Measuring the Functionality of Amazon Alexa and Google Home Applications

Jiamin Wang

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master

in

Computer Science and Application

Gang Wang, Chair

Bimal Viswanath

Hongliang Xin

January 29, 2020

Blacksburg, Virginia

Keywords: Natural language processing, CNN, active learning, RAKE, security

Measuring the Functionality of Amazon Alexa and Google Home Applications

Jiamin Wang

(ABSTRACT)

Voice Personal Assistant (VPA) is a software agent, which can interpret the user's voice commands and respond with appropriate information or action. The users can operate the VPA by voice to complete multiple tasks, such as read the message, order coffee, send an email, check the news, and so on. Although this new technique brings in interesting and useful features, they also pose new privacy and security risks. The current researches have focused on proof-of-concept attacks by pointing out the potential ways of launching the attacks, e.g., craft hidden voice commands to trigger malicious actions without noticing the user, fool the VPA to invoke the wrong applications. However, lacking a comprehensive understanding of the functionality of the skills and its commands prevents us from analyzing the potential threats of these attacks systematically. In this project, we developed convolutional neural networks with active learning and keyword-based approach to investigate the commands according to their capability (information retrieval or action injection) and sensitivity (sensitive or nonsensitive). Through these two levels of analysis, we will provide a complete view of VPA skills, and their susceptibility to the existing attacks.

Measuring the Functionality of Amazon Alexa and Google Home Applications

Jiamin Wang

(GENERAL AUDIENCE ABSTRACT)

Voice Personal Assistant (VPA) is a software agent, which can interpret the users' voice commands and respond with appropriate information or action. The current popular VPAs are Amazon Alexa, Google Home, Apple Siri and Microsoft Cortana. The developers can build and publish third-party applications, called skills in Amazon Alex and actions in Google Homes on the VPA server. The users simply "talk" to the VPA devices to complete different tasks, like read the message, order coffee, send an email, check the news, and so on. Although this new technique brings in interesting and useful features, they also pose new potential security threats. Recent researches revealed that the vulnerabilities exist in the VPA ecosystems. The users can incorrectly invoke the malicious skill whose name has similar pronunciations to the user-intended skill. The inaudible voice triggers the unintended actions without noticing users. All the current researches focused on the potential ways of launching the attacks. The lack of a comprehensive understanding of the functionality of the skills and its commands prevents us from analyzing the potential consequences of these attacks systematically. In this project, we carried out an extensive analysis of third-party applications from Amazon Alexa and Google Home to characterize the attack surfaces. First, we developed a convolutional neural network with active learning framework to categorize the commands according to their capability, whether they are information retrieval or action injection commands. Second, we employed the keyword-based approach to classifying the commands into sensitive and nonsensitive classes. Through these two levels of analysis, we will provide a complete view of VPA skills' functionality, and their susceptibility to the existing attacks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to Voice Personal Assistant

Voice Personal Assistant (VPA) is a software agent, which can interpret the user's voice commands, and respond with appropriate information or action. The primary VPAs on the market are the Amazon Alexa, Apple Siri, Google Home and Microsoft Cortana. The VPA is the passive listening device, users need the wake word to trigger the VPA, like 'Hey Siri' or 'Alexa'. Amazon and Google allow the third-party developers to build new applications (called skills in Amazon Alexa and actions in Google Home), more features of VPA can be added [1] by those applications. Therefore, users can talk with the VPA to complete various tasks, such as play music, order coffee, control smart home appliance, make a phone call, and so on. I will only use "skills" to represent the applications of both Amazon Alexa and Google Home stores for convenience.

Most VPAs such as Amazon Alexa and Google Home use a cloud-based model to host skills and interact with users. The workflow of how Amazon Alexa process a voice command

is shown in Figure 1.1. After the user talks to the VPA device, Amazon Echo, the voice command is first sent to the Amazon/Google cloud. The cloud needs to interpret the natural language commands into API call, and then sends the API call to the corresponding skill server. The third-party developers can either host their skill server directly within the cloud, or run the server separately. No matter where the skill server is, the skills generally depend on the cloud to parse and interpret the voice command and route API calls. If the voice command is used to operate the smart home devices, the cloud or the skill server will send out the request to the smart devices to perform tasks. The smart devices return with the operation status to the skill server. Then the response from skill server will send back to the users. Although users communicate with skills using voice commands, the audio is not accessible to the developers. Instead, all voice commands are translated and sent to the developers as text [2].

There are some skills that require account linking. For example, for the first time using Starbucks skill, it will navigate the users to connect their Starbucks account with this skill. Then the users are able to interact with the skills to order the drink in their own account. Both Amazon Alexa and Google Home employ the OAuth to connect users' accounts with the corresponding skills in VPA.

1.2 Motivation

Although this new technology has interesting and useful features, they also pose new privacy and security risks. Researchers recently uncovered that the vulnerabilities exist in the VPA ecosystems. The natural language processing system in the cloud is the main target of the attacks. For example, Kumar et al. show that users can incorrectly trigger malicious third-party skills, whose name has similar pronunciations to the user-intended skills [3].

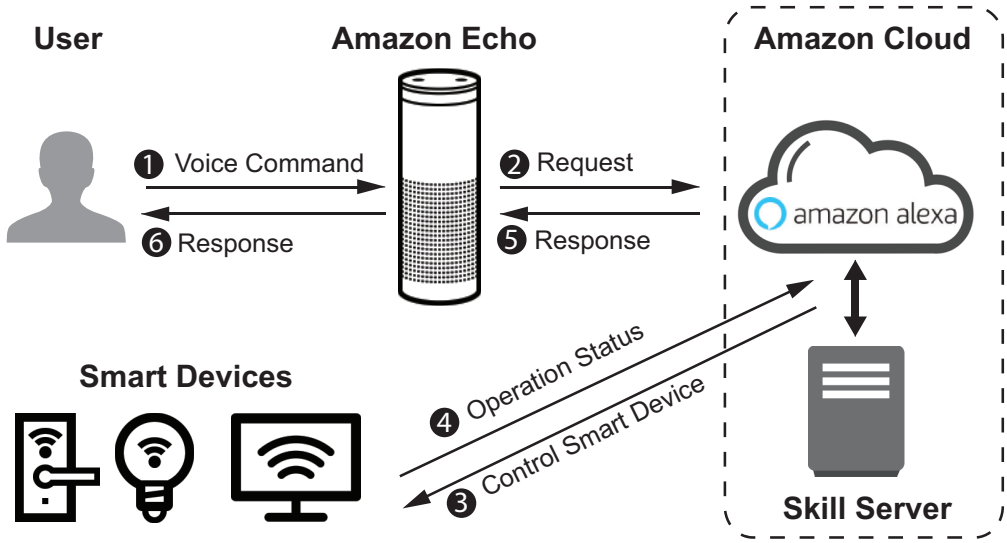

Figure 1.1: Schematic of how Amazon Alexa process a voice command.

Recent news reported that the audio from TV or noise is able to invoke the undesirable actions [4]. The existing researches have focused on proof-of-concept attacks by pointing out the potential ways of launching the attacks, like users improperly invoke the malicious skills or launch the unwanted actions accidentally. There are two primary proof-of-concept attacks have been reported so far that can exploit the VPA systems, hidden voice command attack and skill squatting attack.

Hidden Voice Command Attack: The attackers can send hidden voice commands like the noise from the TV or the inaudible voice that are beyond human's hearing range to surreptitiously trigger the malicious skills [5, 6]. But the inaudible voice can be easily filtered out. As shown in Figure 1.2, the hidden voice command "Alexa, ask Scout to disarm," that triggers the unintended actions, disarms the home security system, without user's notice.

Skill Squatting Attack: The attackers can construct a malicious skill, with a name that sounds similar to the targeted skill. As shown in Figure 1.3, the attacker develops the skill with the name "Sdarbukcs," when the user tries to invoke the desired skill, "Starbucks", the VPA cloud may misinterpret the natural language. Therefore the user is routed to the

malicious skill. Then the malicious skill interacts with the user and collects the sensitive information, like the PIN code, password, et al. By knowing these sensitive information, attackers make the transaction in user's Starbucks account, which causes Alexa owners to suffer from a financial loss.

However, there is no clear idea about what is the consequence caused by these attacks. The lack of a comprehensive understanding of the functionality of the skills and its commands prevents us from investigating the potential consequences of these attacks systematically. In this project, we carry out the extensive measurements on the third-party applications from Amazon Alexa and Google Home stores to assess the attack surfaces. For a given VPA skill, we aim to characterize its risk level by detecting and analyzing its sensitive commands that are subject to the potential attacks.

