Analyzing the Privacy Attack Landscape for Amazon Alexa Devices

Raphael Leong
ryl15@ic.ac.uk
Imperial College London

Abstract

Ever since the introduction of Amazon's new intelligent personal assistant Alexa along with their Echo speaker counterpart, there has been a significant rise in interest and usage of smart-home products with over 10,000 Alexa skills currently available on the market for customers to use with their newfound Echo devices. The focus of our research is to investigate the potential privacy issues that may emerge in the context of always-on speakers that are now prevalent in people's homes. In this work, we organized study papers and industrial research into specific categories which can lead to potential privacy issues. Within our research, we analyzed the all potential security and privacy issues surrounding Alexa skills, Echo firmware and third-party hardware that acts as alternatives to the Echo device. Our core plan initially was to evaluate and statically analyze all the available Alexa skill repositories that were made public by developers on Github. We then branched out to look at issues regarding Echo firmware which is based off the Android OS and also review the state of third-party Alexa devices. Ultimately, we aim to give a concise overview of the current privacy attack landscape for Alexa devices and discuss potential security and privacy issues with voice enabled devices that could arise in the near future.

1. Introduction

Background

With the introduction of Amazon's new intelligent personal voice assistant, Alexa, along with the new Echo speaker to accompany has brought up many concerns about potential security and privacy issues. This new service provides many applications such as ordering items, home automation and music functionality.

However, there have been privacy concerns within these custom skills regarding what personal information and private data are being passed onto the Alexa service and furthermore what is processed and sent to any third party if possible. Amazon has stated that only information is recorded after a 'wake word' is heard and the Echo device is only listening out for this phrase when it is turned on, not recording any extra data. Problems surrounding the hardware have been brought up as well with the possibility of wiretapping due to the Echo device having exposed debug pads at the base of the device and a configuration setting that allows the device to boot via an external SD card.

Alexa Skills are the voice-enabled apps that developers can create for the Alexa service that allow users to interact with their systems. Amazon has provided a platform within the Amazon Web Services (AWS) to assist developers in creating these apps in the form of the Alexa Skills Kit interface. Developers commonly write their skill's codebase in JavaScript or Python and these are built primarily as either basic response apps or RESTful web services to interact with third party API. These are usually hosted on an online service provided by Amazon called the AWS Lambda services or an alternative HTTP endpoint such as Heroku or another cloud-based VM service.

These hardware vulnerabilities are not only limited to Amazon's own products but also to other third-party speakers that are Alexa-enabled with the Amazon Voice Services helping develop these alternative options.

To fully understand the privacy of user data within the Alexa ecosystem, we have to analyze and understand how the data flows from the user to the various platforms within Amazon Web Services and possibly other third-party access points. This is to aid us in recognizing at which points the user's private information may be leaked to either Amazon or other third-party sources which can potentially lead to other misuse and exploit. To help us in understanding how data flows within the ecosystem, we constructed an architectural model to structure the important sections that communicate information between each other.

Initially, the user must speak into the Echo device to interact with Alexa. The speaker then records the voice data from the user and sends the speech input to the Alexa Voice Service (AVS). The user can alternatively be using another third-party device that's not produced by Amazon. Once the AVS receives the voice data, it then parses the speech input and processes it into a JSON format for further usage. It then sends the data to the Alexa Skill Kit interface where it decides which intent function the user has requested to interact with. This involves pattern matching the speech text with which sample utterance it is associated with. It then sends the JSON data to the HTTP endpoint where the server code is running on and will execute the appropriate function. At this point, it may interact with other third-party REST APIs to retrieve and process extra necessary information. Once the server has finished executing the request, it returns the JSON output data back to the Alexa Web Services, where the AVS will process the text into speech data. It then delivers the Alexa voice recording to the speaker device for the user to listen to Alexa's response.

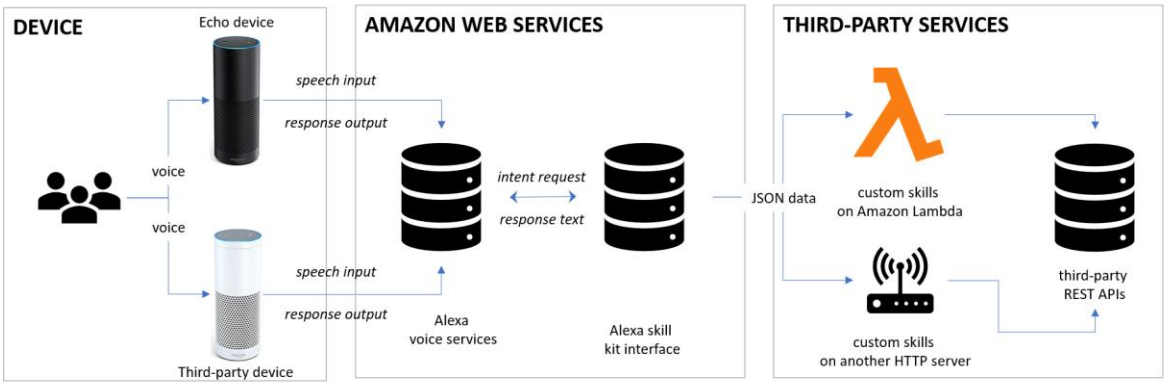

Figure 1: Architecture of Amazon skills and Voice services

With the ever-increasing popularity of always-on voice-enabled devices, customers are becoming more aware and afraid of their privacy potentially being at risk to malicious hackers. With articles discussing how Amazon and Google are constantly eavesdropping and with the possibility of having recorded conversations being leaked to third-parties and vulnerable to exploits. Amazon has also considered giving access to user audio recordings for third-party developers which raises concerns from the relevant parties, unsure of whether their private information is protected and secure.

