Spying through Your Voice Assistants: Realistic Voice Command Fingerprinting

Dilawer Ahmed
North Carolina State University
dahmed2@ncsu.edu

Aafaq Sabir
North Carolina State University
asabir2@ncsu.edu

Anupam Das
North Carolina State University
anupam.das@ncsu.edu

Abstract

Voice assistants are becoming increasingly pervasive due to the convenience and automation they provide through the voice interface. However, such convenience often comes with unforeseen security and privacy risks. For example, encrypted traffic from voice assistants can leak sensitive information about their users' habits and lifestyles. In this paper, we present a taxonomy of fingerprinting voice commands on the most popular voice assistant platforms (Google, Alexa, and Siri). We also provide a deeper understanding of the feasibility of fingerprinting third-party applications and streaming services over the voice interface. Our analysis not only improves the state-of-the-art technique but also studies a more realistic setup for fingerprinting voice activities over encrypted traffic. Our proposed technique considers a passive network eavesdropper observing encrypted traffic from various devices within a home and, therefore, first detects the invocation/activation of voice assistants followed by what specific voice command is issued. Using an end-to-end system design, we show that it is possible to detect when a voice assistant is activated with 99% accuracy and then utilize the subsequent traffic pattern to infer more fine-grained user activities with around 77-80% accuracy.

1 Introduction

Voice assistants offer convenience by automating various day-to-day activities through voice commands (e.g., searching the web, making online purchases, etc.). Furthermore, voice assistants enable the automation of home appliances through voice commands. Today, around 42.1% of the US population uses a voice assistant [26] and this number is expected to increase in the next year [18]. Amazon Alexa, Google Assistant, and Siri are the top three most popular consumer voice assistants [28]. Amazon Alexa is the market leader, making up more than half the smart speaker sales by brand in 2022 [49].

However, such convenience also poses new security and privacy risks. The general perception is that any communication between a user and their voice assistant remains private over the internet and is not visible to any party that is not directly concerned in relation to the request. In other words, any passive network observer, local or remote, should not be able to infer high-level user commands just by passively viewing the encrypted network traffic. Any amount of information leaked is a weakness in this assumption and means that a privacy leakage has occurred, which depending on the amount of information leaked, can be a serious privacy threat.

Voice assistants, generally, have different types of responses to different types of commands. Some commands require built-in responses, while others require a generic web-searched response. Some voice assistants, such as Amazon Alexa, also provide third-party functionality called "skills," which are analogous to the concept of apps on mobile devices. Third parties develop most of the available skills in the Amazon skill store to extend the voice assistant's capabilities. Researchers have recently looked at analyzing the skills ecosystem and have identified various flaws in the vetting process [31] as well as potential malicious skills themselves [30, 48]. Since the primary mode of interaction with voice assistants is through voice commands, prior work has also shown that, under ideal settings, fingerprinting simple voice commands is possible [10,29,33,60]. The latest work by Mao et al. [33] shows that Alexa voice commands can be fingerprinted with around 93% accuracy, highlighting the threat of a passive adversary on user privacy. However, existing works have considered an ideal setting where the adversary can sniff traffic on the local network. In the real world, such a vantage point is hard to achieve for home networks. Furthermore, existing works only consider isolated traffic from voice assistants, thus, do not consider an end-to-end analysis.

In this paper, we consider a threat model in which the adversary is not on the local network and has no direct way of knowing the exact time when any activity was *active* on the voice assistant. We consider the adversary outside the home network/router. While being more realistic, this threat model brings the challenge of not being able to filter out only the desired device traffic from all the home traffic due to the use of NAT, which is standard on home networks. To address the

problem of filtering device traffic and inferring the time of activity, we exploit the fact that the traffic corresponding to the voice assistant activity (i.e., voice command) will be observed on the network immediately following the invocation of the device. So, we first train a model to infer, from the live network traffic, when a voice assistant device is activated, i.e., when the wake word is triggered. We then probabilistically filter the immediate subsequent network traffic to filter out the flows unrelated to the voice assistant and use the remaining flows to fingerprint the actual voice command. We explore additional network traffic-level features that were not considered by previous works. We also expand our work to include streaming commands and skills. Specifically, we focus on popular and privacy-sensitive activities. Furthermore, we analyze three popular voice assistant platforms to ensure our approach is generalizable across platforms.

In particular, we seek to answer the following research questions **RQ1:** *Can we fingerprint the activation of voice interface across different voice assistant platforms?* Detecting the time of invocation is not just an essential first step in fingerprinting voice commands but also information leakage in itself. **RQ2:** *Can we fingerprint voice commands across different voice platforms as well as different forms of voice commands (e.g., skills and streaming commands)?* We fingerprint Alexa, Google Assistant, and Siri under realistic settings to understand how the machine learning model's performance varies across voice assistants. We also study different types of voice commands, such as invoking a streaming service or third-party skills. **RQ3:** *Can we build an end-to-end fingerprinting approach under a real-world setting?* By first inferring the invocation of the assistant and then fingerprinting the activity, we can design an end-to-end solution that can be deployed to fingerprint voice assistants in the wild.

To answer these questions, we collect network traffic for different voice commands using three popular voice assistants (Google, Alexa, and Siri). We also collect traffic for streaming services and skills for Alexa. In addition, to analyze the end-to-end performance in the presence of noise, we also collect a dataset for Alexa, which contains a mix of skills, streaming commands, and simple commands with additional background noise traffic from a smartphone, laptop and smart TV to emulate a noisy home network. Next, we train lightweight machine learning models to detect the activation or invocation of voice assistants from the encrypted network traffic and use the following traffic to fingerprint the actual voice command. In summary, we make the following contributions:

- We show that adversaries can fingerprint voice assistant activities on all three major voice assistant platforms improving the state-of-the-art [33] with AutoML [9, 19] and novel features. We also consider a more realistic setting where the adversary passively monitors all traffic from a smart home, not just the local traffic from a voice assistant.
- We explore additional features using information from flows, bursts, protocols, and endpoints to enhance classifier performance. Compared to existing works [33, 60], such features were often ranked higher by multiple feature importance metrics such as ANOVA and Mutual Information. Furthermore, when comparing with existing works, we see a performance improvement in the range of 2% to 69%.
- We introduce an end-to-end approach, where we first detect wake-word invocation from network traffic using a lightweight machine learning classifier and then use the traffic that follows to predict the actual command with around 77-80% accuracy in the presence of real-world background noise from other devices using the same network.
- We present, to our knowledge, the first study of fingerprinting streaming commands and third-party apps running on top of voice assistants. We have publicly released our annotated data and models for the research community.[1]

2 Related Works

Inference through Network Traffic. There is a large body of work in website fingerprinting, which refers to identifying the website a user visits by analyzing the network traffic generated while visiting a given website [12, 21, 23, 40, 41, 53, 65]. Researchers have also looked at analyzing the encrypted traffic generated by mobile apps to infer not only the app used but also the specific functionality executed on the app [5, 7, 15, 44, 51, 56, 59]. Others have tried to infer contents from encrypted VoIP [63, 64], and video-streaming traffic [46, 52]. Researchers have also shown that it is possible to infer the SSH passphrase by exploiting the timing delays between subsequent IP packets [55].

