Life After Certification: Content Changing Attacks

Shivam Pandit
pandit@clemson.edu

Christin Wilson
cwils28@clemson.edu

I. Abstract

We initially proposed to reproduce the LipFuzzer work. Previous work has been done to examine the attacks due to the semantic misinterpretations. Due to the lack of novelty in producing more rules similar to the ones the authors released, we proposed content changing attacks. These are the attacks caused by the lack of re-certification requirement by Amazon Alexa team when the code is changed in the backend. We further explain 2 consequences of such attacks and demonstrate these attacks by deploying skills on the skill store to mimic real world environment and the vulnerability.

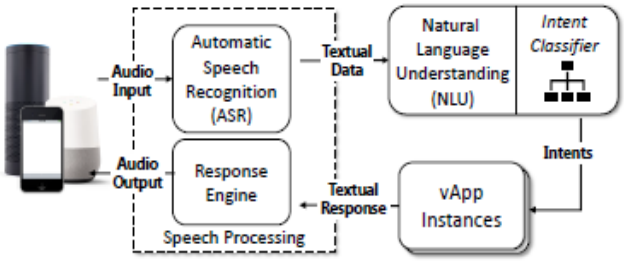

Fig. 1. VUI based VA System Architecture

II. Introduction

Voice user interface(VUI) based systems are getting more popular due to advent of smart devices. The major attraction here is voice based computational tasks that saves users time and is also convenient. Popularity of these systems can be assumed from the fact that there are around 30,000 plus vApps currently on Alexa store. However, due to its popularity it gained other attention as well like from intruders and malicious developers that leverage its acoustic based functionality to perform malicious tasks. Some attacks can be even done by sounds that would be inaudible by human ears making it a major cause of concern.

There are 2 major concerns related to security in Voice Assistants:

1) Semantic Misinterpretations: VAs are found to misinterpret user commands due to the incorrect speech understanding that can cause security concern.
2) Attacks on ASR & Intent Classifier: Attacks on Automatic Speech Recognition and intent classifier are big concern as attackers leverage common spoken errors to breach vApp integrity for malicious intent. Fig. 1. shows a typical Voice user interface based Voice Assistant system.

Most of the previous work focused on Automatic Speech Recognition component(ASR) but is was found later that it is actually intent classifier that contributes more to the misinterpretations by the voice assistants. So, our work will be focused on intent classifier. User voice commands can be misinterpreted by NLU intent classifier to give undesired results. ASR and intent classifier are both proven to misinterpret the spoken command by users that can be leveraged by hackers to intrude privacy. Developers can maliciously modify intent matching process in NLU. Intent classifier plays more important role since it is last step of the interpretation process. Even intent classification trees made as a result of text-to-intent conversion by intent classifier is prone to intrusion where malicious developer can add one more leaf node in the tree to invoke some malicious vApp other than the one the user wanted to invoke.

We discovered that there are vulnerabilities in Alexa skill hosting scenario due to separate back-end and Alexa skill hosting. Most of the Alexa skill implementation happens in a way that there are Alexa skills with all the invocation names and intents defined in Alexa Skills Console and all the programming functionality coded in Lambda on AWS platform. These two ends are then connected and results in the desired app functionality. Even if some Alexa skill which needs database access, DynamoDB is used in the AWS which is also connected to the front end Alexa skill console platform. In all those connections endpoints are connected so that front and backend can communicate.

The major problem with voice assistant application is the screen less access which means you cannot verify to whom you are talking to. You invoke the voice assistant application like Alexa and Google Home and it replies back in voice. We can only hear the reply from the device but cannot assure if the reply is what we requested for. This leads to many attack consequences where a malicious application pretending to be actual application can steal user data or give unrelated data to the user.

III. Background

Our motivation for this project was lack of research in this topic and very few papers discussing the security aspects of the voice assistant applications. One paper discussed about the skill squatting attacks that was due to the Automatic Speech recognition(ASR) which led to the incorrect skills being

invoked and same could be used by malicious programmer to name there skills which sound similar to target skill resulting in a attack which user wont even notice easily since most of the skills work without need to see screen.