Our Contributions

We perform an analysis of several aspects of the Alexa ecosystem with the goal of finding and validating potential sources of privacy leaks and data-related vulnerabilities. Specifically, this the list of aspects of the Alexa ecosystem we plan to research and analyze throughout the project:

- Alexa skills
- Echo firmware and Android APKs
- Third-party hardware

2. Related Work

Academic Research

There are only a handful of academic research papers that delve into aspects of the attack landscape of Alexa devices. Preliminary work involved looking at the basic Alexa architecture and how the user interacts with the Alexa skills and its interface, investigating how their security and privacy may be compromised in using these devices. Since Alexa Echo devices also use an Android OS, we also considered research papers which researched the vulnerabilities of Android apps and firmware.

Android Privacy Issues

There's another research project conducted by researchers at the Citadel that goes through the various methods of analyzing the vulnerabilities of the Amazon Echo [1]. Primarily it goes into the hardware vulnerabilities within these devices and the lack of security within the firmware. It aims to find the various attack points that are available

within the Echo hardware such as the SD card pinout, showing that these can allow the hacker to hijack the firmware and gain unauthorized access into the file system.

TaintDroid was a project designed to investigate smartphone operating systems and highlight how much private data can be potentially misused by third-party applications on the device [2] . The study goes into Android operating systems and how there is a lack of control for the user in how much of their private data can be accessed by third-party apps. They then developed a real-time monitoring service known as TaintDroid, which was designed to provide an analysis of how private information is collected and used by applications downloaded on the system. The service scans for any instance of potential misuse of the user's private data within each application and provides ways to identify potentially malicious applications.

A research paper written by researchers at the University of Wisconsin [3], discusses general firmware vulnerabilities within embedded systems, describing that traditional source-code analysis tools are not that effective when working with firmware programs due to the specific architectural features of these systems. They then went to create FIE, a new tool to be able to analyse firmware programs through symbolic execution. The tool is described as a program that "*builds off the KLEE symbolic execution engine in order to provide an extensible platform for detecting bugs in firmware programs for the popular MSP430 family of microcontrollers. FIE incorporates new techniques for symbolic execution that enable it to verify security properties of the simple firmwares…*". They then demonstrated its utility by applying the tool to 99 open-source firmware programs and are able to verify security for most of the program and are able to detect several bugs.

To fully understand the flaws of Android systems, we looked through survey papers to research and analyze the full extent of privacy leaks when using Android. One survey paper reviews the current state of the Android ecosystem [4], citing several security research sources on the Android platform. They illustrate how the openness and portability of Android results in the system being easily exploitable and vulnerable to many different points of attack. They discuss the problems associated with the ecosystem and present open-ended problems that may be prevalent in the future. Another survey paper written by researchers of Carnegie Mellon University [5] also details the current attack landscape for the Android platform. It discusses aspects of the Android security model and the forms of attack that are made possible due to the weaknesses within the model. Both papers mostly talk about similar concepts of the Android system and propose their own solutions to these problems the ecosystem has.

There's also been a discrepancy between what Amazon has told us about when Alexa is listening in and when it is recording your conversation. Articles such as this one talks about the basic information that are necessary for customers to know when their personal Echo is 'eavesdropping' on their conversations. People fear that their privacy is compromised whenever the device is in their room due to the possibility of wiretapping and hackers accessing their private recordings stored online. Amazon have disclosed that they only record conversations after the 'wake word' is uttered. They have stated that the device is 'always on' and actively listening, which sends back data to Amazon to help develop their speech recognition interface. The Arkansas case is a prime example of this where the authority requested Amazon for all voice recordings from the Echo within the crime scene. Cracking the security of the account and retrieving the private data is possible for Amazon to do but users are concerned whether their private conversations can be easily exposed and potentially misused.

Amazon Echo Privacy Issues

A recent paper was published on the privacy of Amazon's Alexa skills ecosystem [6], analyzing how secure the user's private data is within the platform. In the paper, they study the choices made by Amazon regarding the privacy of the users when they access Skills developed by people using the Alexa Skills Kit platform. Specifically, they analyze the privacy policies set up by the developers for all Alexa Skills available on Amazon. From their results, they find that: "*75% do not have one. Furthermore, even among those Skills for whom a privacy policy is required by Amazon's policy, 3.5% do not have a valid one and 70% are not customized to Alexa*". Throughout the paper, they discuss issues that are evident, describing how private data of the user is handled by Amazon and how much developers have access to it. The paper raises awareness of privacy problems brought by the current design choices and provides possible solutions to help protect customer's personal information. In our work, we consider the actual Alexa skills and not the attached privacy policies or lack thereof.