IoT Device & Activity Fingerprinting. Researchers have used network traffic to identify IoT devices utilizing various approaches, including DNS traffic, DHCP traffic, and HTTP headers [43]. Supervised and unsupervised learning-based approaches have also been used to identify IoT devices [34, 36, 54]. Saidi et al. [50] proposed methods of detecting IoT devices at scale. Ahmed et al. [6] presented the largest to-date study of IoT device fingerprinting in which they consider multiple factors such as fingerprinting across time, region, and different datasets and under different constraints. There has also been research to infer the activities being performed on IoT devices. Copos et al. [16] used Nest Thermostat and Nest Protect to infer their activity and usage patterns from network traffic. OConnor et al. [39], and Triminanda et al. [57] used packet size and direction to infer activities on IoT devices. Apthorpe et al. [8] used traffic volume and shape-based features to infer device activity. Acar et al. [4] identify the state and actions of multiple IoT devices using different protocols such as WiFi, ZigBee, and BLE.

Voice Command Fingerprinting. Kennedy et al. [29] in-

[1] https://github.com/dilawer11/va-fingerprinting

Setup details		Kennedy et. al. [29]	Batyr et. al. [10]	Wang et al. [60]	Mao et. al. [33]	Our
Features used	Packet timing	○	○	○	●	●
	Packet sizes	●	●	●	●	●
	Traffic direction	●	●	●	●	●
	Burst-based	○	○	○	○	●
	Flow-based	○	○	○	○	●
	Ports-based	○	○	○	○	●
	IP/domain-based	○	○	○	○	●
	Protocol-based	○	○	○	○	●
Voice Assistant Platforms	Alexa	●	●	●	●	●
	Google Assistant	○	○	●	○	●
	Siri	○	○	○	○	●
Command types	Simple	●	●	●	●	●
	Streaming	○	○	○	○	●
	Skills	○	○	○	○	●
Adversary type	Local	●	●	●	●	●
	Non-local	○	○	○	○	●
Assumptions	Filtered traffic	●	●	●	●	○
	Aligned traffic	●	●	●	●	○
Analysis	Invocation Detection	○	○	○	○	●
	Activity Detection	●	●	●	●	●
	End-to-End	○	○	○	○	●

Table 1: Comparison with existing work on voice command fingerprinting. Symbols convey the following meanings – ○: not considered, ◐: partially considered, ●: considered.

troduced the voice command fingerprinting attack on Alexa. They collected a dataset of 100 commands and achieved an accuracy of 33.8%. Another work using this dataset used locality-sensitive hashing to improve the accuracy of the initial work to 42% [10]. In a follow-up work to their original work [29], Wang et al. performed open-world analysis, including Google Home Assistant, and improved the accuracy to 92.89% [60]. Mao et al. [33] created a novel deep learning model and included timing-based features to improve the accuracy on the same dataset to 93.36% utilizing features such as direction, time, and size of packets.

Distinction from Prior Work. Our approach is different as we infer user activity within a time window of one minute and engineer various features within this window to extract as much information as possible, including simple and multi-valued features. We extract many new features, such as protocols, hostnames, and flow-based statistics. We also consider a more realistic attack model where an adversary is not on the local network but rather remotely observes network traffic. We use the dataset mentioned above to benchmark our approach and show that by utilizing our features, we can improve the accuracy to 95.70% while using a stricter set of assumptions. Prior works also do not consider invocation detection for any platform and also do not consider activity detection for Siri. To the best of our knowledge, we are the first to consider different types of commands and skills. Previous research on voice command fingerprinting only focuses on simple voice commands, whereas we focus on both commonly used and privacy-sensitive commands which can reveal sensitive information about the user. Table 1 provides a comparison with related works.

3 Methodology and Data Collection

3.1 Threat Model

The threat model consists of an adversary, which can passively observe and capture the network traffic. The adversary's access point does not require them to be on the local network as long as they have access to all the traffic from the target device. Furthermore, the adversary and target device can be separated by a NAT, meaning that the adversary cannot use a simple IP filter to isolate the target device traffic from other devices on the network. As a result, the traffic captured by the adversary can have noise from other devices. The adversary, however, cannot modify packets or decrypt any standard encryption used in certain communication protocols such as TLS. Any plain-text protocols can, however, be sniffed and used for analysis (e.g., using DNS traffic for passive DNS attack). Adversaries' other capabilities include deploying its own data collection setup, as described below, and using the data to analyze and train the required models.

3.2 Proposed Approach

To detect activity or commands on the voice assistant, we first need a mechanism to infer when a voice assistant is activated. We can then use this information to isolate the traffic related to the *command* and fingerprint the actual voice command. To that end, we use a machine learning-based binary classi-

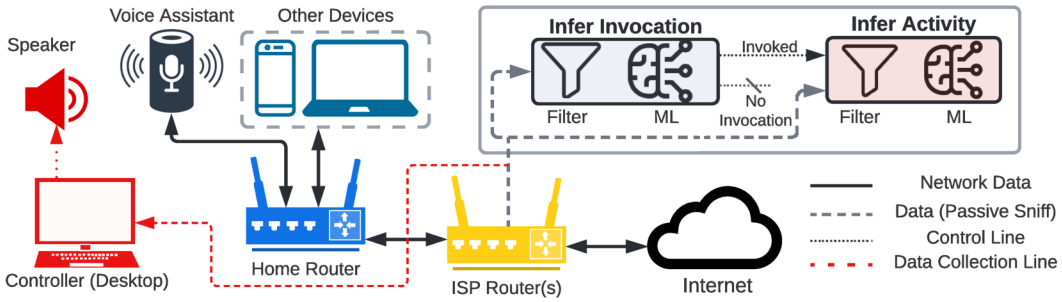

Figure 1: Fingerprint process diagram. Network traffic is sniffed between the home router and internet endpoints. First, the invocation is inferred and upon detection the activity detection phase is triggered. Each of the modules uses different filtering and ML workflows.

fication pipeline to infer *invocations* and a separate machine learning-based multi-class classification pipeline to infer *commands* or *activities*. We refer to the invocation classifier as *Invocation Detection* and the activity classifier as *Activity Detection*. We propose a novel flow filtering method to isolate the traffic from voice assistants from other traffic generated by other devices on the same network. Figure 1 provides an overview of our proposed approach.

3.3 Data Collection Setup

Our data collection setup consists of two routers, a Ubuntu Desktop with a speaker, and voice assistants. We had three voice assistants, each representing one of the top three popular platforms: Amazon Alexa, Siri, and Google Assistant. We use the Amazon Echo Plus (2nd Gen, L9D29R) for Alexa. For Siri, the Apple HomePod mini (MJ2D3LL/A) is utilized. We use the Nest Mini (2nd Generation, GA00781-US) for Google Assistant. Both the routers are Linksys WRT1900ACS routers running OpenWRT. One of the routers acts as a home router and a wireless access point for the voice assistants. We refer to this router as the *home router*. The home router is connected to the internet through another router, which we refer to as the *ISP router*. All the network traffic from the voice assistants will go through the *home router*, which has NAT enabled, and then to the *ISP router*. We collect the traffic at the *ISP router*. Figure 1 shows the data collection setup visually with red colored lines showing the controller and speaker used for automated data collection and traffic capturing. *Tcpdump* [20] running on the *ISP router* dumps the packet capture files (PCAPs) to the Ubuntu Desktop. We use Google Cloud's Text-to-Speech API to convert the text commands to audio files. A script on the desktop plays the commands (audio files) via the speaker.