Second paper was based on the issues due to the intent classifier of the Natural Language Processing Unit of the architecture. In this paper it was found that more than 70 % of the misclassification. Then, research was carried toward the reasons for the misclassification by the intent classifier and Lipfuzzer was proposed as shown in Fig2 below.

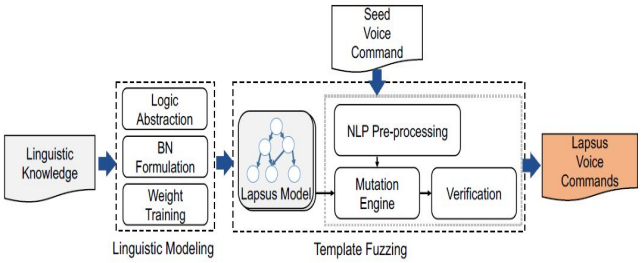

Fig. 2. Lipfuzzer Architecture

In Lipfuzzer Architecture shown above, we can see that it generated LAPSUS commands(any command that made Voice Assistant to misclassify the actual command). These LAPSUS commands were output from the Lipfuzzer model.

We first intended to extend Lipfuzzer by adding new rules and new modules that will generate more LAPSUS but due to lack of novelty and to show actual attacks from those LAPSUS was difficult so we changed our direction to attack scenarios in Voice Assistant applications. We found two of the attack scenarios, one where a kids skill application can have content changed to say non-sense commands to child thus affecting his emotional growth. Second scenario was to capture user data without users knowledge in Dynamodb of aws. These attacks were possible mostly due to the fact that Amazon separates Alexa Skill Console and backend that makes it possible for developers to change backend of the application without the application. The architecture of Voice Assistant Application Development Architecture looks as shown in Fig. 3 below.

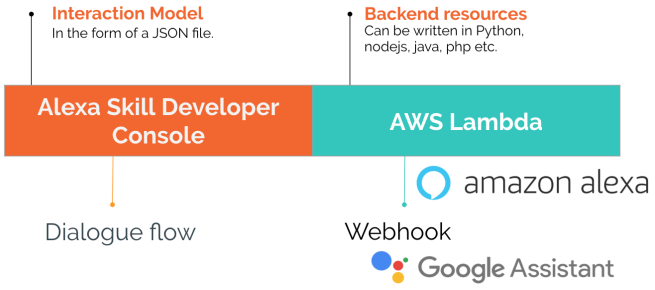

Fig. 3. Voice Assistant Application Development Architecture

Whenever new skill is hosted on Alexa skills platform, it has to go through Amazon Alexa certification which is reviewed by Amazon team if it is following all amazon's policy and finally it is published. Since, all the backend functionality is hosted on AWS Lambda including database like Dynamo Db which also is checked when the skill is hosted. Alexa skill intents can be manually coded in JSON Editor as shown in Fig. 4 below. Once the skill is published only changes made to the skill in Alexa developer console requires re-certification of the skill.