Industrial security research

Skills

A report from EY from the Future of Privacy Forum provides preliminary information about the privacy implications concerning microphone-enabled devices such as the Amazon Echo device. It talks about the recent development in speech recognition technology and more of these devices are becoming readily available. The article brings up the legal issues that surround these kinds of devices being "always on" and how that might violate federal wiretapping laws, cited within the report. There are numerous other articles from various news sources on similar topics, indicating the arising number of issues that concern customers with the security of their private data. As the industry continues to move forward with speech recognition devices, the privacy issues associated will start to grow and they bring up useful guidelines to provide security for users in the future.

A recent report by VoiceLabs discusses the emerging market for 'Voice-First' devices and the importance of the growing ecosystem for these kinds of devices. It goes into detail about Voice-First stack and how different parts of the architectural model interact with each other and how it relates to the customer experience. The report includes many statistics about Amazon's Echo device and its competitors such as the Google Home. As the trend of people having more voice-first devices in their homes, it is slowly getting closer to the mainstream with more companies such as Microsoft and Apple. In addition to this, the report also examines the significant amount of 'Zombie Skills' within the Alexa platform, indicating that out of over 7,000 skills, "only 31% have more than one customer review". This is an important point as there is a huge number of applications that are readily available but are not used or maintained by developers. This could potentially lead to several privacy issues with existing users as these apps may contain several security problems but due to lack of public attention for the skills, developers are less inclined to fix them. Amazon is already trying to combat this by providing more tools and support for developers to keep their skill relevant and increase retention with customers.

Firmware Issues

Within an article written by Mark Barnes at MWR Labs, it describes how they broke down the hardware within Echo device and managed to find several ways to setup possible exploits such as wiretapping a personal Echo device to eavesdrop on people's conversations within their homes. It takes inspiration from a research paper mentioned earlier [1]. This can be done due to a vulnerability within the manufacturing of the hardware during its earlier versions as the debug pads were left exposed at the base of the device, possibly as an oversight. This allowed them to boot the Echo from an external SD card leading to unauthorized access of the device. The post goes into intricate detail of how it manages to break into and setup this exploit and talks about possible countermeasures.

There were several recent blogposts from people who work in cybersecurity detailing security flaws with the Amazon Echo device. The writer demonstrates their method in which he manages to intercept the Android firmware update for the device due to the download being sent over HTTP. This makes it very simple to use programs such as Wireshark to obtain the image file. They then attempt to analyze the file, identifying the file system of the operating system and discovering many APK files in the process. The project enables other researchers to reverse engineer the APK files and possibly observe any vulnerabilities within the code. Google also has a security bulletin for the Android operating system which releases a list of critical vulnerabilities within the firmware, eventually leading to their fixes in the following months.

3. Skills

Background

When creating these skills, developers must provide the intent schema which is a JSON structure that declares what services the skill can provide when it receives the speech input. This is then matched with a sample utterance for the system to recognize which intent is being called. When a phrase is uttered, the Alexa service will attempt to match the voice heard to one of the intents and will execute the function associated with the intent. The spoken input data can also contain custom slot types which are a list of values that are grouped under specific tags which can then be later referenced in the intent functions.

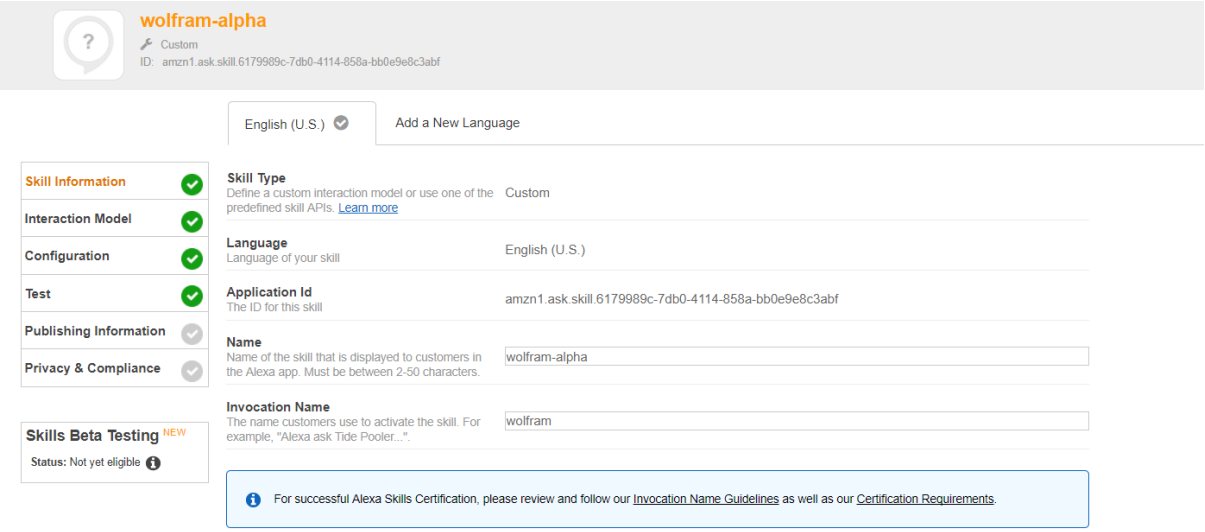

Figure 2: Example of Alexa Skill Kit Interface

After designing their sample utterances and the intent schema, developers need to setup their Skill on the Alexa Skill Kit platform. The interface provides settings for the Skill which the developer must setup such as the name of the application and the invocation name in which the Skill is referred to by when activated. The developer also needs to provide details regarding where the codebase of the skill is hosted on and insert their intent schema and sample utterances. After all the necessary components of the skill have been setup, developers can also use the platform to test the skill with sample responses.