Since the voice assistant's ability to understand the command can vary depending on factors such as noise and an imperfect speech recognition engine, we employ solutions to verify that the voice assistant understands the command it was issued. For example, Amazon Alexa and Google Assistant allow users to view their "interaction history" of what the assistant heard and how it responded. We use these services to validate each invocation for these two assistants. We use Selenium to scrape and retrieve the web-based record of activities for Google Assistant and Alexa to match the phrase passed to the speaker. We noticed about 2% of the samples failed verification and were consequently removed later. Upon successful or failed verification, the script then stores the timestamp and label of the command along with other information in a separate JSON file. Next, the script waits for a certain amount of time (e.g., one minute) and moves on to the next iteration. Finally, if the voice command issued requires a force stop to end (e.g., stopping streaming services), the controller issues the stop command through the speaker (e.g., "Alexa, stop") after the designated waiting period. Siri, however, does not provide any accessible interface to retrieve interaction history, so we manually verified one round of invocations.

3.4 Datasets

We collect datasets from all three platforms. The majority of our analysis, however, focuses on the Alexa platform as it is the largest platform by market size in the voice assistant-enabled smart speakers [18, 28]. Table 2 highlights the different datasets. The full list of commands considered in our work can be found in the extended version of our paper. [2]

Alexa skills, commands and streaming services. These datasets consist of simple voice commands, streaming commands, and skills for Alexa. Simple commands are, generally, short-lived commands which follow the structure of quick request-response. Examples of such commands could be "Alexa, what time is it?" or "Alexa, set the alarm for 7 am". Existing works only consider such commands, as used in the "Deep VC Fingerprinting" dataset [60]. Such commands usually take less than 30 seconds from invocation to completion.

Streaming commands are, generally, long-lived commands which often run until explicitly stopped. Examples of these commands can be "Alexa, play news from CNN" or "Alexa, play music from Spotify". Lastly, skills, which are commonly

[2]Extended paper version: https://github.com/dilawer11/va-fingerprinting/blob/main/paper_resources/ahmed_spying_usenix_2023_extended_version.pdf

Dataset	VA Platform	Type of Commands and Description	Noise	Classes (Samples)
simple_100_alexa	Alexa	Simple voice commands	No	100 (100)
stream_15_alexa	Alexa	Streaming service invocation commands	No	15 (100)
skills_100_alexa	Alexa	Multiple skill invocation commands	No	100 (100)
simple_50_google	Google Assistant	Simple short voice commands	No	50 (100)
simple_50_alexa	Alexa	Simple short voice commands	No	50 (100)
simple_50_siri	Siri	Simple short voice commands	No	50 (100)
mix_100_alexa	Alexa	Simple, skills and streaming commands	Yes	45 +40 +15 = 100 (100)
Deep_VC [29]	Alexa Google	Simple short voice commands	No	100 (1,500)
other	*	N/A	*	N/A

Table 2: Characteristics of different datasets we collected for our study. Web interaction scripts generated background traffic as a source of noise. *other* datasets were collected to only evaluate a certain scenario and we have described these datasets as and when needed.

referred to as apps for the Alexa ecosystem are typically developed by third parties to provide added functionality. An example of a skill could be a bank skill that enables users to fetch their bank balance [3] or a game skill that lets users play the Jeopardy game [24]. We label all different commands which can launch a skill by the identifier of that skill. We capture traffic from 100 simple commands, 15 streaming commands, and 100 skills each for 100 samples to create a total of $215 \times 100 = 21,500$ samples.

The command selection process consisted of selecting popular or sensitive commands. We followed the steps that a general user would adopt to select popular commands. For example, utilizing platform-specific search engines and using top-rated skills. For selecting sensitive commands, we focused on commands/skills that leak a user's *gender*, *religious belief*, *sexual orientation*, *financial interest*, or *political leaning*.

Commands across other voice assistant platforms. These datasets include simple voice commands from all three platforms we consider. We capture traffic from 50 simple voice commands on each platform with 100 samples for each command, totaling $50 \times 100 = 5,000$ samples. The commands between the voice assistants mostly overlap, however, some commands varied between assistants to accommodate for some assistants not supporting certain commands or not responding correctly. For simple voice commands, we again focus on commands which can leak sensitive information about users' lifestyles.

Alexa traffic with background noise. The third type of dataset we collect facilitates a realistic end-to-end experiment. We create a command set with a mix of streaming commands and skills to compose a total of 100 classes (15 streaming commands, 40 skills, and 45 simple voice commands). We also connected other devices to the same router to emulate a realistic home network with traffic from other devices overlapping with traffic from the voice assistant device. We create two web-crawling scripts which generate a high amount of traffic. One script repeatedly visits the top *Alexa 1,000* domains and clicks on the page several times at random locations to generate more randomized traffic. The other script repeatedly opens the Amazon web page and does similar random clicking a few times on each visit. This is to remove any potential bias due to *amazon.com* domains only being accessed by the Alexa device. The resulting dataset was approximately 12 times larger than the dataset with only simple Alexa commands. This shows that the amount of noise added is considerable, and any system that does not consider noise should adversely affect the classifier's performance.

Other public datasets. We also use the "Deep VC Fingerprinting" dataset [29], which was also used by existing works [10,29,33,60] to benchmark our approach and compare it to previous techniques for fingerprinting voice commands. The dataset consists of 100 simple commands selected empirically based on a mixed methods approach of selecting top commands from Google Search, domain knowledge, and personal experience. Each voice command was invoked 1,500 times to generate a total of 150,000 samples. Five different voice templates were used for each command (text-to-speech), and each voice has 300 samples collected through it. This dataset, however, lacks validation of commands to ensure the voice assistant properly perceived the command and the invocation attempt did not lead to a false invocation.

3.5 Data Pre-processing

We use *tshark* [58] to convert the PCAP files to CSV format for further processing. Like most other devices, voice assistants use standard encryption techniques to encrypt most traffic between the device and internet endpoints. Thus, the communication content itself is not observable over the channel. Some protocols, however, are not encrypted and are plain text by default, e.g., DNS. Therefore, we extract only the header level information for all packets other than DNS, i.e., packet length, IP addresses, ports, flags, time, etc. We also extract the query Hostname and response IP addresses for DNS packets to create an IP address to Hostname mapping. This process is similar to passive DNS. Utilizing the IP address – Hostname mapping, we supplement our packet information by adding a *hostname* field. We ignore and remove any traffic not based on IP protocol from our captures, e.g., ARP. We also perform some additional cleaning to remove malformed packets from the dataset. Finally, we store the processed packet files (header information for each packet and the added information) in CSV format.

4 Invocation Detection

Invocation detection refers to identifying when a smart voice assistant device was activated or invoked. Invocation detection is integral to any efficient real-world attack on voice assistants because the adversary would not have a trivial or straightforward method to find when the user invoked a voice assistant just by observing the network traffic. So invocation detection would help pinpoint the time of invocation, which can then be used for activity detection by focusing on traffic following the user request to the voice assistant. Hence, invocation detection is the first step in our attack.

Users invoke or activate the voice assistant by uttering the *wake word* of the corresponding voice assistant. For example, "Alexa" is the default *wake word* on Amazon Echo devices. Apple's Siri assistant, by default, responds to "Hey, Siri". "Ok, Google" and "Hey, Google" are the default wake words on Google Assistant. Voice assistants remain in their *idle state* until they hear the wake word, which triggers them to start recording and communicating with the backend. This network communication results in a significant uptick in traffic immediately following the utterance of the wake word. Figure 2 shows an example of this uptick and spike in network traffic immediately following an invocation.