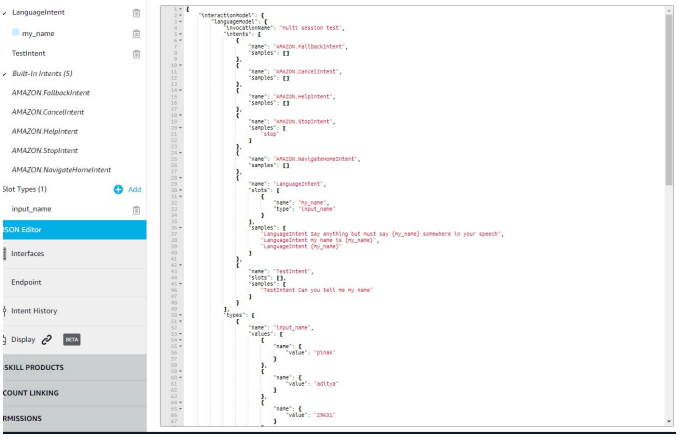

Fig. 4. Alexa Skill Console: JSON file of Skill

All intents can also be added using Alexa developer console tool that saves time from coding JSOn and gives user friendly user interface to specify the user intents or built in intents as shown in Fig. 5 below. Apart from that there are slot types and values that are used to transfer intent as shown in Fig. 5 below. These help in generating custom intents with associated voice commands like "Alexa send money to 'Alice'" where Alice is a slot value and can vary on different user utterances giving flexibility to developer in making a skill.

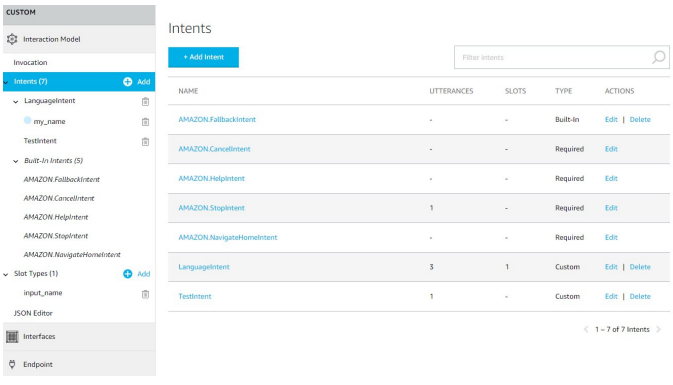

Fig. 5. Alexa Skill Console: Intents

IV. Related work

- **Attacking NLU with semantic misinterpretation. [?]**
  Yangyong et al presented in their paper "Life after Speech Recognition: Fuzzing Semantic Misinterpretation for Voice Assistant Applications" the semantic misinterpretation attack on voice assistants. They extended the work of Kumar et al by introducing the NLU part of the echo device in the picture. They demonstrated that it was the intent classifier and not the ASR that was causing the semantic misinterpretation attack in most cases.
  They also implemented a fuzzing model called Lipfuzzer that fuzzes an intent and provides as output several mutated voice commands based on the input intent. They then tested these mutated vice commands to check how prone the NLU is to misinterpret the voice commands.
  In the last part they also checked how many skills that were available in the skill store were prone to these attacks.

- **Attacking ASR through Acoustic Channels.**
  For audible voice command attacks, Diao et al. proposed to play prepared audio files using non-hidden channels that are understandable by a human listener. Tavish et al. [34] then presented Cocaine Noodles that exploits the difference between synthetic and natural sound to launch attacks that can be recognized by a computer speech recognition system but not easily understandable by humans.
  To achieve a better result, a white box method was used based on knowledge of speech recognition procedures. With complete knowledge of the algorithms used in the speech recognition system, this improved attack guarantees that the synthetic voice commands are not understandable by a human. For the threat model, Audible Voice Command Attacks require the attackers speakers to be placed at a physical place near victims devices ([19] fails with distances over 3.5 meters). Recently, Command Song was proposed to embed a set of commands into a song, to spread to a large amount of audience.
  More powerful attacks using inaudible voice commands were proposed in. They leverage non-linearity of microphone hardware used in almost all modern electric devices. Humans are not able to recognize sound with frequency over 20 kHz, and micro- phones upper bound of recordable range is 24 kHz. For this threat model, Inaudible Voice Command Attacks also require the attackers devices to be physically close to the victims devices (in feet range).

- **Attacking ASR with Misinterpretation.**
  Kumar et al. presented an empirical study of vApp squatting attacks based on speech misinterpretation. Moreover, as a concurrent work, also showcases a similar approach to exploit the way a skill is invoked, using a malicious skill with similarly pronounced name or paraphrased name to hijack the voice command meant for a different skill. Different from them, our work is the first to systematically explore the root-cause in NLU and Intent Classifier, and create the first linguistic-model-guided fuzzing tool to uncover significantly more vulnerable vApps in existing VA platforms.
  Nan Zhang et al. extended this work by performing two attacks namely voice squatting attacks and voice masquerading attack and provided extensive test results.