There exists one key challenge for automatically analyzing the functionality of VPA skills. Unlike the mobile phone apps, whose source codes are accessible for analysis. The skills are web programs that are hidden behind the cloud. Therefore, we cannot characterize the skills by the traditional API analysis but the new tools are required to analyze the natural language interface. In order to analyze the skills' commands, we develop two classification models to analyze the capability and sensitivity of the commands, respectively. We firstly developed a natural language processing framework to classify the given commands into two classes based on their capability: action injection and information retrieval. The action injection command is to ask the VPA skills to perform some actions, like "Alexa ask Classic Jazz to Play". The information retrieval command is to collect information from users like "What is the target price of eBay." Secondly, we further classify the commands into sensitive or non-sensitive classes. If a command involves collecting sensitive information, such as password, PIN, email content, or may pose a security risk if it is operated secretly, we consider this command as a sensitive command. The examples for the sensitive commands are "Alexa,

Ask Climax Security to disarm," "Alexa, Ask Newton who just mailed me."

Figure 1.2: Example of Hidden Voice Command Attack

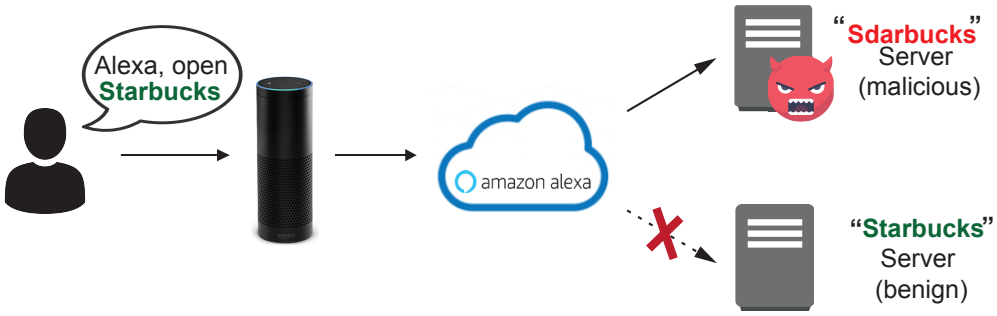

Figure 1.3: Example of the Skill Squatting Attack

From these two levels analysis, we identify the sensitive information retrieval and sensitive action injection commands. The reason why we perform capability analysis is that the two different capabilities of the commands are corresponding to these two types of attacks. The action injection is related with the hidden voice command attack. Injection attack is a major problem in web security which allows the untrusted input to a program. It is listed as the number-one web application security risk. Once the VPA skill secretly executes the sensitive action injection command, the skills will blindly send these voice commands to the external devices and the status of the device system is altered without users' acknowledgement, e.g., the status of the home security system is changed after receiving the command "disarm the home security system". The sensitive information retrieval is corresponded to the skill squatting attack, which mainly collects the sensitive information. The adversary builds the malicious skills to collect the sensitive information, and/or respond the wrong information

to the users. Through these two levels of analysis, we will provide a complete view of VPA skills, and their susceptibility to the existing attacks.

1.3 State of art of NLP application in enhancing security and privacy

NLP has a wide application in enhancing security and privacy. For example, Tian et al. devised a new technique Smartauth, NLP was applied to extract the functionality described in the IoT App advertise automatically. The gap between this described functionalities and true capabilities was minimized by Smartauth, in order to avoid overprivileged apps and enhance security and privacy in the home-automation ecosystem [7]. Kumar et al. demonstrated that skill squatting attack, by exploiting intrinsic error within the opaque NLP, can invoke malicious skills instead of targeted ones in Amazon Alexa [3]. Harkous et al. [8] introduce Polisis, an automated framework for privacy policy analysis, that utilizes the NLP to build domain-specific word vectors, which were used to train neural network classifier. The Polisis's scalable queries make privacy policies much more reader-friendly. Liu et al. [9] present that word2vec embeddings pre-trained on Google News dataset were applied to train the neural networks to model the vagueness of words in privacy policies. NLP is exerted to filter out unrelated text fragments and allowing crowdworkers to focus on the relevant content. By combining NLP, machine learning, and crowdsourcing, Sadeh et al. develop algorithms that semi-automatically extract and summarize key privacy policy features from privacy policies [10]. Du et al. [11] propose flowcog to analyze flow leaks in Android apps, using NLP method to determine whether extracted flow-specific semantics are consistent with the given flow. WHYPER [12] employs the NLP technique to identify sentences from the description of the usage of permission, to bridge the semantic gap between the true

functionality of applications and users' expectations. Sun et al. [13] develop ReGenerator, a system automatically generate human-readable reports of malware analysis based on antivirus vendors' published reports. Nan et al. [14] present UIPicker, to identify sensitive user inputs within Android apps. NLP was utilized to extract and reorganize selected layout resource texts. Liao et al. [15] presented iACE, a system uses NLP to automatically extract key attack identifiers from unstructured text.

Chapter 2

Computational methods

2.1 Computational Methods

2.1.1 Word2vec embedding

Word embedding is to use the vector to represent a particular word. It is able to capture the semantic and syntactic similarity of a word and context. There are various word embedding models such as word2vec(Google), Glove(Stanford) and fastest(Facebook). Word2vec is one of the most popular techniques to learn word embedding using a shallow two-layered neural networks. It was developed by Tomas Mikolov [16] in 2013 at Google. The word2vec model takes the corpus of text as input and generates vector space, each word in the corpus is associated with a vector in the space. The word vectors are closer to each other if the words have a similar meaning in the corpus. There are two architectures used by word2vec model: Skip-Gram and Continuous Bag of words (CBOW).

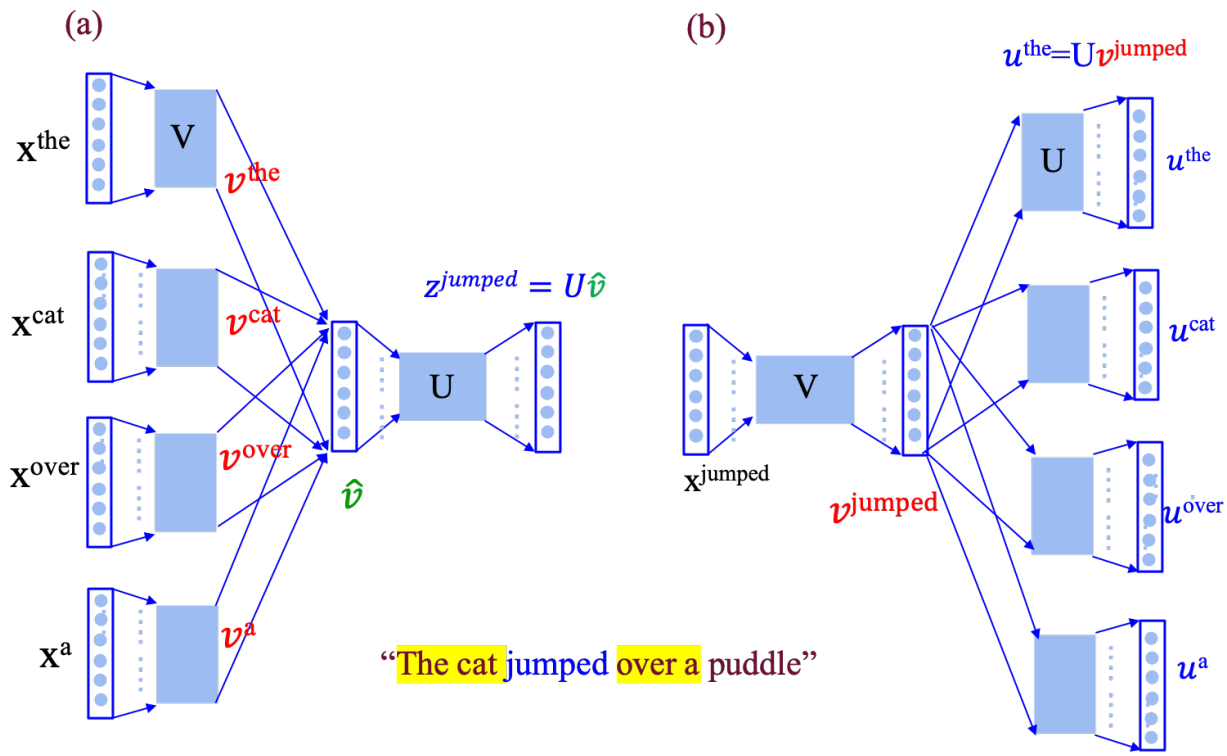

Figure 2.1: The two architectures used by word2vec (a) CBOW and (b) Skip-Gram.