In a real-world attack, the invocation detection process should be quick, lightweight, and efficient. A complicated and complex model can waste significant computing resources, so we opt for only a few features and relatively simple machine learning models. To achieve this, we opted for a prediction every 2 seconds as this allows us to be precise enough with predicting invocation while also balancing the load of prediction calls on the underlying hardware. We created sliding windows of *width* 4 seconds over the continuous network traffic, which *slides* 2 seconds (corresponding to one shift every 2 seconds). The value 4 for *window_ width* was selected experimentally by evaluating different values such as 2, 4, 6, and 10. We found that across all three platforms, *window_width=2* gave the worst performance (around 70% accuracy) while *window_width=4* gave the best performance (around 99% accuracy). Increasing the *window_width* to 6 and 10 decreased the accuracy by 1% and 4%, respectively. We mark each window as having an invocation (1 or positive class), if we invoked the voice assistant in the first 2 seconds (out of 4 seconds) of the window. Otherwise, we mark it as not having an invocation (0 or negative class). The goal is to teach the classifier to differentiate between an *invocation* and *idle state*. Thus, while creating training data, we skip the traffic after a positive class label for a specific time interval. This "skipped" traffic is used for activity detection as it corresponds to an activity or voice command. Including this traffic as 'non-invocations' or negative class samples will lead to false positives as the voice assistant would not be in an idle state. We then extract features from each of these windows.

4.1 Feature Extraction & ML Model

Features Used. We saw that each voice assistant generated a sudden network traffic spike to a particular set of domains for each invocation. For example, saying the *wake word* to Alexa resulted in a sudden spike of network activity to two domains, namely, `unagi-na.amazon.com` and `avs-alexa-4-na.amazon.com`. Using UDP protocol, Google Assistant communicated with `www.google.com`. Siri contacted `dejavu.apple.com` using TCP protocol. Figure 8 in Appendix B highlights the top five endpoints contacted in the four-second window of an invocation for all three voice assistants. The figure confirms the unique *selected* domains (in orange color) for all three voice assistants.

We found that Alexa sends, on average, more traffic to `device-metrics-us.amazon.com`, but is only present in a few of the windows and hence not a good candidate. Conversely, for Google Assistant, `8.8.8.8` is present in all invocations but lacks in the amount of network traffic in each window, and this is Google's DNS domain and hence also not suited. On the other hand, the domains we *selected* are present in all the data windows and send/receive a significant amount of traffic. We, therefore, compute and extract the total size of incoming and outgoing packets to/from these *selected* domains as features to detect invocation. As a result, we have two to four features for each platform corresponding to the incoming and outgoing traffic size for the *selected* domains. For example, in the case of Alexa, we have four features, i.e., total output and input size for `avs-alexa-4-na.amazon.com` and the total output and input size for `unagi-na.amazon.com`. Similarly, for Siri, we have two features, i.e., total output and input size for `dejavu.apple.com`.

To measure the effectiveness of these features and understand if other applications (e.g., web activity) would conflict with these features and potentially cause false positives, we collected additional traffic to these domains through various sources such as web browsers, mobile apps etc. We plot the scatter plot of these features in Figure 3. We see that traffic from voice assistants to these domains significantly differs from the traffic generated to these domains using other applications. For example, voice assistants usually send more outgoing traffic (as they send voice recordings) and receive less incoming traffic. It was also interesting to note that even mobile versions of these voice assistants have varying traffic, as Siri uses less traffic to these domains on the mobile version. In contrast, the Alexa app uses more traffic to these domains and does not use the `unagi-na.amazon.com`. In our case, the Alexa app used `avs-alexa-13-na.amazon.com` instead of `avs-alexa-4-na.amazon.com`. However, we consider these domains to be the same as they are most likely used for load balancing across platforms. Thus, an attacker can use such unique domains to segregate network traffic.

Machine Learning Model. We compare the performance of multiple machine learning models such as AdaBoost classifier,

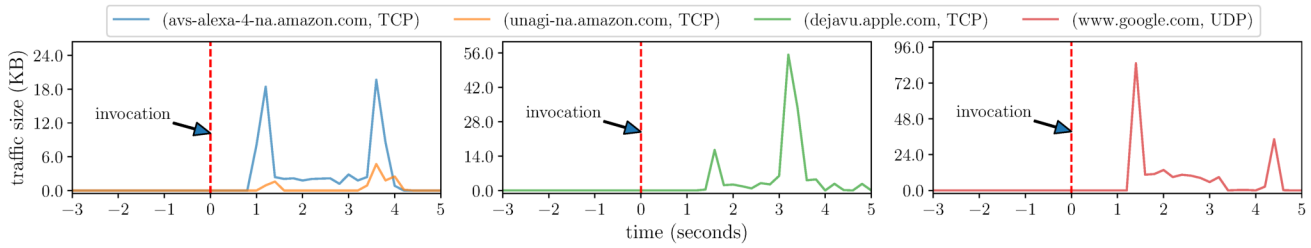

Figure 2: Immediately following the invocation, a spike can be seen in traffic to specific domains for each voice assistant. For example, Alexa communicates with `avs-alexa-4-na.amazon.com` and `unagi-na.amazon.com`; Siri communicates with `dejavu.apple.com` and UDP protocol, Google Assistant communicated with `www.google.com` using UDP protocol.

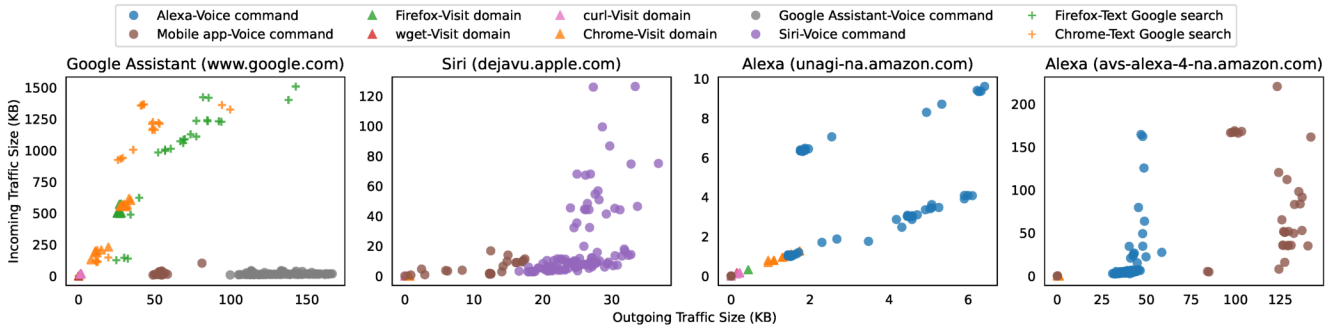

Figure 3: The figure shows the traffic patterns for features we use for invocation detection. We see that invocation traffic from voice assistant commands is very different from other types of traffic and usually invocation detection generates more outgoing traffic and less incoming traffic.

K Nearest Neighbours (KNN), Random Forest, and XGBoost classifier from the scikit-learn [42] and xgboost [13] Python libraries. We input the extracted features into the model, train on the binary labels (1 for invocation and 0 otherwise), and compute the commonly used metrics, such as accuracy, precision, and recall to evaluate the performance.

4.2 Evaluation

We performed 5-fold cross-validation on the dataset to ensure each sample gets to be in the train and test set across runs. We train the classifiers for each of the three voice assistants, and the results for each model and voice assistant are summarized in Table 3. All classifiers perform similarly in detecting invocation, but we found the Random Forest classifier to be the best across all metrics. Hence, we select Random Forest as the default classifier for the end-to-end system evaluation (in Section 6). Our analysis of the misclassifications found that most errors were due to incorrect or improper invocations. For example, one source of error was if the *wake word* was uttered, but no command followed it, or if the *wake word* was uttered and was followed by a command, but the voice assistant did not proceed to process the request.