Finally, the developer can submit the skill to the store once it is ready to be deployed. They can opt to submit a privacy policy along with the skill. For the app to be processed in the store, it has to go through a review process by Amazon to ensure the skill fits the necessary security and privacy requirements.

Privacy Issues Around Skills and Potential Attacks

There is an arising issue in how user's personal data has been handled by Amazon within the Alexa ecosystem. As mentioned earlier, a report detailing the privacy of the Alexa Skills platform states that up to 75% of skills lack a privacy policy whilst a significant percentage of ones that do have one are invalid or not customized specifically for Alexa [6]. This is a huge problem for customers using Alexa skills as there is a lack of communication between them and the developers concerning how much of their personal information is being exposed and used for their application and possibly at the risk of malicious activity.

These skills pose several different issues regarding the privacy of the user's personal and private information. The Alexa devices are only listening and recording information when the phrase 'Alexa' is uttered and only the relevant skill is activated when the phrase 'ask <Skill Name>' is heard. Therefore, the potential problem that arises here is when the user may say private information that is not intended to be recorded. This may coincidentally pass the requirements for Alexa to record this input and pass the data into the skill which may further lead to the sensitive information being transferred to a third party.

Potential Attack Scenario

There are several potential attacks involving Alexa skills that can leave the user's privacy vulnerable to malicious activity. There are common methods of exploit which are related to security flaws within standard web applications such as having several points throughout the code where private data may leak to third parties or possible points of injection where it can affect the user's system.

A unique attack scenario for Alexa skills may involve designing sample utterances and custom slots in a way where the user may accidentally utter important information in conversation where Alexa will record and process it without their intention. This can contain private data which malicious developers can now easily access. This occurs when an intent's utterance pattern is designed with short common phrases, with a custom slot that can be matched to a user's specific private information.

An example of this is where the sample utterance is **"get CODE_NUM"** and the intent function reads the value of CODE_NUM in a number format, sending the information to a third-party API. The phrase only includes two words and can possibly be misinterpreted during conversation by Alexa, potentially leaking private information to third parties.

Analysis of Alexa Skills

Initially, to start analyzing potential security flaws within Alexa apps, we had to collect all available public repositories of skills that were available on Github. Since there are over 10,000 skills and all of them use a wide variety of SDKs to develop their app, we try to narrow down the type of skills we choose to analyze to ensure they are written in the same pattern, making it easier to perform static analysis. Mainly, we aim to analyze skill that follow a certain design format using the node module, alexa-app. Our initial plan was to do basic analysis of these type of apps as it allowed us to use another module named alexa-app-server to test these skill within a local environment.

To accomplish this, I wrote a python script that uses the Github search API which scans and retrieves any repository that fit the following requirements:

- Repository search term contains the term 'alexa'
- Codebase is written in Javascript
- Contains the pattern: require('alexa-app')

The script collects the URL of all available repositories that only downloads and stores the skills that are relevant for the static analysis. Since the search API limits the number of requests that can be made before a timeout occurs, we only collected repositories that were created after 2015, around the time when Amazon Alexa was released to the public.

We ended up finding around 200 repositories that can be used for static analysis. For the static analysis, we used a bash script to run grep commands to identify useful features of each repository. The script searches for any common patterns relating to the statistics we wish to find out about:

- Number of slots
- Number of data sink module calls
- Number of intent functions

Here we try to find out ways users can somehow have their privacy violated when using the various skills available to them. Slots are important to understand where private information can potentially leak to third party and intent functions are the primary methods for users to interact with Alexa. We also want to understand when data is leaving the application to either Amazon or third-party sources, as this will give an insight to when user private data is potentially at risk.

Evaluation of Privacy Issues in Alexa Skills

Manual Evaluation

Name	Analytics (Notes)	Destinations (URI Endpoints)	NPM modules used as data sinks
airportinfo	1 app intent call **Intent['airportinfo']:** Displays airport info, triggered by user asking for airport status code, uses FAA data helper module file Uses 'request-promise' module to send HTTP request to URI endpoint. "rp(options)"	'http://services.faa.gov/airport/status/'	request-promise
astronomer	4 app intent calls (2 amazon support intents) Primarily uses request module across all intent functions 'request' module is used to send HTTP request to URI endpoint	http://api.sunrise-sunset.org/json http://api.usno.navy.mil/moon/phase?date= api.openweathermap.org/data/2.5/weather	request
home automation	9 app intent calls (3 amazon support intents) **Intents: {Switch, SetColor, SetLevel, SetTemp, SetMode, GetMode, VoiceCMD}** Used for controlling house appliances automatically that are connected with openHAB application Uses 'request' module to send HTTP request to openHAB server URI setup by user in config.js **Intents['Research']:** Allows for research using wolfram API, setup using wolfram.js library Uses 'https' module to send HTTP request to URI endpoint for wolfram search API	User's openHAB config server https://api.wolframalpha.com/v2/query?input=	request https
wolfram	1 app intent call **Intent['askWolfram']:** Handles queries uttered by user, uses wolfram service api to search for result 'request-promise' module used to send request to URI endpoint. "rp(opts)"	http://api.wolframalpha.com/v2/query	request-promise
tesla	20 app intent calls **Intent: {'VehicleCountIntent', 'BatteryIntent', 'RangeIntent', 'PluggedIntent', etc.}** Used for controlling user configured tesla electric cars Makes use of teslajs REST api to communicate with the URI endpoint using request module within.	https://owner-api.teslamotors.com https://streaming.vn.teslamotors.com/stream	Teslajs (which uses 'request' module)
pivotal tracker	1 app intent call **Intent['CheckNotifications']:** Checks user's notifications based on their pivotal tracker account Fetches data from URI endpoint using module 'node-fetch'	https://www.pivotaltracker.com/services/v5/my/notifications?envelope=true	node-fetch