Continuous Bag Of Words (CBOW)

The CBOW model predicts the center word based on the surrounding context. For example, in a simple sentence, 'The cat jumped over a puddle,' the 'jumped' is chosen to be the target word, if we consider a context window of size 2, the context 'the cat' and 'over a' is used to predict the target word, as shown in Figure 2.1 (a). **One-hot vector** is used to represent all the words with all zero values and only a 1 at the index of the word. In the above example, the word 'jumped' can be illustrated by 5×1 vector $x^{jumped} = [0, 0, 1, 0, 0]$ and $|x| = 5$ is the size of the corpus. The hot word vectors for the context is

$$[x^{the}, x^{cat}, x^{over}, x^{a}]^T = [[1, 0, 0, 0, 0], [0, 1, 0, 0, 0], [0, 0, 0, 1, 0], [0, 0, 0, 0, 1]]^T. \tag{2.1}$$

The embedded word vectors for the context is that the hot word vectors multiply by the weight matrix $v^i = Vx^i$, $[v^{the}, v^{cat}, v^{over}, v^a]^T$. The average of the embedded word vectors is

$$\hat{v} = \frac{v^{the} + v^{cat} + v^{over} + v^a}{4} \tag{2.2}$$

The averaged vectors are processed by another weight matrix U, we obtain the score vector $z = U\hat{v}$. The softmax activation function applies to the score vector resulting in the probabilities, $\hat{y} = softmax(z)$. The goal is to minimize the difference between the true probabilities y and $\hat{y}$ by tuning the two weights matrices V and U. The cross entropy $H(\hat{y}, y)$ is used to represent the objective function to measure the distance/loss:

$$H(\hat{y}, y) = -\sum_{j=1}^{|v|} y_i log(\hat{y}) \tag{2.3}$$

The optimization algorithm gradient descent is used to minimize the objective function. The one-hot vector of each word multiplies with weight matrix V receives it own word embedding.

Skip-Gram

The Skip-Gram model implements the opposite function of the CBOW model. It predicts the context words for a given target word. We still use the above "The cat jumped over a puddle" as example. By given the center word 'jumped', the Skip-Gram model predicts the context words ('the', 'cat', 'over', 'a'), as shown in Figure 2.1 (b). The procedure to generate the Skip-Gram model is similar to CBOW. The input is the one-hot word vector of the 'jumped', $x^{jumped} = [0, 0, 1, 0, 0]$ and the output word vectors are the one-hot vectors

$$[y^{the}, y^{cat}, y^{over}, y^a]^T = [[1, 0, 0, 0, 0], [0, 1, 0, 0, 0], [0, 0, 0, 1, 0], [0, 0, 0, 0, 1]]^T. \tag{2.4}$$

The embedded word vector is the input word matrix V times input word vector, $v^{jumped} = Vx^{jumped}$. The score vector u is output word matrix U multiply the v^{jumped}, $u = Uv^{jumped}$. The softmax activation function applies to the score vectors gives the probabilities $y = softmax(u)$. The word vectors generated from this Skip-Gram model $[\hat{y}_{the}, \hat{y}_{cat}, \hat{y}_{over}, \hat{y}_a]^T$ should match the true probabilities $[y^{the}, y^{cat}, y^{over}, y^a]^T$ by tuning the matrices U and V.

Compared with the CBOW approach, the Skip-Gram works well for a small amount of the training data, represents well even rare words or phrases. The CBOW is a few times faster to train than the Skip-Gram, moderately better accuracy for the frequent words.

Negative Sampling

The neural networks training is the process to tune the model weights to minimize the difference between model prediction and datasets output value. However, the word2vec model has an enormous number of weights, and all the weights need to be updated by every pair of training dataset, which is computationally demanding and slow down the training. Negative sampling solves this problem by only updating a small portion of the weights, instead of all of them.

2.1.2 Active Learning

We used active learning in the commands capability analysis. For most machine learning tasks, data labeling requires expensive manual efforts and can be very time-consuming. Active learning is a method that can help to train an accurate model with a small set of training data [17]. The idea is to select data samples that contribute the most to the training process and send them to human annotators to perform labeling. In this way, significant manual efforts can be saved. The key step of active learning is sample selection. There are

different types of query/sampling algorithms, including uncertainty sampling [18], query by committee [19] and pool-based sampling [20]. We mainly use uncertainty sampling, margin sampling and entropy sampling to select the unlabeled samples. In the classification problem, the margin sampling selects samples where the margin between the two most likely classes is small, that is, the classifier is struggling to distinguish between these two most likely classes. The entropy sampling queries the unlabeled samples that maximize the posterior probabilities of each class by the classifier.

After data labeling, the newly labeled data will be added to the training dataset to retrain the machine learning model. The whole process repeats until achieving convergence. In this work, we need to analyze tens of thousands of voice commands from Amazon Alexa and Google Home. Labeling all these commands is a seemingly impossible task. Therefore, applying active learning by only labeling most informative commands to reduce human efforts.

2.1.3 Rapid Automatic Keyword Extraction(RAKE)

We employed a rapid automatic keyword extraction method(RAKE) in the risk analysis. Keyword extraction is a fundamental step in text mining and natural language process. RAKE is an algorithm to quickly and automatically extract keywords from a document[21]. In English, keywords usually include one or more words, but barely contain punctuation or stop words, such as "or," "am," "an." RAKE algorithm first utilizes the phase/word delimiter and stop word as a separator to divide the document into candidate keywords. Then the significance of the candidate keywords is measured by

$$\text{keyword score} = deg(w)/freq(w) \tag{2.5}$$

where *deg(w)* is the degree of a word which defines how many times the keyword w co-occurs with other words; *freq(w)* is the frequency of word w. Finally, the candidate keywords with the top score are selected as keywords. RAKE has been proved to be a computationally efficient and accurate keyword extraction method. We will use RAKE for keyword extraction in our model design.

The term frequency-inverse document frequency (TF-IDF) is another commonly used approach for keyword extraction. It calculates the word frequency (TF), the normalized frequency of a word appears in a document, and inverse document frequency (IDF), the logarithm of the total number of documents divided by number of documents that contains a word. The TF-IDF score is the multiplication of TF and IDF. Compared with TF-IDF, the RAKE algorithm disfavors the word that appears too frequently, as the RAKE is inverse proportional to the $freq(w)$ as shown in Equation 2.5. RAKE also disfavor the word that is not exists in long phrase. In the keyword extraction from the commands, the word "ask", "what", etc. will not be popped out, as they appears too often and not come along with the long phrase.

2.1.4 Convolutional Neural Networks (CNN) for NLP

CNN is one type of deep learning algorithm and the full CNN architecture includes a convolutional layer, pooling layer and fully-connected layer. It has a wide application in computer vision. The CNN models have also achieved an excellent success for various NLP tasks [22], with the vector matrix representing the text as the input.

The schematic of how CNN classifying the sentence is shown in Figure 2.2. The dimensionality for one word vector is chosen to be 150, so the vector matrix for the sentence "ask invkname for the latest news" is 6×150. We have two the convolution filters with size 2 and

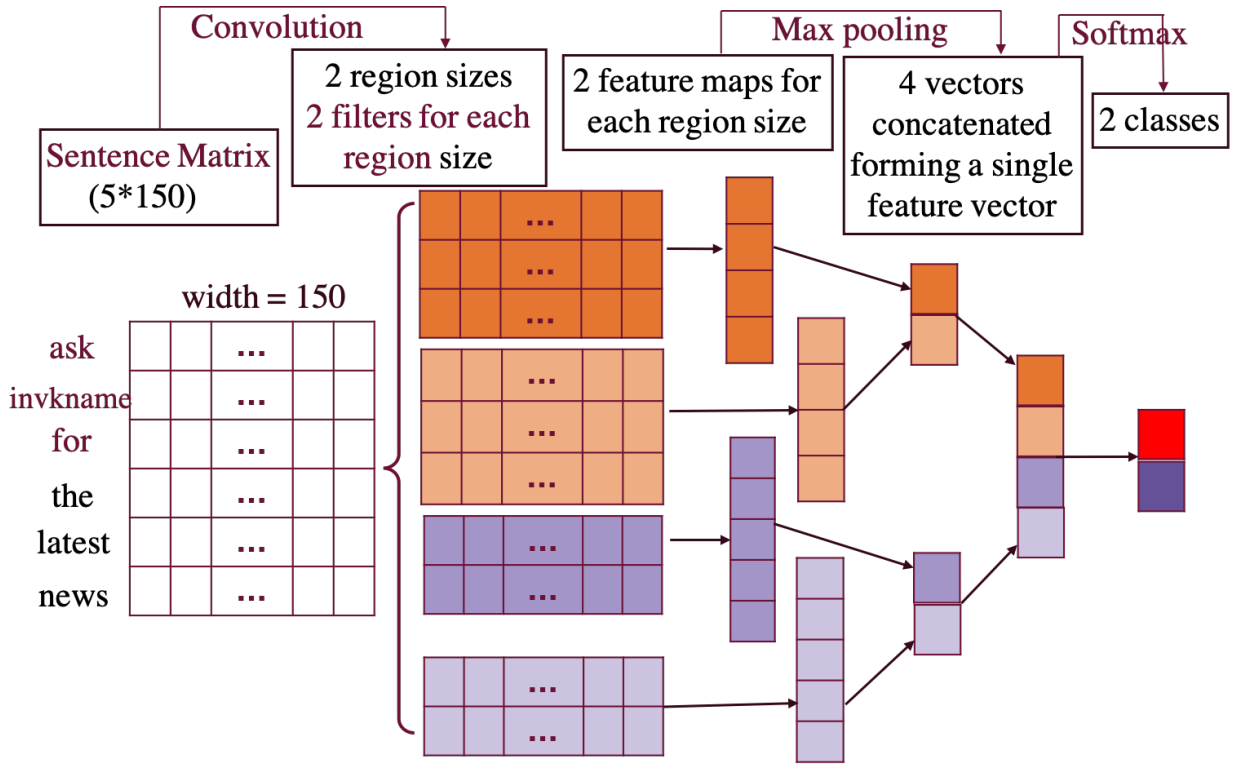

Figure 2.2: Schematic of CNN architecture for sentence classification. We have the vector matrix (6×150) for "ask invkname for the latest news." Two filter region sizes 2,3 are applied to the matrix, each region size has 2 different filters. The 4 feature maps are obtained after the filter. The max pooling applies to the feature maps to extract the 4 maximum values from the feature maps. Then the extracted features are concatenated forming one feature vector. The feature vector fed into the fully connected layer and activation function to classify the commands.