Voice assistants are much more likely to be in their 'idle' state rather than the 'invoked' state at any given time, resulting in much more non-invocations than invocations. This difference in prior probabilities can lead to a base-rate problem similar to what intrusion detection systems face. In short, even a near-perfect 99% accuracy would mean many false positives due to a higher number of the negative class in the data. To evaluate the extent of this problem, we collect 'idle' traffic from all three voice assistants for about 12 hours with and without traffic from other devices. Traffic from other devices was simulated using web crawling scripts. We used four scripts, one of which opened the Alexa Top 1000 websites and randomly clicked on the landing webpage to generate more traffic than simply loading the page. The other three scripts repeatedly opened `http://amazon.com`,

Assistant	ML Model	Accuracy	Precision	Recall
Alexa	AdaBoost	99.72	99.59	99.86
	XGBoost	99.75	99.59	99.91
	Random Forest	99.81	99.63	100.0
	KNeighbors	99.79	99.59	100.0
Google	AdaBoost	99.65	99.71	99.61
	XGBoost	99.70	99.71	99.71
	Random Forest	99.70	99.71	99.71
	KNeighbors	99.65	99.71	99.61
Siri	AdaBoost	98.95	98.19	99.81
	XGBoost	99.45	99.51	99.42
	Random Forest	99.50	99.71	99.32
	KNeighbors	99.55	99.52	99.61

Table 3: Comparison of different machine learning models for invocation detection on Alexa, Google Assistant, and Siri. While different models perform better with respect to certain metrics. We found Random Forest most consistent across all metrics.

http://google.com, http://apple.com websites and performed random user interactions. Raw PCAP size without traffic from other devices is about 15MB, and with background traffic, it was about 7.2GB. We extract the same features from this 'idle' traffic dataset and use the already trained model to test against this 'idle' traffic trace. We achieved perfect results of 100% accuracy across the three voice platforms with or without noise.

We calculated the time-to-predict, which measures how long it takes to predict an output label given the raw input data, to be, on average, 9.91 ms across all three voice assistant platforms. The time-to-predict counts the total time from converting the raw data to CSV, cleaning and preprocessing, extracting features, and finally using the trained ML model to predict a label on a single-core CPU with no additional hardware accelerators.

Takeaway. We show that inferring the activation of a voice assistant interface through encrypted network traffic is practically feasible across the three popular platforms. We achieve near-perfect results for all three voice assistants. We also show that our results do not suffer from the base-rate problem since traffic in an idle state is dissimilar from traffic immediately following an invocation and, thus, easy to distinguish by a classifier. We also show that background noise does not significantly affect the performance of detecting the invocation of the voice assistants.

5 Activity Detection

Activity detection is the process of inferring the actual command or application (i.e., skill) executed. The workflow to use a voice assistant begins with invoking the voice interface with the wake word and following it with the phrase relevant to the voice command. To achieve this, we extract a fixed duration of traffic, following and including the invocation of the device. Consequently, any traffic before the invocation or after the command expires is unrelated to the command or skill and is typically a source of noise. Such traffic can be eliminated as it usually is general device traffic, e.g., keep-alive connections, pings, and device status messages.

To set an optimal threshold of the amount of traffic or interval to consider for fingerprinting commands, we have to consider the *run time* of the command. Different commands and skills can have diverse *run times*. For example, simple commands usually only last for up to 30 seconds, but streaming commands can often run until explicitly stopped (e.g., Spotify). In such cases, finding an optimal value for *window size* would depend on the command at hand. Considering multitiple different traffic cutoffs (e.g., one, two, and five minutes of traffic), we found that increasing this window size from thirty seconds to two or five minutes would increase the model's performance on streaming commands, but it did not help identify simple commands. To balance the tradeoff, we use a window size of one minute.

5.1 NAT Issue

Since there might be a NAT active on the home router, the adversary would not inherently be able to separate all the traffic from an Amazon Echo (target) device from other devices on that network. This is due to how NAT changes the IP address from individual devices to the router's public IP address, and so, for an out-of-local network observer, all the traffic would be seen coming from the router. An adversary would, however, be able to separate one *flow* of traffic from another because NAT does not change external IP, port, or protocol. Therefore, we define a flow as all traffic with the same set of IPs, ports, and protocols. For IP protocols that do not use ports, we set the port number to 0. We also consider flows to be direction agnostic, i.e., we do not distinguish between incoming and outgoing flows (between the same endpoints). We set the inactive flow timeout, which is the time after a flow is considered to have expired, to 15 seconds. Active flow timeout is set to one minute, the same as our activity detection window, since we do not consider traffic longer than a one-minute duration. Note that our definition of flow does not depend on the flow/connection definition of any particular protocol, so if a TCP connection ends and another connection to the same endpoint (IP, port) starts (e.g., new TCP handshake) within 15 seconds, we consider them as a single-flow instead of two separate flows.

5.2 Separating Voice Assistant Traffic

To isolate the traffic associated with voice assistants from traffic generated by other devices, we use the fact that voice assistants would generate activity-related traffic only after it is invoked. Since we can only see flows without knowing which device inside the *home* network they belong to, our goal is to filter the flows related to voice assistant activity from the others. Traffic related to voice assistant activity could manifest either as a new flow or part of an established flow. The new flows, generally, would start immediately after the invocation of the voice assistants, and we can capture the flows of interest by capturing traffic from all flows that 'begin' in the first m seconds following the invocation and label these as *new flows*, where m is a variable time threshold. This allows us to capture only the more relevant flows to the activity rather than the noisy flows from other devices. The already active flows would be active before the invocation of any particular activity and would therefore be activity agnostic, generally contacting company-specific end-points, e.g., device-metrics-us.amazon.com to gather information about device metrics. To capture traffic from relevant active flows, we created a list of company-specific domains for each voice assistant that was frequently contacted in traffic traces across all commands, e.g., avs-alexa-4-na.amazon.com, unagi-na.amazon.com, api.alexa.com, etc. for Amazon Alexa. We capture all the traffic to these end-points and label

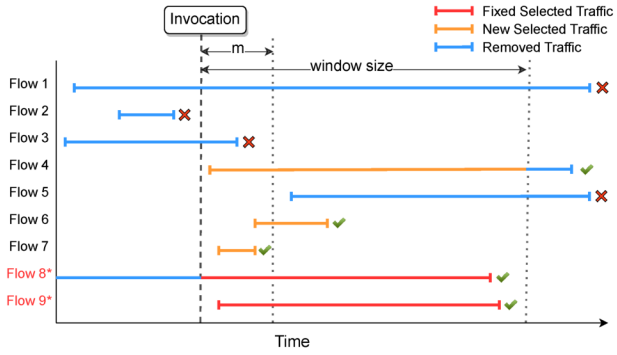

Figure 4: An illustration of our flow separating technique to filter out irrelevant flows (blue). Some flows are fixed and will always be considered (red). Other flows will only be selected (not removed) if they begin within *m* seconds of invocation. Any traffic outside of the *window* will be not considered.

these flows as *fixed flows*. This way, we filter out all traffic other than the *new flows* and *fixed flows*. This technique would allow us to capture most, if not all, of the target device's packets while minimizing the traffic from other devices. This process is visually explained in Figure 4.