- **Analyzing the Privacy Attack Landscape for Amazon Alexa Devices**
  The research conducted by Raphael Leong focused to investigate the potential privacy issues that may emerge in the context of always-on speakers devices like Amazon Echo and Google home. They organized study papers and industrial research into specific categories which can lead to potential privacy issues. They analyzed the potential security and privacy issues surrounding Alexa skills, Echo firmware and third-party hardware that acts as alternatives to the Echo device. They evaluated and statically analyzed all the available Alexa skill repositories that were made public by developers on Github. They 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, they gave a concise overview of the current privacy attack landscape for Alexa devices and discussed about potential security and privacy issues with voice enabled devices that could arise in the near future.

V. Consequences

We are mainly considering three consequences of this attack.

- A skill can affect the children by providing inappropriate content.
- A skill can change Alexa replies to ask users for private data.
- An attacker who obtains access to a developers development console can alter the backend without their notice to perform unwanted operation on the users data.

VI. Implementation and Results

A. Attack Scenario 1

The content being delivered on skills meant for children can be changed and content that is not suitable for children can be given.

1) Kid's skills: A developer can specify in his developer console at the time of submitting for certification the category in which the skill belongs to. If it falls under the "Kids - Education Reference" an additional 'Parents permission required' check can also be provided which will require the parent to provide a consent on his/her Alexa app for the skill to enabled on the device before it can be used by the kids.

2) Freetime: Amazon describes free time as Amazon FreeTime on Alexa includes a new set of parental controls and family-focused features such as explicit song filtering, bedtime limits, educational QA, household communications, positive reinforcement for using the word please, disabled voice purchasing, and more . Freetime is a service meant for families with children which would provide better access controls for the children in the family. It is a paid subscription plan thats being offered to users for $2.99 per month. With he parent dashboard thats being offered with freetime, the parents can add content that they want the children to use with their Alexa device and the children are not able to add skills or make purchases while using the free time enabled Alexa device. A major catch with this method is that a device must be set apart for children use since free time is a mode that has to be enabled int he Alexa app. Thus, Once free time mode is enabled on a device, every user regardless of whether its a parent or child is not allowed to access the full functionality of the voice assistant. In order to use it normally the free time mode must be disabled. This can be problem in families which do not have a separate device for children use or where children access the Alexa device in every room in the house. Parent Dashboard providing Amazon freetime screen dumps is shown if Fig. 6 below.

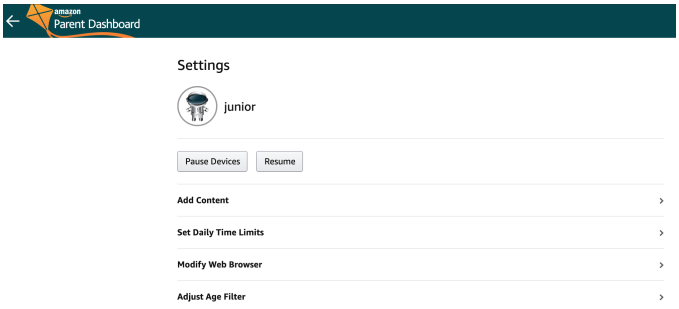

Fig. 6. Parent Dashboard provided with Amazon Freetime

Thus the benefits of including a free time membership include preventing children from accessing unwanted skills and also preventing semantic misinterpretation attacks. It also prevents children from making in-vApp purchases.

Regarding the content changing attack, we have demonstrated that an Alexa skill that the user wants his/her children to use can be added to the freetime enabled devices. The user can monitor and demo the skill before providing it to the children to use. During this time, the skill can have appropriate data that is meant for children. Once the skill has been added, if the content of the skill is changed to something inappropriate for children, this can lead to the children being delivered data that is inappropriate for their age. Since Amazon doesnt monitor this change in data, the children will continue to receive such data from the skill. Also, the parents wont be able to realize that this is happening unless they sit with the children while they are using it. The activities tab on the alexa device only shows the interaction by the user to the echo device and not what the device has given as response. Thus the parents wont be able to monitor it remotely and thus not realize that their children is encountering such an attack.