3. The sliding window with size 2 or 3 is moving through the vector matrix, then the filter is applied to each window. The value in the sliding window multiplies the filter resulting in one scalar value. For example, the first window with width 3 contains three word vectors $w^{ask}, w^{invkname}, w^{for}$

$$x_1 = [w^{ask}, w^{invkname}, w^{for}]^T \in R^{3\times150} \tag{2.6}$$

The convolution filter u with size three applies to this window, giving rise to a scalar value r_1:

$$r_1 = g(x_1 \cdot u) \in R \tag{2.7}$$

Four scalar values are obtained for filter with size 3 moving through the sentence, and these four scalar values form one feature map. The pooling layers are used after the convolution layer. The pooling layers subsample the inputs by reducing the dimensionality of the feature maps to extract the dominant features. The max pooling is the most commonly used approach to extract the maximum value in the feature maps. Thus, the pooling layers generate a fixed size output. The outputs from the pooling layers are concatenated forming one feature vector. The feature vector fed into the fully connected layer and softmax dense layer to classify the sentences.

2.2 Interpretation of performance measures

When labeling the commands, we employed 1 (positive) to represent the information retrieval command and sensitive command, and 0 (negative) to represent the action injection and nonsensitive command. The metrics Accuracy, Precision, Recall and F1 score are used to evaluate the performance of the classification model. The definition of these metrics are shown below:

	Predicted Class		
Actual Class		Class = 1	Class = 0
	Class = 1	True Positive	False Negative
	Class = 0	False Positive	True Negative

Table 2.1: Parameters used to calculated the performance measures.

True Positive (TP): It is the correctly predicted positive value which means that the value of the actual class is 1 (true) and the value of predicted class is also 1 (true). E.g., the command is information retrieval (class 1), and the model predicted class is the same as the actual class.

True Negative (TN): It is the correctly predicted negative value which indicates that the value of the actual class is 0 (false) and the value of the predicted class is also 0 (false). E.g., the command is nonsensitive (class 0), and the model predicted class is the same as the actual class.

False Positive (FP): The actual class is 0, but the model predicted class is 1. E.g., the command is action injection (class 0), but the model predicted class is information retrieval (class 1).

False Negative (FN): The actual class is 1, but the model predicted class is 0. E.g., the command is sensitive (class 1), but the model predicted class is nonsensitive (class 0).

Accuracy: Accuracy is the most intuitive performance measure. The definition is shown in Equation 2.8. It is the ratio of correctly predicted observations to the total observations. The Accuracy is a great measure only if the dataset is symmetric, i.e., the distribution between False positive and False negative is even. Otherwise, other parameters like Precision, Recall,

and F1 Score are used to evaluate the performance of the model.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.8}$$

Precision: The Precision is defined as the ratio of correctly predicted positive observations to the total predicted positive observations. This metric answers that among the commands that are predicted to be sensitive, but how many of them are actually sensitive.

$$Precision = \frac{TP}{TP + FP} \tag{2.9}$$

Recall: Recall is the rate of correctly predicted positive observations to the observations that are actually positive. E.g., among the actual sensitive commands, how many commands are predicted correctly.

$$Recall = \frac{TP}{TP + FN} \tag{2.10}$$

F1 score: F1 score is the harmonic mean of precision and recall, as shown in Equation 2.11. It takes False Positive and False Negative into account to balance the Precision and Recall. F1 gives a more precise measure than the Accuracy, especially if the dataset has uneven class distribution.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.11}$$

Chapter 3

Experiment Design

3.1 Commands Collection

We designed two crawlers using Selenium to collect the skill information from Amazon Alexa and Google Home stores in 2019. We adopted the breadth-first search to gather all existing skill IDs. For both two stores, we start from the home page, access each link hosted in the store domain and collect the skill IDs during the search. We accessed the corresponding introduction page by using the skill IDs afterward. The introduction page contains all the detailed information of that skill, such as skill name, developer information, user review, description of skill's capabilities, commands, skill access permission, account linking options etc [2]. We crawled all the accessible skills and the associated information for the further analysis.

Here we employed the Amazon Alexa as an example to give an overview of the distribution of Alexa skills. There are 23 categories in Amazon Alexa store: {Business & Finance, Communication, Connected Car, Education & Reference, Food & Drink, Games & Trivia,

Health & Fitness, Home Services, Kids, Lifestyle, Local, Movies & TV, Music & Audio, News, Novelty & Humor, Productivity, Shopping, Smart Home, Social, Sports, Travel & Transportation, Utilities, Weather}.

Among these categories, most skills in {Smart Home, Connected Car} categories require account linking. The skills like shopping services, banks, smart home devices, cars require the user to link the service account with the corresponding VPA skills. Many skills designed for entertainment and leisure like the {Movies & TV, Music & Audio, Novelty & Humor, Kids, et al.}, do not require the account linking. Totally, we collected 31,413 Amazon Alexa skills (80,129 commands) from 23 categories and 3,148 Google Home actions (9,096 commands) from 18 categories.

3.2 Commands labelling

After gathering all the commands, we manually label some commands to prepare for the machine learning model training and validation. In the capability classification, we use 0 to denote the action injection class and 1 to represent the information retrieval class. In the sensitivity analysis, 0 is used to represent the nonsensitive command and 1 stands for the sensitive command. Based on the developers' documentation, both Amazon Alexa and Google Home skill command usually follow a similar format:

(wake word i.e., "Alexa") + verb(start/open/launch/ask/begin/load/ + invocation name + ask skill to do something or ask skill to obtain some information (rest part of the command)

Only the last part of the command is the most meaningful, which indicates the command's functionality or user's intent. Because the commands from both two stores have the similar pattern, so we only label some of Amazon Alexa's commands, which are used to train and

validate the machine learning model, then the trained model is used to categorize the rest commands in Amazon Alexa and all the commands in Google Home.

In the capability classification model, classifying the skills into action injection and information retrieval, we randomly choose 1,810 Amazon Alexa commands as training dataset to train the model and another 475 commands for validation. A small portion of the training dataset is used to evaluate the model when tuning the model hyperparameters. The 475 validation commands is the actual test dataset, which provides the gold standard to evaluate the model. For the sensitivity classification model, categorizing the skills into sensitive or non-sensitive class, we randomly select 1,652 commands from Amazon Alexa for training and 275 commands for validation.

3.2.1 Commands cleaning

There are some commands only contain one verb, such as "resume," "stop," et al., and some skills have a very long invocation name, which would bias the machine learning model. Before the experiments, we preprocessed the commands using the following ways:

- **Delete invocation commands** There some commands can only invoke the skill, called invocation commands, the format of these skills follows the pattern:

  wake-word(Alexa) + trigger phase(start, open, launch...) + skill's invocation name (Starbucks, Aura Home, Amex) [23].

  These commands do not actually trigger other actions and they do not bring in security and privacy risks directly, which can be removed. We delete 29,784 invocation commands from Amazon Alexa and 2,949 invocation commands from Google Home.

- **Commands Structure Management**. As the invocation name cannot reflect the

skills' actual function, we use the general name *invk_name* to replace the invocation name. The trained model will not be biased by other meaningless information and only focus on the real features of the commands.

- **Adding category name**. The commands are typically brief, which provides little information in the model training and validation. However, we find that the commands usually have similar functionality in the same category, we add the category name at the end of the commands to make the commands more informative.

- **Duplication of the single word commands.** There are some single word commands, which only contain one word such as "stop," "help," "pause" and "resume." In the capability classification model, the work2vec was used to build the embedding layer, but the vector for the single word contains limited information and the model system is not able to confidently classify the single word commands. Therefore, we duplicate the occurrence of a single word. For example, we replace "help" with "help help," and "stop" with "stop stop."