In a normal household with multiple voice assistants, it would be uncommon for all devices to be sending a high volume of traffic at the same time as the target voice assistant, and for any of the flows not associated with the voice assistant to start within *m* seconds of it being invoked. The optimal value of *m* depends on the voice assistant and the amount of expected noise. For example, *m=10* is optimal for Siri, whereas *m=5* is optimal for Google Assistant and Amazon Alexa. Setting *m=5* would also filter flows more strictly and can help if the network has too much noise but can remove a few relevant flows (hence reducing performance as well). In our evaluations, we set *m=10* as we analyze all three platforms; however, any value of $5 < m <= 10$ would likely exhibit similar outcomes.

5.3 Feature Extraction

The extracted features can be broadly grouped into single-valued and multi-valued categories. Single-valued features contain various counts and distribution-related characteristics over the entire window, e.g., median inter-packet delay and incoming vs. outgoing traffic ratio. Multi-valued features are the counts or occurrences of different attributes, such as how often a given Hostname is contacted or the counts of packet lengths observed and expressed as a key-value-like structure. For example, in a given window, if nine packets are sent to `domain1`, five packets are sent to `domain2` and four packets are sent to `domain3`, the value of Hostname multi-valued feature would be `{'domain1': 9, 'domain2': 5, 'domain3': 4}`. Since each is essentially a dictionary in the feature vector, we need to encode/serialize them before training a model to convert them to multiple single-valued features.

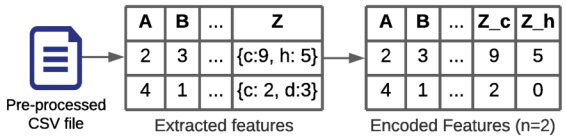

Figure 5: A simple example of feature encoding or serialization from extracted features. The parameter *n* is set to 2 here so value *Z_d* is dropped when encoding because only top 2 would be selected.

When training the model, we select the top *n* pairs with respect to the occurrence on the training set. We then encode the dictionary into multiple single-valued features. The parameter *n* is empirically set to 10 for the count of protocols, 100 for TCP-related features, and 50 for UDP-related features. Figure 5 highlights the encoding process where Features *A* and *B* are regular single-valued features and hence unchanged during the encoding process. Feature *Z* is a multi-valued feature, and hence *top* 2 'keys' are selected based on descending values across all samples and then encoded for each sample. Table 6 in Appendix A provides a list of all features used.

5.4 Machine Learning Pipeline

We use AutoGluon Tabular [19], which is a part of AutoML [9], to train the Machine Learning models. AutoGluon iterates through many different models, including Random Forest variants and Deep Learning models, to automatically optimize the parameters, create ensembles, and select the best-performing model based on training data. The training data is split into training and validation [3], while the test data is used to evaluate the final model's performance.

Training for AutoGluon Tabular without accelerators can take up a considerable amount of time due to the number of models and hyperparameters, but after the training process is complete, the inference or prediction process is faster and can be easily performed on commodity hardware. Since this model would not need to predict as frequently as the invocation detection model, it does not have a similar constraint of being lightweight. We also found that replacing AutoGluon Tabular/AutoML with Random Forest classifier or XGB Classifier usually speeds up the training process with an average of around 2-4% loss in performance. The time-to-predict, from raw traffic data to activity class label, was, on average, 467 ms for AutoGluon Tabular. This time, however, depends on the model that is finally selected as optimum. Our analysis found the time-to-predict to be in the range of 250 ms to 900 ms. To evaluate the quality of the model and performance on test data, we use commonly employed metrics such as average accuracy, precision, and recall.

[3] AutoGluon creates a validation dataset from a part of the training dataset.

Dataset Description			Wang et al. [60]	Mao et al. [33]	Our			# Labels
Assistant	**Type of activity**	**Dataset**	Accuracy	Accuracy	**Accuracy**	**Precision**	**Recall**	
Alexa	Simple commands	DeepVCFingerprint [60]	89.00%	90.36%	**95.70%**	**95.80%**	**95.70%**	100
Alexa	Simple commands	simple_100_alexa	46.30%	52.20%	**80.30%**	**80.58%**	**81.44%**	100
Alexa	Skills	skills_100_alexa	29.22%	36.34%	**82.76%**	**85.33%**	**82.42%**	100
Alexa	Streaming	stream_15_alexa	29.75%	52.45%	**99.39%**	**99.30%**	**99.34%**	15
Alexa	Simple commands	simple_50_alexa	38.40%	42.80%	**87.70%**	**87.46%**	**88.20%**	50
Google	Simple commands	simple_50_google	89.60%	90.60%	**92.67%**	**92.66%**	**92.96%**	50
Siri	Simple commands	simple_50_siri	77.40%	83.09%	**92.80%**	**92.91%**	**93.18%**	50
Alexa	All types with noise	mix_100_alexa	50.79%	58.70%	**81.33%**	**81.28%**	**81.45%**	100

Table 4: Activity detection performance for various datasets. Results show that all three platforms investigated more or less leak information about the types of commands they execute. Streaming commands, in particular, are easier to fingerprint. Our method also outperforms existing approaches.

5.5 Comparison with Existing Works

To evaluate our approach and set a benchmark for performance, we compare it to existing works using the Deep VC Fingerprinting dataset [60]. Previous works [10, 29, 33, 60] have all used this dataset for their evaluations. In particular, we use the closed-world Amazon traces from the dataset with a total of 150,000 samples (1,500 each for 100 simple voice commands). Since the dataset simply contains separate PCAP traces for each of the individual commands and does not mark the timestamp of invocation, we cannot use this dataset for invocation detection or track the flows that start within a certain time period after invocation. We can, however, consider the PCAP files to contain all flows relevant to the voice commands and perform voice command fingerprinting to set a benchmark for this dataset. Table 4 compares the performance on fingerprinting 100 closed-world Amazon Echo commands from previous works. On this dataset, our model obtains an accuracy of 95.70% using the same train and test split (90:10 for training and testing), which is an improvement over the current state-of-the-art reported accuracy of 93.36%. This shows that our model outperforms the current state of the art, and thus we set a baseline for our approach. We will also compare existing approaches on our datasets to demonstrate significant improvement over the existing state of the art. [4]

Previous works have only considered individual packet size, direction, and timing-based features. However, endpoints, protocols, and flow-level information can provide additional insights, such as which domains are being contacted in each activity and the distribution of traffic per flow/burst rather than the overall traffic. This additional information can provide the classifier with additional context and make the classifier more robust in the presence of background noise. For example, activating a certain Alexa skill might contact an endpoint unique to that skill or a set of skills and can identify the skill developer's backend. Various feature importance/ranking metrics such as ANOVA, Mutual Information and Random Forest's Gini index also rank these additional features (e.g., flow-based and burst-based features) more frequently in the top 10 features. More details available in Table 7 in Appendix C. The usefulness of such features becomes even more evident when we evaluate existing approaches on our dataset containing a more realistic setup containing background noise. In fact, we obtain a performance improvement in the range of 2% to 69% compared to previous works. On average, across all datasets, we obtain a performance improvement of 25.76% from Mao et al. [33] and 32.77 % from Wang et al. [60]. The results for each dataset are shown in Table 4. Thus, our approach outperforms the current state of the art and establishes a new baseline in a more realistic setup.

[4] For Mao et al., [33] we tried our best to replicate their proposed deep learning models, but some parameters were not provided (e.g., input dimension, learning rate, optimizers, and activations), and thus we were not able to exactly replicate their performance. Similarly, for Wang et al., [60], we had a performance drop of 2-3% compared to the reported performance. However, as we show on our realistic datasets, we outperform by around 2% to 69%.

