–137, 2009.

[160] Jordan Wirfs-Brock. Learning to listen to data: Voice interaction, sonification, and narrative. In *Companion Publication of the 2021 ACM Designing Interactive Systems Conference*, DIS '21 Companion, page 7–10, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450385596. doi: 10.1145/3468002.3468231. URL https://doi.org/10.1145/3468002.3468231.

[161] Wolfson, Sam. Amazon's Alexa Recorded Private Conversation and Sent it to Random Contact. https://www.theguardian.com/technology/2018/may/24/amazon-alexa-recorded-conversation, 2022. Accessed: 2023-02-14.

[162] Zhizheng Wu, Nicholas Evans, Tomi Kinnunen, Junichi Yamagishi, Federico Alegre, and Haizhou Li. Spoofing and Countermeasures for Speaker Verification: A survey. *speech communication*, 66:130–153, 2015.

[163] Wenjun Xiong and Robert Lagerström. Threat modeling – a systematic literature review. *Computers & Security*, 84:53–69, 2019. ISSN 0167-4048. doi:

https://doi.org/10.1016/j.cose.2019.03.010. URL https://www.sciencedirect.com/science/article/pii/S0167404818307478.

[164] Jiao Yabing. Research of an improved apriori algorithm in data mining association rules. *International Journal of Computer and Communication Engineering*, 2(1):25, 2013.

[165] Junichi Yamagishi, Xin Wang, Massimiliano Todisco, Md Sahidullah, Jose Patino, Andreas Nautsch, Xuechen Liu, Kong Aik Lee, Tomi Kinnunen, Nicholas Evans, and Héctor Delgado. ASVspoof 2021: Accelerating Progress in Spoofed and Deepfake Speech Detection, 2021.

[166] Chen Yan, Yan Long, Xiaoyu Ji, and Wenyuan Xu. The Catcher in the Field: A Fieldprint Based Spoofing Detection for Text-Independent Speaker Verification. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 1215–1229, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367479. doi: 10.1145/3319535.3354248.

[167] Chen Yan, Xiaoyu Ji, Kai Wang, Qinhong Jiang, Zizhi Jin, and Wenyuan Xu. A Survey on Voice Assistant Security: Attacks and Countermeasures. *ACM Comput. Surv.*, 55(4), nov 2022. ISSN 0360-0300. doi: 10.1145/3527153. URL https://doi.org/10.1145/3527153.

[168] Qiben Yan, Kehai Liu, Qin Zhou, Hanqing Guo, and Ning Zhang. SurfingAttack: Interactive Hidden Attack on Voice Assistants Using Ultrasonic Guided Waves. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020. URL https://www.ndss-symposium.org/ndss-paper/surfingattack-interactive-hidden-attack-on-voice-assistants-using-ultrasonic-guided-waves/.

[169] Sung-Hyun Yoon, Min-Sung Koh, Jae-Han Park, and Ha-Jin Yu. A new replay attack against automatic speaker verification systems. *IEEE Access*, 8:36080–36088, 2020.

[170] Zhiyuan Yu, Yuanhaur Chang, Ning Zhang, and Chaowei Xiao. SMACK: Semantically Meaningful Adversarial Audio Attack. 2023.

[171] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, XiaoFeng Wang, and Carl A. Gunter. Commandersong: A Systematic Approach for Practical Adversarial Voice Recognition. In *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, page 49–64, USA, 2018. USENIX Association. ISBN 9781931971461.

[172] Robert Zager and John Zager. Ooda loops in cyberspace: A new cyber-defense model. *Journal Article— October*, 20(11):33pm, 2017.

[173] Zengfeng Zeng, Dan Ma, Haiqin Yang, Zhen Gou, and Jianping Shen. Automatic intent-slot induction for dialogue systems. In *Proceedings of the Web Conference 2021*, pages 2578–2589, 2021.

[174] Albert Zeyer, Parnia Bahar, Kazuki Irie, Ralf Schlüter, and Hermann Ney. A comparison of transformer and lstm encoder decoder models for asr. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 8–15. IEEE, 2019.

[175] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. DolphinAttack: Inaudible Voice Commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 103–117, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349468. doi: 10.1145/3133956.3134052.

[176] Linghan Zhang, Sheng Tan, and Jie Yang. Hearing Your Voice is Not Enough: An Articulatory Gesture Based Liveness Detection for Voice Authentication. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 57–71, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349468. doi: 10.1145/3133956.3133962.

[177] 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 *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1381–1396, 2019.

[178] Yichi Zhang, Bryan Pardo, and Zhiyao Duan. Siamese Style Convolutional Neural Networks for Sound Search by Vocal Imitation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(2):429–441, 2018.

[179] Yi Zhou, Xiaohai Tian, Haihua Xu, Rohan Kumar Das, and Haizhou Li. Cross-lingual Voice Conversion with Bilingual Phonetic Posteriorgram and Average Modeling. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6790–6794, 2019. doi: 10.1109/ICASSP.2019.8683746.

[180] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005.