Thus we are bypassing a method that Amazon has introduced in their devices to stop children from accessing inappropriate data and only receiving data that their parents find is suitable for their children. We launch a skill that appears trustworthy in the beginning and misuses this trust to launch an attack on their children with their notice.

3) Attack model: We initially developed a simple fact skill called "Cool Kid Facts" that says simple facts to the user when the skill is invoked. This skill was included as a kids skill that required parental permission to use. We submitted the skill for certification and the skill was published to the skill store. Once the skill was published, we had appropriate data initially as shown in fig 7 below.

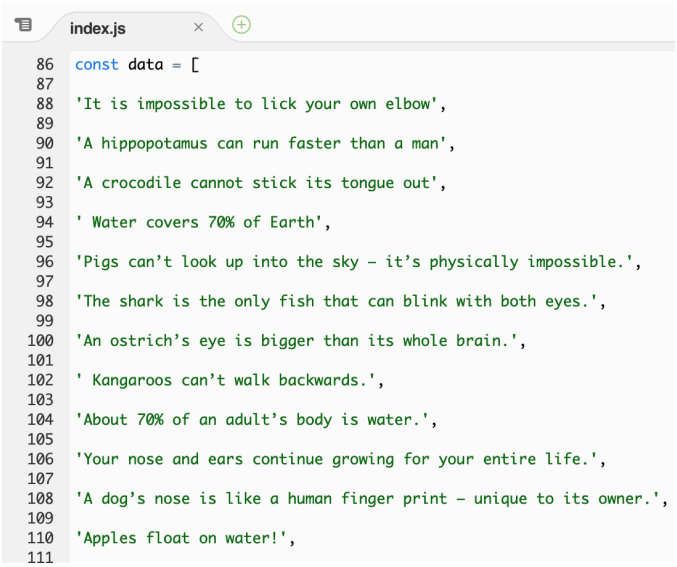

Fig. 7. AWS Lambda: Backend while publishing skill

Then we changed the content in the Lambda to something inappropriate for a child. We observed that the content changes in the live skill simultaneously. An example of the data can be seen in fig 8 below.

We kept changing the content for about a month with inappropriate data. This was done to check if Amazon monitors the changing content in skills. We never received a notification or flag from amazon regarding this though. We later added this skill to a freetime enabled device and then kept changing content to see if its reflected their as well simultaneously. Therefore we successfully bypassed Amazon's solution to protecting children from attacks on VAs by appearing as a trusted skill in front of the parents but then after the skill is enabled, it performed inappropriate behaviours with the children thus making the freetime control vulnerable to attacks.

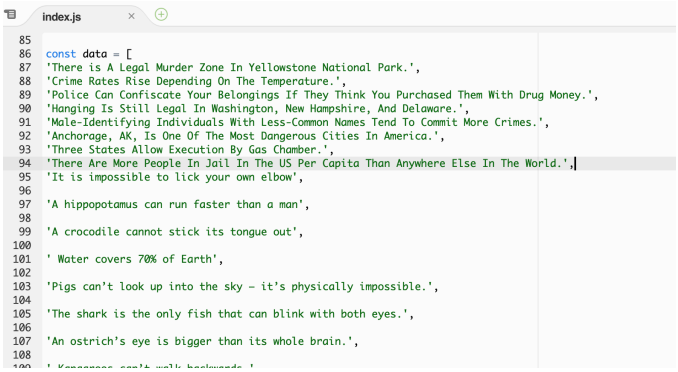

Fig. 8. AWS Lambda: Backend content changed after publishing skill

B. Attack Scenario 2

Here, we focused on *capturing important data from user through a story or a error message to do attacks.*