- **Remove redundancy in the dataset**. Some skills have the same or similar commands. In the News category, more than 70% of the skills have the same commands "Alexa, what's my Flashing Briefing?" and "Alexa, what's in the news?". We exclude these replicated commands, as these commands would biase the model. The above mentioned command prepossessing can make the two different command become identical. For example, the commands "ask Doctor who Facts for a fact" and "ask Unofficial Stargate Facts for a fact" become identical after replacing the invocation name with *invk_name*. We also delete these commands to make the dataset integrity. We have removed 1,141 duplicated commands from Amazon Alexa and 296 duplicated commands from Google Home.

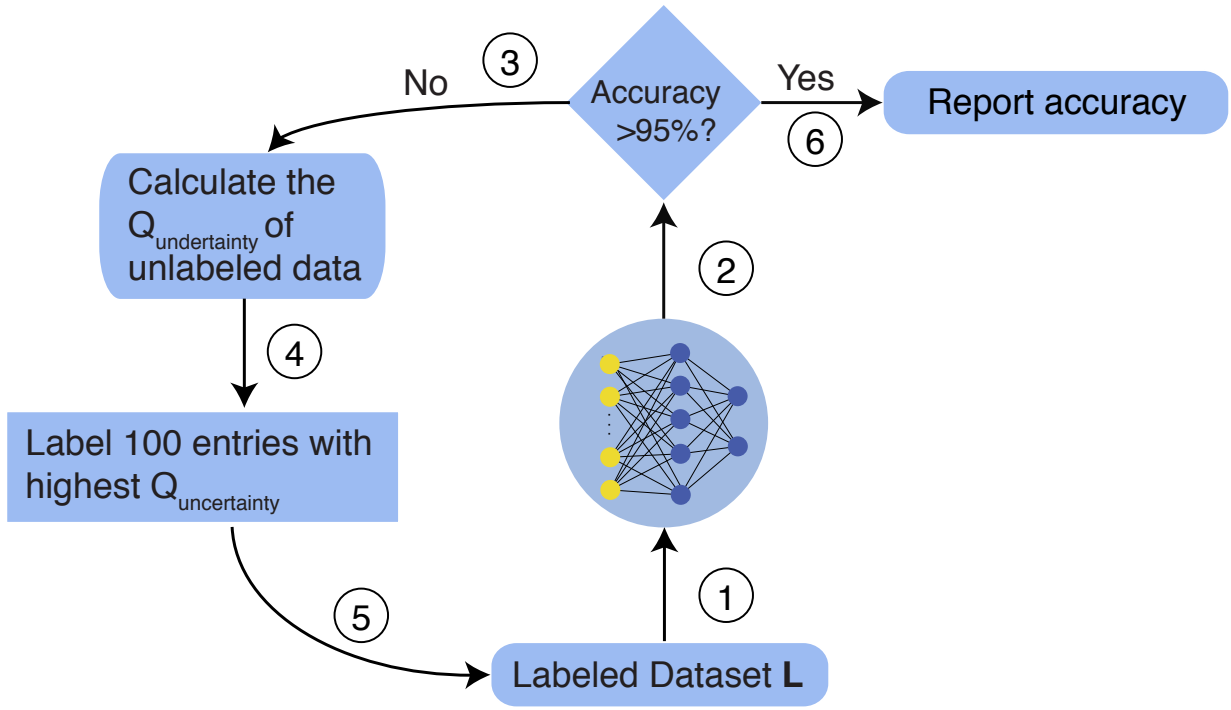

Figure 3.1: Workflow of the CNN model with active learning.

3.3 Experiment Design

For the given commands, we have performed classification analysis according to their capability and sensitivity, which provides deeper insights about the potential security and privacy risk caused by the attacks.

3.3.1 Capability classification Model

We designed the NLP framework using CNN with active learning aim to classify the commands into two categories according to their capabilities: action injection and information retrieval. We build the embedding layer by word2vec CBOW model in Gensim [24] using 80,129 commands. The window size, i.e., the maximum distance from the center word to the predicted word is 10. We set the dimensionality of the word vector is 150. The model

ignores all words with a total frequency of less than 10 and the negative sampling is used to simplify the calculation.

The open-source Keras library [25] backend by Tensorflow [26] is used to perform the CNN training and validation. In the CNN model, the embedding layer, spatial dropout, convolution layer, dropout, convolution layer, global max pooling, and drop out are stacked sequentially. The sigmoid activation function is used to do the binary classification, and we use the cross-entropy as the loss function. The Adam optimizer is utilized to optimize the loss function.

We have the human manually labeled 1,810 Amazon Alexa commands to train the CNN model. During the CNN model training, there are 10% of the commands are randomly selected to do the evaluation to prevent over fitting. The active learning algorithm, margin sampling is implemented to query the unlabeled samples. The workflow is shown in Figure 3.1. The labeled dataset is $L_i = \{x_i, y_i\}$, x_i represents the commands, and $y_i = 0\ or\ 1$ is the corresponding category. The number 1 stands for the information retrieval and 0 represents the action injection. For example,

$$L_k = \{Alexa,\ give\ me\ a\ fact,\ 1\}$$

$$L_{k+1} = \{Alexa, launch\ asian\ drama\ facts,\ 0\}$$

The command text was converted into the vector matrix by word2vec model. We calculate the accuracy of the CNN model based on a fixed validation datasets (this fixed datasets precisely is the testing datasets), which contains 475 entries.

In step (1), all labeled datasets L_i are fed in the CNN model. In the first round of training (2), the model has low accuracy, which is lower than the target criteria 95%. Then in step (3), the trained CNN model computes the uncertainty of the unlabeled data entries. The

uncertainty of the dataset entry is defined:

$$Q_{uncertainty}(x_i) = |P(0|x_i) - P(1|x_i)| \tag{3.1}$$

Where, $P(y_i|x_i)$ is the probability of command x_i in category y_i. If $Q_{uncertainty}$ has small value, it indicates that the classification model has difficulties to differentiate whether the command x_i belongs to category 0 or 1. The command x_i has high uncertainty. We identify 100 commands with the highest uncertainty, then manually label these commands and deposit them into the training datasets L, as shown in steps 4 and 5. Then the new training datasets L are used to refine the CNN model. After several iterations, the training datasets have enough sampling, the CNN model accuracy reaches the criteria.

3.3.2 Sensitivity classification analysis

The sensitive commands have the potential to disclose a user's privacy or access to a security system without authorization. For example, "Alexa, lock the door" and "Alexa, show/stop the camera" are the sensitive commands, while "Alexa, ask CNN for the latest news" is a non-sensitive command. In this part, we classify the commands based on their sensitivity. All these manually selected sensitive commands inspire us to use the keyword based approach to identify the sensitive commands. As the commands are usually brief, one verb and/or noun in the rest part of the command generally express function of the command. The basic idea is that we first create the sensitive keywords list {"lock," "camera," "password," "unlock," et al.}, if a given command contains any sensitive keywords, then it will be taken as a sensitive command.

1) We manually extract some sensitive keywords to create the initial keywords list R_0. From the labeled sensitive and non-sensitive commands, we employ the RAKE algorithm to extract

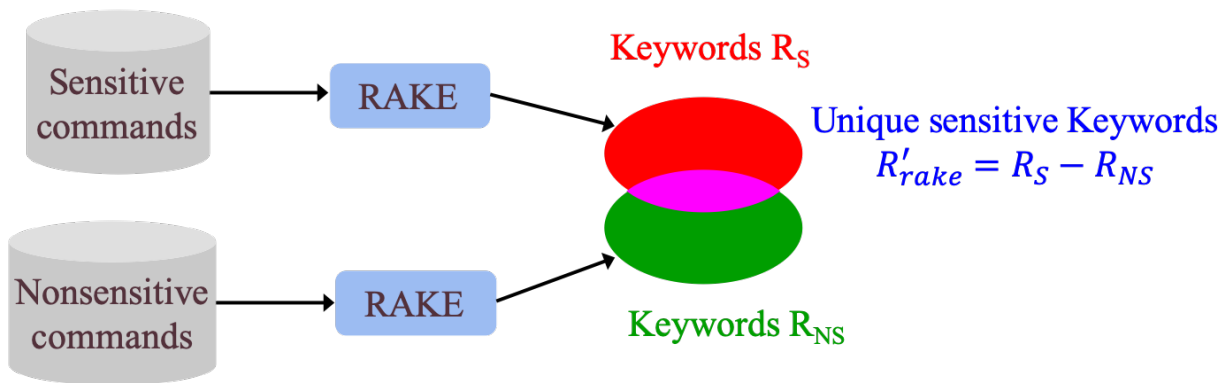

Figure 3.2: Workflow of sensitive keywords extraction from the labeled sensitive/non-sensitive commands

the keywords list R_S and R_{NS}, respectively, as shown in Figure 3.2. The unique initial sensitive keywords list is constructed $R_{rake}\prime = R_S$ - R_{NS}, deleting the keywords in R_S that already existed in R_{NS}, as shown in the red color. However, the RAKE algorithm has one drawback that it can extract the non-typical keywords, which causes the list $R_{rake}\prime$ contain the non-representative sensitive keywords. We manually delete those keywords and obtain R_{rake}. The sensitive keywords list from this step is $R_1 = R_0 \cup R_{rake}$.