5.6 Comparing Voice Assistant Platforms

To compare the extent to which different voice assistant platforms are vulnerable to voice command fingerprinting, we evaluated three platforms (i.e., Alexa, Google Assistant, and Siri). We used the traffic trace of 50 commands collected from Nest Mini (Google Assistant), HomePod mini (Siri), and a set of 50 similar commands for Alexa. The commands were similar for each dataset, and we only made minimal changes to the command set when an assistant did not properly respond to a command, i.e., did not respond at all or replied with a non-trivial or unexpected response.

Table 4 shows the performance across all three voice assistants. We see that all voice assistants are similarly vulnerable to voice fingerprinting attacks. The model performs slightly better for Google Assistant and Siri compared to Alexa. We found that most inaccuracies are due to commands that generate a similar response from the voice assistants, e.g., setting and cancelling alarms are often mislabeled. We found another source of errors to be the commands which initiate an internet 'web' search by the smart speaker, and the response is usually generic, e.g., "Here is what I found on the web for..."

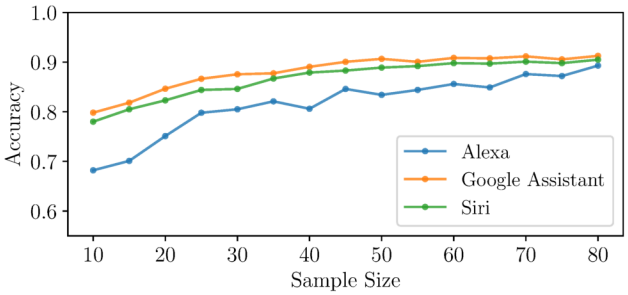

Figure 6: Impact of training sample size on accuracy across three platforms.

We compute and compare the mutual information gained across each feature to understand what feature groups are important and provide the most information for each voice assistant platform. Table 8 in Appendix C highlights the feature ranking based on mutual information across the three platforms. We see that flow times, lengths, and packet lengths are often the most informative features among all features (full feature list available in Appendix A). Features such as IP, Hostname, and External Ports are ranked towards the bottom since, for simple commands, platforms only contact generic cloud-based end hosts and serve content directly without depending much on third-party content. However, when analyzing other forms of commands, such as streaming services, Hostname provides a significant amount of mutual information (more details in Section 5.7).

Impact of Training Size. We vary the sample size for each class and train AutoGluon Tabular classifier for these varying sample sizes to understand the number of sample sizes needed to obtain stable performance. Figure 6 shows the evolution of accuracy for inferring the activity as a function of sample size. We used a standard 80:20 train:test split of the data in each case and sampled the given amount from the train set while evaluating performance on the test set. After approximately 60 samples (i.e., around 48 training samples), the performance stabilizes with minimal performance gain for a larger training set.

5.7 Comparison of Different Command Types

To go one step ahead and understand if different types of commands affect the ability to fingerprint, we consider different voice commands and the resulting activities performed on the voice assistant device. For this analysis, we use Alexa, the most popular home voice assistant device by market share, and provides the most diverse set of voice commands and third-party skills. As previously mentioned, we collect three different kinds of activities: simple commands, streaming commands, and skills for Alexa. Then, we extract features and use our machine-learning pipeline to compute the results. The results for each category of commands are reported in Table 4. The results show that we get high accuracy for streaming commands (99.39%), among other contributing factors, due to different commands originating from different vendors and usually having unique endpoints to serve the streaming content directly to the device [25]. Hostname and IP multi-valued features also rank higher (compared to other types) in feature ranking given in Table 8 of Appendix C based on mutual information. For skills, we often found them contacting generic Amazon/AWS endpoints or other general skill hosting services (e.g., Voice Apps [2]).

We analyzed the mislabels from the classifier and found that skills of similar functionality developed by the same developer were mistaken for each other. For example, the following skill pairs: ("Morning Sikh Prayer" and "Evening Sikh Prayer") and ("Bird Sounds" and "Rain Sounds") created confusion for the classifier. In both cases, the same vendor posted these skills pairs to Amazon Skill Store and had similar functionality. Another set of skills that were often predicted as each other were fact-based skills such as "Dog Facts", "Cat Facts", "Unofficial Chuck Norris facts", and "Creepy factoids." These skills have practically identical traffic patterns when asked for a fact, and they reply with a fact randomly from a selection of facts before returning control to Alexa. Upon further investigation, we also found that commands with a similar meaning or similar trigger-activity response created misclassification errors. For example, commands such as "add milk to shopping cart," "add pork to my shopping cart" or "remove bananas from my shopping cart" were often mistaken for one another. We also found that asking Alexa "how to deal with anxiety" and "how to feel less depressed" triggered the same response from Alexa and hence resulted in incorrect prediction by the classifier. We provide more examples in Table 5. The performance of our model on different types of commands and skills shows that, in general, the model can accurately identify commands. We further evaluated any potential loss in performance due to potentially losing some traffic when filtering out traffic not from *fixed* or *new* flows. We found the effect negligible (less than 1%) and an acceptable overhead in all cases. To understand any difference in performance due to background noise, we used the *mix_100_alexa* dataset to evaluate the performance of Activity Detection in Section 6.

5.8 Generalizability Analysis

To understand the generalizability of our developed approach across different networks and hardware (i.e., a different voice assistant from the same vendor), we collected a dataset of 50 simple commands (i.e., the same commands used in *simple_50_alexa* dataset) using a different Alexa voice assistant device (Amazon Echo Dot 3rd Gen) deployed in a different network operated by a different ISP (i.e., a home network as opposed to the lab network; no network configuration changes were made to the home network). We then used the classifier trained on the *simple_50_alexa* dataset collected from our

Skills Pair	Count
skill: Daily Evening Sikh Prayer* ↔ Morning Sikh Prayer*	16
skill: Dog Facts ↔ Unofficial Chuck Norris Facts	9
skill: Buddha Sense ↔ Me an Interview Question	8
skill: Cat Facts ↔ Dog Facts	7
Skill: Rain Sounds* ↔ Bird Sounds*	7
Simple Command Pair	**Count**
where are the closest therapist? ↔ where can I find a divorce lawyer	13
tell me a Hannukah joke ↔ tell me a halloween joke	10
what are good date spots around ↔ where is the closest sperm bank?	9
how to deal with diabetes ↔ where can I find a divorce lawyer	9
add bananas to my shopping list ↔ remove milk from my shopping list	8

Table 5: Top misclassified pairs in Alexa skills and commands. Skills marked with '*' have same developer.

lab (as listed in Table 2) to fingerprint the activities from this new dataset. The model achieved an accuracy of 83.81% with a precision and recall of 83.52% and 83.87%, respectively. We see a slight drop of 3.89% when trained and tested on the same dataset (the accuracy was 87.7% as shown in Table 4). The errors follow a similar pattern where commands of similar nature are mislabeled as each other. For example, the top classification error was the command "what are good date spots around" being labeled as "what are good hotels in Las Vegas". This evaluation shows that the model generalizes quite well and can be deployed across ISPs and devices manufactured by the same vendor.

Takeaway. We show we can fingerprint sensitive voice commands effectively and with high accuracy across all three voice assistant platforms. Further, we show that different types of voice commands and skills have slight differences in how easy/difficult it is to fingerprint them. Similar commands and skills have similar traffic and can be easily wrongly labeled as each other. Finally, we demonstrate that our approach generalizes across different networks and devices.