alexa-mta	1 app intent call **Intent['MTAIntent']:** Checks for next train time depending on what user uttered for train name Uses train schedule in repo to find next train time. No information is passed outside of application to other third party APIs	No endpoint just uses train schedule found in repo	n/a
marketing cloud	5 app intent calls (1 amazon support intent) **Intent: {'transactions', 'maxTransactions'}** Finds out information about number of transactions from time period specified by user. **Intent: {'issuesDay', 'issuesMonth'}** Finds out the number of issues during specificied time period MC-data-helper module helps with retrieving information Uses 'request-module' to communicate with marketing cloud API to retrieve information "rp(options)"	https://trust.marketingcloud.com/trans/count/last/ https://trust.marketingcloud.com/timelines/timelineUIDayPublic https://trust.marketingcloud.com/timelines/timelineUIMonthPublic	request-promise

Case Studies

In this section, we will be going through three different Alexa skills to help further understand any potential vulnerabilities for leaks of private data. The three skills that we will discuss in detail are:

- Alexa-airportinfo
- Alexa-HA
- Alexa-tesla

The skill, alexa-airportinfo, is an application which helped us initially to understand how a standard RESTful skill should be designed. The skill is part of a tutorial that teaches developers how to create Alexa skills locally using Nodejs with the 'alexa-app' module. When analyzing this skill manually, we found out that it was a relatively standard web application written in the NodeJS framework. There are few potential leaks for private data within the skill. There is only one sink point within the application where the user asks for the status of a certain airport, inputting custom data within a slot called 'FAACODES' which only includes certain combination of various code names for airports. If the user accidentally utters private data, the contents of the voice data will still be parsed and recorded but will not likely be sent to the third-party API as it will not be processed due to failing the pattern match. Most standard skills are structured in this method, using various RESTful third-party APIs to communicate with outside HTTP endpoints resulting in minimal risk for leaks of private data.

Home automation is one of the unique features that Alexa provides. Smart-home products involve themselves by connecting to a hub device or server point where they will communicate with the skill to perform various functions. We decided to investigate a skill that uses OpenHAB as its main home automation system. The skill offers many more intent functions compared to the alexa-airportinfo skill, including more custom slots. Therefore, there are more points for potential leaks of private data using this application although as with other skills, it is not likely to occur. The user must also setup their account and server details within the application for Alexa to be able to communicate with OpenHAB and the relevant devices. This information is stored within a file in the repository which is not encrypted and secure. This can lead to several potential risks as hackers can use the server details to maliciously attack the smart-home devices and possibly gain access to private information inside the user's home.

We also decided to research a skill that uses the TeslaJS node module to help monitor a user's vehicles. The data sink API is more securely designed compared to other REST node APIs, using an OAuth-based authentication for their security measures. For each new session, the application must generate a token to authenticate the user to be able to access and process their account information. Compared to other Alexa skills that are designed in this manner, this is significantly more secure as it applies complex security measures where others may not use them. Therefore, the user's privacy is unlikely to be compromised when using this skill

Static Analysis

Here are the results after running the scripts to collect the Alexa skill repositories and statically analyze them using a combination of bash and python commands. The histograms are generated using a python module called plot.ly.

```
199 repos analysed.
88 repos could not be analysed.
request: 28
request-promise: 55
others/local server: 116
```