However, the sensitive keywords list R_1 is identified from the labeled sensitive commands, which is not applicable to the whole commands. Therefore, in step 2), we enlarge the keywords list based on other unlabeled commands.

2) The word2vec model trained with 80,129 commands in section 3.3.1 is used to identify the new sensitive keywords, as illustrated in Figure 3.3. For every existed sensitive keyword in R_1 obtained in step 1), we search the similar new keywords and calculate their similarity score by the word2vec model. If the similarity score is larger than the threshold, then the keyword will be added into sensitive keywords list R_2. Finally, we complete the sensitive keywords extraction from the total commands, and the final sensitive keywords list is $R = R_1 \cup R_2$.

3) The sensitive keywords list R is fine tuned by the online survey. We started the user

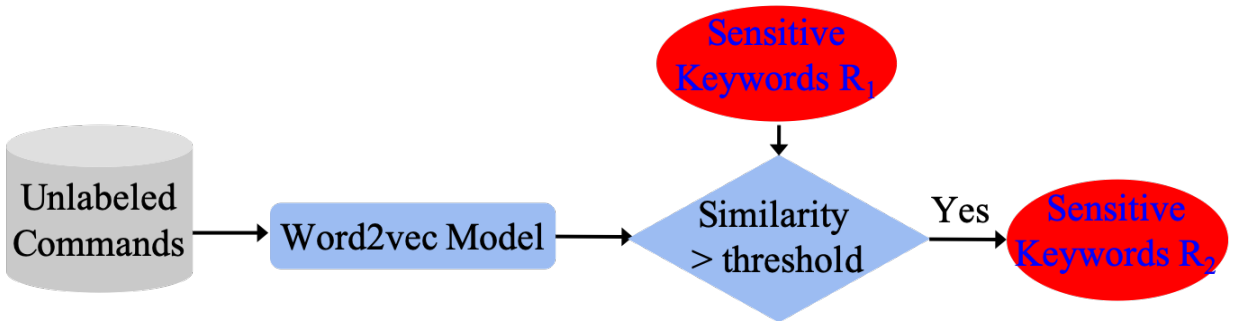

Figure 3.3: Workflow of sensitive keywords enlargement from work2vec model.

survey by Amazon Mechanical Turk [27]. At the beginning of the survey, we ask some general questions to the users: have they ever used the voice assistant, do they use the voice assistants in the phone, how long have they have been using VPA. Then we designed a list of commands to ask the user's opinions on the sensitive keywords and one example is provided as shown in Figure 3.4. In the command, the sensitive keyword "stay" that represents the main functionality is highlighted, the user need to classify this keyword to into different scales: "most sensitive", "sensitive", "neutral", "less sensitive" and "not sensitive." Only if the keyword in "most sensitive" or "sensitive" scales is taken as the sensitive keyword. The "Other" option allows the user to provide other keywords if he/she thinks the provided keyword can not represent the command's functionality. The last part of the survey is to ask the demographic information, like the age, gender, education level and occupation. We received 404 valid responses in a total. Among these responses, for one specific keyword, if

$$\text{No. of most sensitive} + \text{No. of sensitive} > \text{No. of neutral} + \text{No. of less sensitive} + \text{No. of not sensitive}$$

then this keyword is assigned to the sensitive class, otherwise, the keyword is categorized into the nonsensitive class. Through the fine tuning keywords process by the user survey, we obtain the final sensitive keywords list R_{final}.

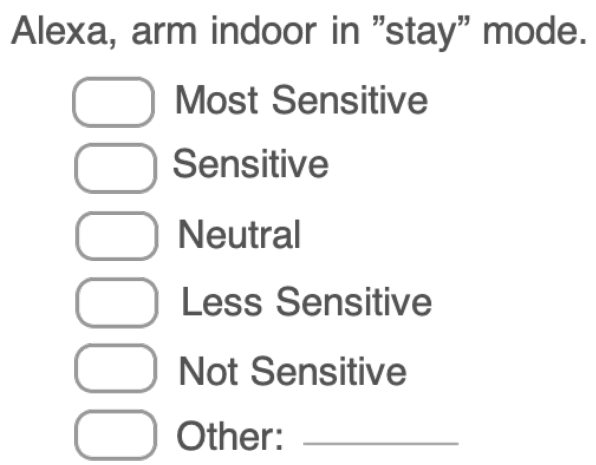

Figure 3.4: One example question from the user survey.

Chapter 4

Results

We applied the machine learning model to analyze 80,129 Amazon Alexa skills and 9,096 Google Home actions. Among the applications, we successfully identify 9,174 sensitive "action injection" commands and 3,280 sensitive "information retrieval" commands. The sensitive skills account for a small portion of 5.55% and the percentage of the sensitive commands to the overall commands is 11.62%. The accuracy, recall, precision and F1 score are measured for these two classification models, which have achieved excellent performance as shown in Table 4.1.

	Capability Classification	**Sensitivity Classification**
Accuracy	95.16 %	95.64 %
Recall	98.04 %	88.16 %
Precision	94.68 %	95.71 %
F1 score	96.33 %	91.78 %

Table 4.1: Performance of capability and sensitivity analysis

4.1 Capability classification evaluation

In the capability classification model, we have implemented active learning to enhance the sampling and the accuracy for this model reaches 95.16%. We have labeled 2,285 commands based on their capabilities. Among these labeled commands, we randomly picked 475 commands as validation datasets, and the rest 1,810 commands (501 commands are action injection, and 1,309 commands are information retrieval) were used to train the machine learning model.

By using the available labeled datasets, we obtained the initial trained CNN model. Then the model was applied to classify the unlabeled commands. The margin sampling was implemented to select the commands with high uncertainty. After collecting 100 commands, we manually label them again and deposit these commands into the training datasets to refine the machine learning model. The validation datasets were used to evaluate the accuracy of the model at each iteration. The criteria to stop active learning is when the model reaches an accuracy of 95%. If the accuracy is smaller than the target value, the framework will continue to the next iteration, and the margin sampling algorithm will query another 100 samples, which are employed to enhance the performance of CNN. After 10 iterations, the active learning sampled enough datasets and the CNN achieved the target accuracy 95.15%, the precision, recall and F1 score for this model is 94.68%, 98.04%, and 96.33%.

We compared the performance of our CNN + margin sampling with other baseline approaches, as shown in Figure 4.1. The Recurrent Neural Networks (RNN) have been widely used in NLP with great success and the RNN has a good performance for the commands capability classification. However, compared with RNN, the CNN model has improved accuracy, precision, recall and F1 score. In CNN + data clean approach, we preprocessed commands in the section 3.2.1. The accuracy of the CNN model slightly drops but the pre-

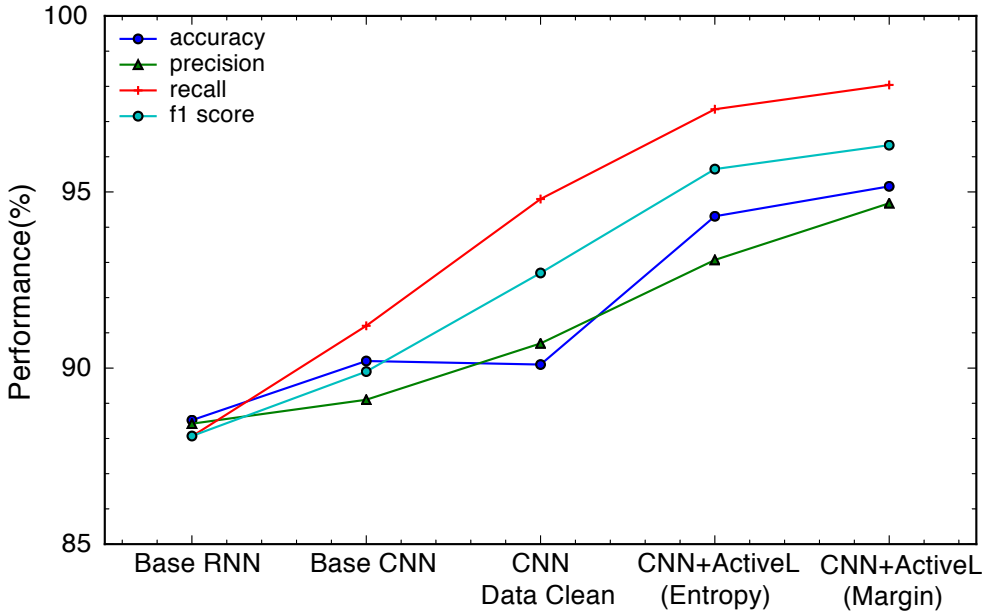

Figure 4.1: Performance of the classification model by using different methods

cision, recall and F1 score are all increased. Additionally, the active learning algorithm is used to further enhance the training data sampling to train a robust model. We utilize the entropy metric approach to select the unlabeled commands in each iteration, to increase the training datasets. The accuracy, precision, recall and F1 score have a significant improvement in CNN + entropy sampling. However, the margin sampling implemented in active learning shows better achievement than the entropy metric method. The CNN + margin sampling has the best performance over other methods.