1) Custom Slots: In Alexa skills, the developer has to mention a slot where the user is expected to provide some information to the skill like a name, yes/no etc. The slot will be like a blank and be filled with the value the user provides. The predefined slots are of various types like USFirstNames, Date, number, Address, Country etc. In order to accommodate all kinds of values we will be using a custom slots for our skill. The possible values for this custom slot will be specified with word belonging to multiple categories namely names, numbers, places etc so that it recognizes all the kinds of answers. This will also ensure that no further changes have to be made in the interaction model after the skill is published in order to accommodate certain value requirements. A sample custom slot we made is shown in fig 9.

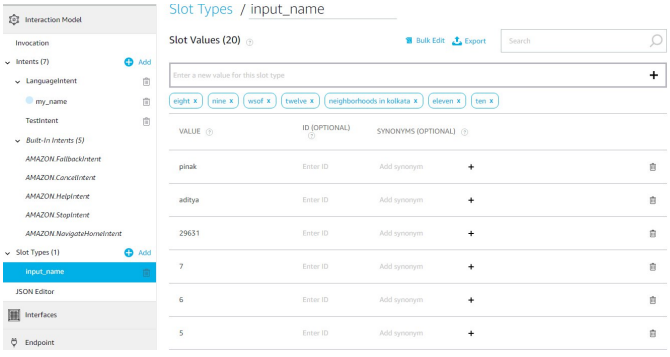

Fig. 9. Alexa Developer console: Custom Slot defined

2) Attack Model: Here we made an alexa skill that can interact with the user in a story way to make whole interaction intuitive. Alexa will be interacting with the user and asking some things that will be captured and saved in the database DynamoDb. In Fig. 10 below, we can see the lambda code written in Node js inside AWS developer console. Any changes made here didn't require to re-certify skill thus resulting in attacks. We can change code in Lambda multiple times while the skill is published that can do malicious things like capturing user data without users knowledge.

Fig. 10. AWS Lambda: Backend code of skill

Also all the user utterances to the alexa interaction are saved in DynamoDB in real-time thus attacker can capture full information from user masquerading as safe story telling skill application as shown in Fig. 11 below. To get users trust, attacker can publish a safe skill and make it run as safe application for some time after which lambda can be changed to do malicious behavior.

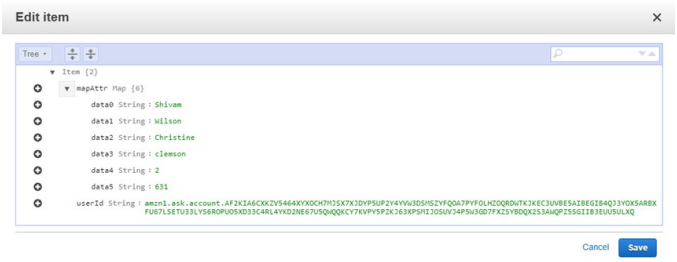

Fig. 11. Data being saved in DynamoDB

VII. Prevention

In order to prevent these sort of attacks, amazon should monitor the backend of the skills in regular intervals. Since the manual reviewing of the 90,000 skills in the skill store is not possible an automated analyses must be implemented. This analyses should compare the changes in code being made and the difference being made should be run through a sentimental analyses check which will determine if the content is of negative nature or positive nature. In case an inappropriate content is observed in the change in content, a flag must be raised and the skill should be sent for a re-certification. This will ensure a balance in usability for the developers since they wont have to rectify their skills after every change. Only in cases where there is suspicion of the content being

inappropriate should the re-certification being required which will not discourage the developers from making updates to their skills due to the requirement to rectify the skill after making changes.