Figure 3: Result output from console after executing bash script

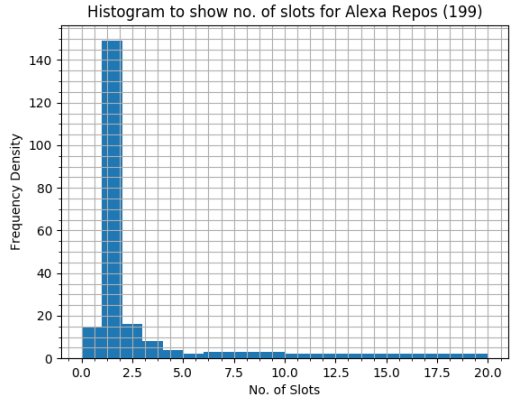

Figure 3(i): Histogram showing no. of slots per Alexa skill

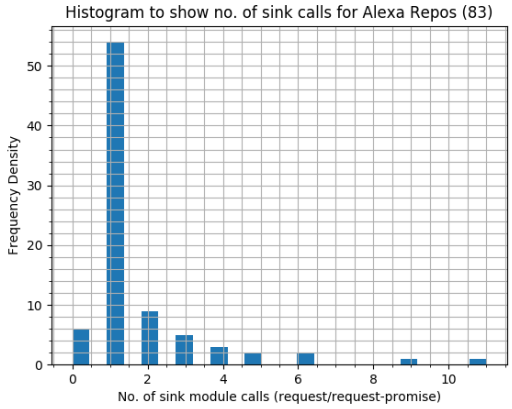

Figure 3(ii): Histogram showing no. of sink calls per Alexa skill

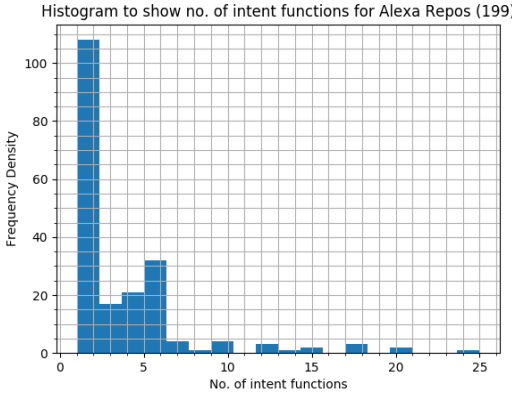

Figure 3(iii): Histogram showing no. of intent functions per Alexa skill

Applying Static Analysis to Alexa Skills

- Many of the repositories we found used request and request-promise as data sink modules, from the static analysis results above.
- Majority of skills have at least one intent function and one custom slot as shown by the first two histograms.
- The skills analyzed that had request and request-promise modules mostly have at least one data sink call shown by the above histogram.
- Data sink module calls often lead to third party HTTP endpoints as either GET or POST request.
- Outliers with no sink module calls are often simple response skills with no HTTP calls.

In conclusion, there are access points within each skill where the user's data will leave the app for third-party APIs to potentially use and exploit. Despite this, most of these skills act as standard Nodejs applications using RESTful APIs to accomplish the application's intent. The only difference between normal Nodejs applications and Alexa skills is that they use custom SDKs to communicate with the Alexa platform and Alexa Voice Services to enable speech recognition, using custom slots to transfer information. Therefore, customers should be expecting similar risks when downloading Alexa skills, similarly when using standard web apps.

4. Amazon Firmware

Background

Firmware exists in all devices we use everyday, therefore it is not surprising there are also many exploits for these kind of software. These kinds of attacks are made possible due to the nature of firmware being very vulnerable and insecure. Firmware exploits have been reported since as early as 1998 with the CIH computer virus. Hackers have use methods such as concealing malicious code on USB sticks to attack a user's computer when they access it. In an article, security researcher, Karsten Nohl, managed to perform this attack last year, stating that "*there's firmware everywhere in your computer, and all of it is risky*", proving that even common day-to-day devices are not even safe from firmware attacks. Inherently, most hardware designers do not make their products secure against firmware vulnerabilities, even lacking the most basic of countermeasures such as cryptically signing a signature to authenticate the firmware. Despite this, creating a firmware attack takes a considerable amount of effort and ability due to the changing versions of firmware and having to develop exploits separately.

In a research post investigating the Echo firmware, the writer highlights that the Amazon Echo is built on the Amazon Fire OS which is a stripped-down version of the Android OS. Therefore, our research aims to look at vulnerabilities within Android firmware and any potential privacy issues that may arise when using the Echo device.

There are many other ways to exploit the firmware vulnerabilities rather than going through the trouble of modifying code of the system. Theoretical research on firmware hacking has led to new ways of being able to jump through loop holes left exposed in the system architecture. Common attacks involve 'bricking' the device by corrupting the firmware code rather than modifying it. Charlie Miller demonstrated this after discovering Apple laptop lithium ion batteries were shipped with default passwords within the code, allowing him to manipulate the firmware and cause the system to overcharge.

Despite this, Apple is still one of the few companies that set up significant firmware protection against such exploits, where they digitally sign their firmware and firmware updates for the iPhone. Although secure countermeasures are in place, research teams at Kaspersky's Lab have stated that you can instead exploit the baseband firmware vulnerabilities and attack the phones when they connect to the network. In 2011, Ralf-Philipp Weinmann has done this by hacking the firmware and delivering malicious code to any mobile phones connected to the network. This could turn them into listening devices that could tap into your conversations, though he had to set up his own fake cell tower for the devices to connect to.

There have been several suggestions in ways to combat against firmware attacks. Currently, antivirus software does not scan the firmware for any malicious activity. Manufacturers can start building their hardware with better security measures against these kinds of exploits. Signing their hardware with authenticated security certificates would prevent unauthorized access to the firmware. This would be a basic way to stop low level tampering of the firmware but more persistent hackers could easily obtain master keys to sign their attack code with and still exploit the firmware's vulnerabilities.

Disassembling Alexa Firmware

In researching the possible vulnerabilities surrounding the Amazon Echo devices, people have managed to get ahold of a version of the Amazon Android OS firmware through intercepting updates sent to their devices. Initial background research started with poking around the hardware, attempting to find any vulnerabilities within the system. They discovered that it was possible to attain copies of the firmware from Amazon as they pushed the .bin files over HTTP making it very simple to intercept these transmissions. They discovered that the filesystem was a typical Android OTA update filesystem and began analyzing its structure and contents.