4.2 Sensitivity classification model evaluation

4.2.1 Model evaluation

Our goal is to identify and characterize sensitive commands that are likely subject to attacks. We employed the keyword based approach to classifying the commands into two classes, sen-

sitive and nonsensitive. We first created the sensitive keyword list, if any sensitive keywords exist in the given command, then it is defined as the sensitive command. The accuracy of this keyword based approach reaches an accuracy of 95.6%.

We manually labeled 1,927 commands (515 sensitive commands and 1,412 nonsensitive commands) based on the sensitivity. Then 1,652 commands (508 sensitive commands and 1,144 nonsensitive commands) are randomly picked up to build the initial sensitive keyword list R_{rake}, which contains 68 sensitive keywords. The rest 275 commands are used for the final validation of our approach. The validation commands are excluded for building the sensitive keyword list.

How to build the robust sensitive keyword list mostly determine the accuracy of sensitivity classification model. From the step 1) in section 3.3.2, we obtained the initial sensitive keywords list R_1 which contains 68 words. Another 38 new sensitive keywords saved in the list R_2 are extracted by similarity calculation with the threshold of 0.8 by word2vec model. The similar word pairs are {("unlock", pin), ("disarm", "arm"), ("disarm", "activate"), etc.}. The sensitive keywords list by aggregating R_1 and R_2, $R = R_1 \cup R_2$ contains 106 sensitive keywords. $R =$ {"order", "mail", "lock", "babycam", "door", "keys", "pin", "email", "pin", "camera", "order", "balance", "unlock", "shipment", "location", "text", "check" etc.}. We varied the similarity threshold from 0.6 to 0.95, and the size of identified sensitive keywords list decreases. The threshold value of 0.8 gives rise to reasonable sensitive keywords list, and we obtain 38 new sensitive keywords. With threshold value of 0.75, the similarity method generates inappropriate word pairs like (decrease, seventy-five), (dim, percent). The sensitive keywords list is fine tuned by the user survey and the final sensitive keywords list R_{final} contains 57 keywords, R_{final} ={"clocking", "add", "message", "unlock", "door", "key", "text",...}.

The keyword based approach for the sensitivity classification achieves very good performance,

Skill name	Commands	Category
FordPass	*"Ask FordPass to start my car"* *"Tell FordPass to list all cars on the account"*	Action Injection Information retrieval
Ask My Buddy	*"Ask My Buddy to send help"*	Action Injection
Climax	*"Ask Climax Security to disarm"*	Action Injection
Newton Mail	*"Ask Newton who just mailed me"*	Information retrieval
Schlage Sense	*"Lock the door"*	Action Injection
Blink SmartHome	*"Show/stop the camera"* *"Show me the last activity from front door"*	Action Injection Information retrieval

Table 4.2: Examples of skills and their sensitive commands

with accuracy of 95.6%, precision of 95.71%, recall of 88.16% and F1 score 91.78% on the validation datasets. When we performed the sensitivity analysis, we found that the skills require account linking have higher probability contain sensitive commands. However, in the categories like {Education & Reference, Weather, Trivia & Access, News, Music & Audio, Games, Movies & TV, Novelty & Humor, Sports Kids}, most of the skills are designed for entertainment, relaxation, or extracting some general information (news, weather, stock price), they do not require the account linking and barely contain sensitive keywords.

4.2.2 Case Study: Motivating Examples

We provided sensitive skill examples as shown in Table 4.2. `Newton Mail` is a skill that can send emails or read incoming ones. This skill supports retrieval commands such as "Alexa, ask Newton who just mailed me," which reads the email for the user, as well as injection commands such as "What do you want me to do with this mail?" that allows user to delete, snooze, mark as read, archive or mark as spam. An attacker can utilize the skill to get

the user's emails and delete users' emails or sent fake emails. `Ask My Buddy` is a skill in the {"Health"} categories for emergency assistance. The user can save emergency contact information in Alexa. When there is a medical emergency, the user can shout to Alexa by saying the injection command "Alexa, ask My Buddy to send help." to call for help (*e.g.*, notifying the emergency contact to check on the user). If the skill is attacked, it will not send out an alert when the user asks for help. `Blink` and `Climax` are home security skills that control the front door cameras and alarms. For example, the user can change the home security mode by saying an injection command "Alexa, ask Blink to arm/disarm my home system," and check the camera feeds by a retrieval command "Alexa, show me the last activity from the front door." The underlying danger is that the security mode can be changed by attackers and it might release the user's recorded video information. `Schlage Sense` controls the smart locks on the doors. The user can use this skill to lock/unlock the door using the injection command "Alexa, lock/unlock(PIN required) the front door" and check the door status by saying the retrieval command "Alexa, is the front door locked." Alexa will tell the user if the battery is low after locking the door by voice command. The possible threat is that this skill gives incorrect door status to the user, leaving the user's home in dangerous situations. `FordPass` is still a skill to control network connected cars, user can control the car by injection commands "Alexa, ask FordPass to start(PIN required)/stop my car," and obtain vehicle information by retrieval commands "Alexa, ask FordPass is my tire pressure low."

4.3 Measure security implication of skills

We carried out extensive analysis on 89,225 commands from 30,000+ skills in Amazon Alexa and Google Home platforms. By two levels of investigation, capability and sensitivity, the

	No. of Sensitive cmd		No. of Nonensitive cmd	
	Inject.	Retri.	Inject.	Retri.
Amazon Alexa	8,503	2,939	20,550	48,128
Google Home	671	341	3,018	5,066
Total	9174	3280		

Table 4.3: Commands distribution based on their capability and sensitivity in Amazon Alexa and Google Home stores.

distribution of the commands is shown in Table 4.3. From these commands, there are 29,053 action injection and 51,067 information retrieval commands from Amazon Alexa, and 3,689 action injection and 5,407 information retrieval commands from Google Home. The amount of sensitive commands is 11,442 from Amazon Alexa and 1,012 from Google Home (12,454 sensitive commands in a total).

Among these sensitive commands, there are 9,147 sensitive-action injection commands and 3,280 sensitive-information retrieval commands. The sensitive commands predominantly exist in categories {Smart Home, Connected Car, Productivity} and {Home Control, Productivity} categories in Amazon Alexa and Google Home, respectively, as shown in Table 4.4 and 4.5. The {Smart Home} or {Home Control} are the most critical categories. The number of sensitive action injections commands is three times more than the number of sensitive information retrieval commands. The skills are mainly for operating the smart home device. For example, turn on/off the security system, lock/unlock the smart door. The skills in {Connected Car} are designed for extracting car information and operating car by voice, like ask how much batter life and turn on/off the engine. The {Productivity} category aims to help life to be more productive and convenient. For instance, ask the skill to send a message to a friend, read the message, make a phone call. In these two categories, there are more sensitive information retrieval commands than the sensitive action injection commands. The

future researches can focus on testing these sensitive skills and commands, and validate the vulnerability against different attacks.

Other categories like {Music & Audio, Sport, News, Weather, Kids, Movies & TV, etc.} designed for relaxation leisure and entertainment barely contain sensitive commands. For {Health & fitness} category, people intuitively assume the skills can store user's private information, like age, weight, health condition, and there is a risk that this information will be leaked. However, most of the skills just give the guidance for fitness and are nonsensitive skills.