VIII. Discussion

This is still a research topic and thus we do not have well documented results to prove the effectiveness of these attacks and how often this happens in the real world. The work we have done is a demonstration of the attack. To evaluate how often the developers change the content of their skills is impossible from our side due tot he large number of skills available in the skill store now. Also since the input and output are in the form of speech, the time required for each utterance is considerably high and thus it is not practical to evaluate this.One of the things we could do was develop a skill and publish it on the store. After the skill is published we can keep changing the content of the skill in the backend and observe if amazon is monitoring the change in content. After a while we can also add some inappropriate content in the skill and evaluate again. Another direction that we can take the project forward is by conducting multiple surveys. The participants of the survey can be both developers and users. The developers can be asked to give their opinion about how their reaction would be if amazon mandates a re certification to the skill after every change in content. This survey can help in learning if the developers would be discouraged or not to update the content of their skills in order to keep it interesting to the users. Also we can survey existing developers to see how often they make changes in the backend for their skills. The users can be surveyed by creating a skill that would collects sensitive information from users. The users would be made to interact with the skill and the backend utterances would be changed according to the users response. With this survey we can observe the amount of data the user is willingly providing to the skill. Also the usage of free time in families with children can be observed. If the user does use free time, the survey can also ask the parent for how often he/she monitors the content thats being delivered to the children.

IX. Conclusion

We initially began our class project planning to reproduce the Lipfuzzer which performs fuzzing to identify possible semantic misinterpretations. We later altered our path to demonstrate and conduct a content changing attack. We considered 2 consequences of this attack, namely the attack on children by delivering inappropriate content and the attack which captures sensitive information from users by cleverly changing what the skill does in order to manipulate the user and subconsciously request information from the user and deployed skills on the Alexa skill store which would perform these attacks. Additional surveys have to be conducted in the future to evaluate the effectiveness of these attacks and how an intervention by Amazon against this sort of attack will affect the usability of Alexa skill development as a whole. We also proposed a solution to prevent this attack from happening while at the same time balancing the ease of updating the content by the developers to keep their skills interesting.

X. Team Member Contributions

We first implemented existing implementation that was based on Lipfuzzer code by the authors of our base paper. That part was done by both of us. The second part of implementing something new required brainstorming sessions where we tried to figure out what are the possible things that can be changed or innovated here in addition to existing implementation. We thought of adding new rules that can generate more LAPSUS commands to misclassify more user commands. But due lack of novelty, we have to focus on attack scenarios.

1) Shivam Pandit: Majorly worked on attack scenario to capture user data and save it in DynamoDb thus leading to possible attack by stealing user senstive data without users knowledge.

2) Christin Wilson: Majorly worked on researching about the possible security attacks and vulnerabilities on VAs. Published skills on the Alexa store and worked on the attack scenarios.

References

[1] Zhang, Yangyong, Lei Xu, Abner Mendoza, Guangliang Yang, Phakpoom Chinprutthiwong, and Guofei Gu. "Life after Speech Recognition: Fuzzing Semantic Misinterpretation for Voice Assistant Applications."

[2] Kumar, Deepak, 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), pp. 33-47. 2018.

[3] Zhang, Nan, Xianghang Mi, Xuan Feng, XiaoFeng Wang, Yuan Tian, and Feng Qian. "Understanding and mitigating the security risks of voice-controlled third-party skills on amazon alexa and google home." arXiv preprint arXiv:1805.01525 (2018).

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

[5] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, Hidden voice commands. in USENIX Security Symposium, 2016, pp. 513530.

[6] W. Diao, X. Liu, Z. Zhou, and K. Zhang, Your voice assistant is mine: How to abuse speakers to steal information and control your phone, in Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones Mobile Devices. ACM, 2014, pp. 6374.

[7] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, Dolphinattack: Inaudible voice commands, in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017, pp. 103117.

[8] Haack, William, Madeleine Severance, Michael Wallace, and Jeremy Wohlwend. "Security analysis of the Amazon Echo." Allen Institute for Artificial Intelligence (2017).

[9] Leong, Raphael. "Analyzing the Privacy Attack Landscape for Amazon Alexa Devices."

[10] Alexa freetime "https://www.amazon.com/b?ie=UTF8node=17738015011"