We managed to get a hold of the download copy of the .bin image file from the one of the writers earlier mentioned. Using this, we attempted to disassemble and analyze possible vulnerabilities within this version of the firmware of the Echo device.

Initially, I tried using several different image file processing tools to turn the bin file to an .iso image file where then I can mount it to a virtual drive and analyze the filesystem. This lead to several problems as most of the time the ISO software returned failed results and would report that the file was corrupted. Within the blogpost, the writer mentioned that it used a program called binwalk to reverse engineer and extract the firmware. This resulted in a standard Android filesystem.

```
-rw-r--r-- 1 user user   678364 Dec 25 17:15 amazon.jackson-19.apk
-rw-r--r-- 1 user user    59016 Dec 25 17:15 android.amazon.perm.apk
-rw-r--r-- 1 user user    15936 Dec 25 17:15 AuthUtilsService.apk
-rw-r--r-- 1 user user  1592978 Dec 25 17:15 Bluetooth.apk
-rw-r--r-- 1 user user     2817 Dec 25 17:15 BluetoothController.apk
-rw-r--r-- 1 user user  1336616 Dec 25 17:15 CertInstaller.apk
-rw-r--r-- 1 user user   137950 Dec 25 17:15 com.amazon.communication.apk
-rw-r--r-- 1 user user  3381266 Dec 25 17:15 com.amazon.device.bluetoothdfu.apk
-rw-r--r-- 1 user user   749596 Dec 25 17:15 com.amazon.device.sync.apk
-rw-r--r-- 1 user user   967225 Dec 25 17:15 com.amazon.device.sync.sdk.internal.apk
-rw-r--r-- 1 user user     9982 Dec 25 17:15 com.amazon.dp.logger.apk
-rw-r--r-- 1 user user   603222 Dec 25 17:15 com.amazon.imp.apk
-rw-r--r-- 1 user user  3209008 Dec 25 17:15 com.amazon.kindleautomatictimezone.apk
-rw-r--r-- 1 user user  3972953 Dec 25 17:15 com.amazon.kindle.rdmdeviceadmin.apk
-rw-r--r-- 1 user user     3799 Dec 25 17:15 com.amazon.platformsettings.apk
-rw-r--r-- 1 user user   322178 Dec 25 17:15 com.amazon.tcomm.apk
-rw-r--r-- 1 user user   211709 Dec 25 17:15 CrashManager.apk
-rw-r--r-- 1 user user     9099 Dec 25 17:15 DefaultContainerService.apk
-rw-r--r-- 1 user user    52082 Dec 25 17:15 DeviceClientPlatformContractsFramework.apk
-rw-r--r-- 1 user user  1252024 Dec 25 17:15 DeviceMessagingAndroid.apk
-rw-r--r-- 1 user user    38995 Dec 25 17:15 DeviceMessagingAndroidInternalSDK.apk
-rw-r--r-- 1 user user    40536 Dec 25 17:15 DeviceMessagingAndroidSDK.apk
-rw-r--r-- 1 user user   278931 Dec 25 17:15 DeviceSoftwareOTA.apk
-rw-r--r-- 1 user user    47557 Dec 25 17:15 DeviceSoftwareOTAContracts.apk
-rw-r--r-- 1 user user   190978 Dec 25 17:15 DownloadProvider.apk
-rw-r--r-- 1 user user   165353 Dec 25 17:15 FireApplicationCompatibilityEnforcer.apk
-rw-r--r-- 1 user user    10792 Dec 25 17:15 FireApplicationCompatibilityEnforcerSDK.apk
-rw-r--r-- 1 user user     4618 Dec 25 17:15 fireos-res.apk
-rw-r--r-- 1 user user     7157 Dec 25 17:15 FireRecessProxy.apk
-rw-r--r-- 1 user user 14446992 Dec 25 17:15 framework-res.apk
-rw-r--r-- 1 user user     8440 Dec 25 17:15 FusedLocation.apk
-rw-r--r-- 1 user user   140960 Dec 25 17:15 InputDevices.apk
-rw-r--r-- 1 user user    48406 Dec 25 17:15 KeyChain.apk
-rw-r--r-- 1 user user   767179 Dec 25 17:15 LogManager-logd.apk
-rw-r--r-- 1 user user    39111 Dec 25 17:15 MetricsApi.apk
-rw-r--r-- 1 user user   347814 Dec 25 17:15 MetricsService.apk
-rw-r--r-- 1 user user     2783 Dec 25 17:15 Provision.apk
-rw-r--r-- 1 user user    15956 Dec 25 17:15 RemoteControlManager.apk
-rw-r--r-- 1 user user   149422 Dec 25 17:15 RemoteSettingsAndroid.apk
-rw-r--r-- 1 user user    39168 Dec 25 17:15 RemoteSettingsInternalSDK.apk
-rw-r--r-- 1 user user    99144 Dec 25 17:15 SettingsProvider.apk
-rw-r--r-- 1 user user    33401 Dec 25 17:15 Shell.apk
-rw-r--r-- 1 user user     4002 Dec 25 17:15 shipmode.apk
-rw-r--r-- 1 user user     3713 Dec 25 17:15 SimpleLauncher.apk
-rw-r--r-- 1 user user     3909 Dec 25 17:15 ThrottleDownloads.apk
```

Figure 4: List of APKs from Echo filesystem (taken from blogpost)

Some of these APKs are common developer software and after searching online, we managed to find several download links for them. We then attempted to use different Android Security tools, such as QARK, to get an idea of what kind of vulnerabilities and exploits are evident within each APK file.