Category	Sensitive		Nonsensitive	
	Inject.	Retriev.	Inject.	Retriev.
Smart Home	8,095	2,218	157	265
Connected Car	118	207	13	62
Productivity	84	64	990	2,335
Lifestyle	45	60	1,648	2,952
Business & Finance	45	60	1,648	3,842
Shopping	29	25	152	374
Home Service	18	15	164	244
Food & Drink	14	43	562	1,770
Music & Audio	11	4	4,527	4,735
Game & Trivia	10	7	391	5,359
Education & Reference	8	9	1,948	4,667
Communication	7	52	439	601
Travel & Transportation	6	10	601	1,175
Health & Fitness	6	19	432	900
Social	4	7	227	624
Local	3	7	448	1,381
Sports	1	2	703	1,916
Movies & TV	-	3	303	776
Novelty & Humor	-	2	1,063	4,169
Weather	-	2	491	1,189
News	-	-	3,222	7,443
Kids	-	-	592	1,078

Table 4.4: Amazon Alexa commands distribution based on their capability and sensitivity in different categories

Category	Sensitive		Nonsensitive	
	Inject.	Retriev.	Inject.	Retriev.
Home Control	642	218	1,092	604
Productivity	10	14	119	147
Shopping	6	20	135	331
Health & Fitness	4	-	163	257
Communication	4	-	64	172
Movies & TV	3	-	99	130
Travel & Transportation	1	6	96	327
Food & Drink	1	3	66	163
Business & Finance	-	11	29	325
Education & Reference	-	3	104	643
News	-	2	416	518
Local	-	1	26	269
Music & Audio	-	-	246	61
Games & Trivia	-	-	208	600
Sports	-	-	57	119
Weather	-	-	50	115
Art & Life	-	-	34	226
Kids	-	-	14	59

Table 4.5: Google Home commands distribution based on their capability and sensitivity in different categories

Chapter 5

Limitations, Future Works and Conclusions

5.1 Limitations

There exists a few limitations in this paper. The first limitation is related to how we label the sensitive voice commands, given that the definition of sensitivity can be quite subjective. The second limitation is that some commands' capability are blurring, how to label the command only depends on the annotator's intuition. For example, the commands "Alexa, tell History of Poland Podcast to go back 5 episodes" and "Alexa, tell Based on a True Story to play the newest episode" [28] can be argued to belong to the action injection class, as the user ask the skill to play. But these commands also give the user information i.e., the content of the episodes. For these commands, it is difficult to distinguish whether they are information retrieval or action injection commands.

The third limitation is that the literal meaning of the command does not clearly express its

functionality. For instance, in the skill "Smart Door lock" [29], we test the conversation with VPA as shown:

1. Alexa: Welcome to Door lock application, you can ask me who is in the front door, open the door.
2. Me: Open the door.
3. Alexa: The door is open.
4. Me: Who is at the front door.
5. Alexa: Guest is waiting at the door.

Actually, we do not have a smart device installed and just simply enable this skill. The description for this skill is that "I made this skill for main door security. I publish the code for raspberry pi in hackster.io/taifur. By running the code in your raspberry pi and using this skill you can make your own smart door lock." [29] This skill seems to provide the code to help the users to make their own smart lock. It is not sensitive skill and the commands are not sensitive, but the commands still contain the sensitive keywords. Therefore, it is not enough to only analyze the commands provided by the webpage, but a real conversation with skills and the accompanying description on the page should also be considered.

5.2 Future works

This project provides the preliminary understanding about the functionalities of skills and their commands. The identified sensitive skills and our designed classification models are able to provide guidance for the future researches.

1) The sensitive skills, especially for the skills that contains the information retrieval commands, can be used to perfect the developers' documentations. For example, the newly

developed skill should have the invocation name that sounds very differently from the existing sensitive skills.

2) For the sensitive skills, the developers need to introduce extra layer of protection for authenticating. For instance, the PIN code or voice profiling is asked to set up when first time invoking the sensitive skills. But for the nonsensitive skills, there is no need for the authentication. So it will reduce the overhead time for the users to use nonsensitive skills and maintain a good user experience.

3) Our designed framework can be used as a screen tool, to pre-test the skills and their commands if they have the potential security risk before publishing to public.

4) The analysis from this project is static, the future study can focus on dynamically testing the sensitive skills and validate the vulnerabilities against different attacks. The Amazon Alexa only shows up to three sample commands in the introduction page. Other commands are introduced in the description part or informed users when the user interact with the skills. We can test the real conversation with the skills, to uncover the hidden commands. Different voice commands are sent to the VPA, after receiving the input, the skill will send back corresponding responses. The voice-to-text module transcribe the conversation into text. Our designed framework can be used to measure the hidden commands.

5.3 Conclusions

In this project, we defined sensitive/nonsensitive commands. The sensitive commands have the potential to leak the users' private information and pose a security threat to the users. First of all, we built a CNN with active learning framework to investigate the capabilities of commands (action injection or information retrieval) in VPA to find the attack surface.

Secondly, the keywords based approach, which is defined as that the command is categorized into sensitive class if it contains any sensitive keywords, is implemented to measure the sensitivity of the commands. The sensitive keywords list was built by using the RAKE algorithm, word2vec model and online user survey.

These two levels of analysis allow us to identify the sensitive-action injection and sensitive-information retrieval commands. We successfully identify 12,454 sensitive commands in a total from both Amazon Alexa and Google Home. Additionally, we found that most of the sensitive commands mainly located in the Smart Home (Home Control) categories.

Bibliography

[1] Matthew B Hoy. Alexa, siri, cortana, and more: an introduction to voice assistants. *Medical reference services quarterly*, 37(1):81–88, 2018.

[2] Abdulaziz Alhadlaq, Jun Tang, Aleksandra Korolova, and Marwan Almaymoni. Privacy in the amazon alexa skills ecosystem. *Star*, 217:11, 1902.

[3] 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, 2018.

[4] Burger King faces backlash after ad stunt backfires burger king faces backlash after ad stunt backfires. https://wgntv.com/2017/04/13/burger-kings-whopper-ad-stunt/. Accessed: 2020-01-22.

[5] Liwei Song and Prateek Mittal. Inaudible voice commands. *arXiv preprint arXiv:1708.07238*, 2017.

[6] Nirupam Roy, Sheng Shen, Haitham Hassanieh, and Romit Roy Choudhury. Inaudible voice commands: The long-range attack and defense. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, pages 547–560, 2018.

[7] Yuan Tian, Nan Zhang, Yueh-Hsun Lin, XiaoFeng Wang, Blase Ur, Xianzheng Guo,

and Patrick Tague. Smartauth: User-centered authorization for the internet of things. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 361–378, 2017.

[8] Hamza Harkous, Kassem Fawaz, Rémi Lebret, Florian Schaub, Kang G Shin, and Karl Aberer. Polisis: Automated analysis and presentation of privacy policies using deep learning. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 531–548, 2018.

[9] Fei Liu, Nicole Lee Fella, and Kexin Liao. Modeling language vagueness in privacy policies using deep neural networks. In *2016 AAAI Fall Symposium Series*, 2016.

[10] Norman Sadeh, Alessandro Acquisti, Travis D Breaux, Lorrie Faith Cranor, Aleecia M McDonald, Joel R Reidenberg, Noah A Smith, Fei Liu, N Cameron Russell, Florian Schaub, et al. The usable privacy policy project. *Technical report, Technical Report, CMU-ISR-13-119*, 2013.

[11] Xiang Pan, Yinzhi Cao, Xuechao Du, Boyuan He, Gan Fang, Rui Shao, and Yan Chen. Flowcog: context-aware semantics extraction and analysis of information flow leaks in android apps. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1669–1685, 2018.

[12] Rahul Pandita, Xusheng Xiao, Wei Yang, William Enck, and Tao Xie. {WHYPER}: Towards automating risk assessment of mobile applications. In *Presented as part of the 22nd {USENIX} Security Symposium ({USENIX} Security 13)*, pages 527–542, 2013.

[13] Bo Sun, Akinori Fujino, and Tatsuya Mori. Poster: Toward automating the generation of malware analysis reports using the sandbox logs. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1814–1816. ACM, 2016.

[14] Yuhong Nan, Zhemin Yang, Min Yang, Shunfan Zhou, Yuan Zhang, Guofei Gu, Xiaofeng Wang, and Limin Sun. Identifying user-input privacy in mobile applications at a large scale. *IEEE Transactions on Information Forensics and Security*, 12(3):647–661, 2016.

[15] Xiaojing Liao, Kan Yuan, XiaoFeng Wang, Zhou Li, Luyi Xing, and Raheem Beyah. Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 755–766. ACM, 2016.

[16] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[17] Cynthia A Thompson, Mary Elaine Califf, and Raymond J Mooney. Active learning for natural language parsing and information extraction. In *ICML*, pages 406–414. Citeseer, 1999.

[18] Yazhou Yang and Marco Loog. Active learning using uncertainty information. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 2646–2651. IEEE, 2016.

[19] Yoav Freund, H Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine learning*, 28(2-3):133–168, 1997.

[20] Andrew Kachites McCallumzy and Kamal Nigamy. Employing em and pool-based active learning for text classification. In *Proc. International Conference on Machine Learning (ICML)*, pages 359–367. Citeseer, 1998.

[21] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, 2010.

[22] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[23] Nan Zhang, 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.

[24] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

[25] François Chollet et al. Keras. https://keras.io, 2015.

[26] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[27] Amazon Mechanical Turk amazon mechanical turk. https://www.mturk.com/. Accessed: 2020-01-20.

[28] Amazon alexa history of poland podcast. https://www.amazon.com/Wizzard-Media-History-Poland-Podcast/dp/B07BSLN1G4/ref=sr_1_1?keywords=History+of+Poland+Podcast&qid=1578603563&s=digital-skills&sr=1-1.

[29] Amazon alexa smart door lock. https://www.amazon.com/Md-Khairul-Alam-Smart-Door/dp/B079Z8L87Y/ref=sr_1_1?keywords=smart+door+lock&qid=1578601493&s=digital-skills&sr=1-1.