QARK analysis

In researching which APKs are accessible to us for analyzing, we managed to successfully retrieve and reverse engineer 3 files using jadx. Notably these are the ones that also produced significant results from using QARK:

- com.amazon.device.sync.apk
- com.amazon.tcomm.apk
- SimpleLauncher.apk

After decompiling the APK files, the QARK software fully analyzes the source code. It lists out possible sources of issues and identifies which part of the code causes it.

The types of errors and exploits it finds range from insufficient permission checks within plugins and method calls that can provide the means for malicious hackers to perform data injection attacks. It also provides details about when potential leakage of data can occur within the source code. After it provides us with a list of potential holes within the code, we check the code that QARK identifies a problem with and attempt to analyze if the issue is critical to the security and privacy of the user.

Based on our limited experience with this, however, we didn't see the tool identifying anything that can potentially be an exploitable vulnerability. So, in a way, the search for possible exploitable issues in firmware APKs continues.

5. Third-party Hardware

Explaining how it's built

The introduction of Alexa Voice Service has opened possibilities for developers and manufacturers to build their own custom microphone enabled device. With examples such as Anker's Eufy Genie, there is a new market for cheaper alternatives of Alexa enabled devices, allowing customers to interact with the Alexa interface and its skills without having to invest in the Amazon's Echo device.

Constructing a custom Alexa prototype can be easily done with the sufficient knowledge and background in the area. An online project highlights how it is possible for experienced users to build their own hands-free Alexa prototype on a Raspberry PI. It demonstrates that the Alexa Voice Service platform enables many third-party developers to help build and manufacture various custom Alexa enabled speakers.

There are inevitably several risks when using third-party hardware due to the lack of authenticity and integrity of a first-rate product. An off-brand device may lack the necessary security and privacy regulations compared to what Amazon currently have. Customers will have to take this into account when deciding to use alternative products for the Alexa service.

Potential Attacks

There are many risks involving third-party hardware as there are points of vulnerabilities that malicious developers can easily setup and access. Potential attacks include but are not limited to:

- **Wiretapping the device:**
  Manufacturers are free to design and build their hardware in any fashion they want to. This allows the developers to easily tamper with the device maliciously to their needs such as wiretapping Echo speakers to eavesdrop on their users.

- **Unauthorized access to stored conversations within Alexa:**
  Developers can potentially hijack the device and manipulate how the Alexa Voice Service is installed on the hardware. This can potentially allow them to gain unauthorized access to the user's Amazon account details, leading to their private information being compromised.

- **Unauthorized transactions from skills:**
  Similar to the previous attack, malicious developers can hijack the AVS to tamper with how the user interacts with their skills. Important skills such as bank account management skills or other skills that

require the user to make transactions can allow the hacker to tamper with the hardware to allow them to create unauthorized transactions from the user's account.

These kinds of attacks are already potentially possible within a standard Amazon Echo device. However, these are more likely to happen on a third-party device due to the security measures and practices that Amazon implement which make it less likely for the Echo device to be vulnerable to such exploits.

6. Conclusions

In conclusion, the rise in usage of Alexa-enabled speakers and other voice-first devices has increased the awareness of several privacy issues among users and developers alike. Upon analyzing and researching these possible issues, there are several sources of potential exploits and vulnerabilities but ultimately it appears to be well-designed and secured with no huge oversights within the security of the overall design.

Initially, we anticipated more issues around Alexa Skills, however, we generally found the feature to be carefully designed and not prone to easily exploitable privacy attacks. However, we predict that third-party hardware running the Alexa Voice Service API will be not as well designed and prone to security issues. We also expect some more sophisticated attacks on the Echo firmware.

We have given a brief overview of the potential attack landscape within the Alexa ecosystem and with the increase in popularity of speech recognition technology, it would be not surprising to have new and more complex security problems appear in the foreseeable future.

7. Bibliography

[1] I. Clinton, L. Cook, and S. Banik, "A Survey of Various Methods for Analyzing the Amazon Echo."

[2] W. Enck *et al.*, "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones," in *Communications of the ACM*, 2014, vol. 57, no. 3, pp. 99–106.

[3] D. Davidson, B. Moench, S. Jha, and T. Ristenpart, "FIE on Firmware: Finding Vulnerabilities in Embedded Systems using Symbolic Execution," *Proc. 22nd USENIX Secur. Symp.*, pp. 463–478, 2013.

[4] M. Xu *et al.*, "Toward Engineering a Secure Android Ecosystem," 2016.

[5] T. Vidas, E. C. E. Cylab, D. Votipka, I. N. I. Cylab, and N. Christin, "All Your Droid Are Belong To Us : A Survey of Current Android Attacks," 2011.

[6] A. Alhadlaq, "Privacy in the Amazon Alexa Skills Ecosystem," in *PETS*, 2015.