6 End-to-End Detection

We design an end-to-end classification system [5] for real-world analysis and inference of traffic from a voice assistant. The system trains separate invocation and activity detection models as described in the previous sections. Figure 7 highlights the overall process. The end-to-end system consists of a software control loop that captures 2 seconds of network traffic from the router and stores it in a queue-like structure. It then uses 4 seconds of traffic and extracts invocation detection-related features and provides them to an already trained invocation detection model. The invocation detection model then provides a binary prediction of *invocation* detected or not. If no invocation is detected, the loop goes to the next iteration and provides the next 2 seconds of traffic. However, if an invocation is detected, the next 56 seconds of traffic is used to compose a traffic window of 60 seconds to infer the actual activity on the voice assistant as described in Section 3. Using the one-minute traffic, we perform *flow filtering* to extract the *fixed* and *new* flows. Next, we extract the features for activity detection from these flows and use the pre-trained AutoGluon Tabular [19] classifier to predict the activity.

For evaluation, we used the *mix_100_alexa* dataset, which has traffic from a combination of 100 different Alexa commands and skills. The dataset contains a mixed total of 100 different simple voice commands, streaming commands, and skills. The dataset also contains *background traffic* generated randomly from desktop devices, which will overlap with traffic from Amazon Echo (Alexa voice assistant) and create noisy traffic. This allows us to evaluate the performance of both the end-to-end model and the effectiveness of the flow-separating technique to differentiate flows from the voice assistant to the flows from other internet-connected devices. We used the first 80% data to train and optimize the classifier and used the latter 20% to evaluate the performance. We did not use the 5-fold cross-validation or random sampling to ensure the results were not biased due to temporal factors such as training samples temporally sandwiched between the test data samples.

To first isolate and understand any difference in performance due to background noise, we evaluate invocation detection and activity detection separately. Using the Random Forest classifier for invocation detection, we got 98.70% accuracy, 97.77% precision, and 99.68% recall, respectively. For activity detection, we achieved 81.33% accuracy, 81.28% precision, and 81.45% recall, respectively. This shows that the system performs well due to our flow filtering process, even in the presence of relatively high background noise. Finally, we combine the two models (as shown in Figure 7) and create our end-to-end system. The end-to-end system has an accuracy, precision, and recall of 79.91%, 80.34%, and 79.30%, respectively. The average time-to-predict if Alexa was invoked is 9.9 ms (i.e., only the invocation detection model). However, the average time to predict the exact voice command requires 891 ms (excluding the one-minute wait to obtain activity traffic). This shows that our end-to-end system can infer voice assistant activity in under one second.

Analysis of misclassifications in end-to-end scenarios shows that most inaccuracies are due to similar commands or skills being mislabeled as one another (similar to Table 5), especially ones that have similar functionality. We also found an interesting case where an exercise skill ('6-Minute Full body stretch') was labeled as a travel skill ('MyVoiceTravel') and vice versa. Upon further investigation, we found that due to our data collection only invoking the skill by only the 'initial' invocation phrases and not performing deeper interactions (e.g., replying to if you want to start a workout or not), we had some cases of skills just asking one prompt and then exiting upon not hearing a response from the user. This is a limitation of our data collection approach, and in the future, we plan to explore deeper automated interaction with skills to potentially

[5] A short demo is available at: https://youtu.be/2Dv8cSkvC1g

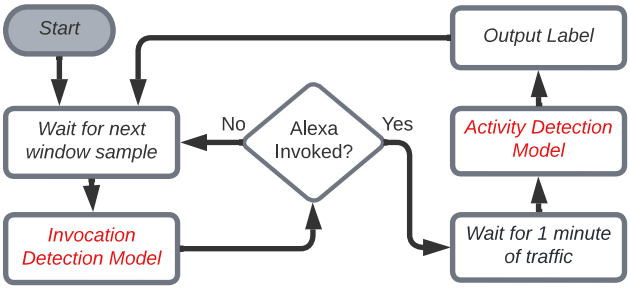

Figure 7: The state diagram of how the end-to-end system works. The input is a stream of packets fed into the system every 2 seconds. Upon detecting invocation, the system waits for 1 minute of traffic to distinguish between new flows and ongoing flows to determine the actual voice command.

gain more meaningful fingerprints.

Real-world End-to-end Evaluation. To confirm our findings in a real-world setting, we perform a small-scale IRB-approved user study over five sessions of one hour each spread across five days. We used an Amazon Alexa voice assistant for this experiment and set it up in our lab. We set up three additional devices to the lab network that we monitor via an upstream router: a laptop (Lenovo AMD Ubuntu Laptop), a mobile phone (Google Pixel 3a), and a smart TV (LG WebOS TV) to act as other household devices and generate realistic noise when participants use and interact with these devices. We instructed the participants in our study to use these devices as they would use them in their own homes (e.g., watch Netflix on a preinstalled account, surf the web, or play video games online, etc.). However, for ethical considerations, we instructed participants not to connect their personal devices to the lab network or use sensitive applications (e.g., logging into email, social networks or financial accounts) on the devices we provided. We also avoided privacy-sensitive commands in this study since participants may feel uneasy saying or listening to the responses to these commands in the presence of others. We had an average of three participants in each session (for each one-hour session, participants were paid $10). In the sessions, once participants began using the devices, we asked the participants to give a command every few minutes (on average approximately 2-3 minutes) from a list of 15 randomly selected commands (which contained 6 simple commands, 5 skills, 4 streaming commands) selected from the *mix_100_alexa* dataset. We also asked the participants to give a random command not from the list to evaluate the performance of invocation detection on unseen commands. For evaluation, we used the invocation and activity detection models trained on the *mix_100_alexa* dataset.[6]

Participants invoked the voice assistant, in total, 80 times across the five sessions for the closed-world end-to-end evaluation, each time choosing a random command from the 15 unique known commands. The number of invocations for each command was unbalanced since participants were randomly choosing a command; however, we ensured that each command was at least invoked three times. Participants were also asked to invoke voice assistants on unseen random commands 20 times in total across the five sessions, and we found that in all cases, the invocation was correctly detected. For detecting the exact voice command, we trained a multi-class classifier on 'known' commands using the *mix_100_alexa* dataset and used the classifier's confidence score (i.e., the *argmax* of the prediction probability) to distinguish if a test sample is 'known' versus 'unknown' as commonly adopted by existing literature [6,12,53]. For the 'unknown' dataset, we utilized our other datasets, e.g., *simple_100_alexa*, *skills_100_alexa*, and *stream_15_alexa* datasets. Combined, these datasets consist of 215 commands, 100 overlapping with the *mix_100_alexa* dataset and hence tagged as 'known', and the remaining 115 can be considered 'unknown'. Using a ROC curve, we determined the optimal threshold value as 0.26 (i.e., minimizes false positive and maximizes true positive). The threshold value is defined based on the confidence score of the multi-class classifier, and if the value is greater than the threshold for a given sample, the sample is considered as 'known' and otherwise as 'unknown'. Using this threshold value, we achieved an average accuracy of 90% in distinguishing known samples from unknowns, with precision and recall being 92% and 91%, respectively.

Next, we used the multi-classifier trained on the *mix_100_alexa* dataset to predict the samples classified as 'known'. For the end-to-end classification accuracy, we consider both the successful filtering of unknown samples and the successful classification of known samples into their respective class labels for any input. Out of the 80 'known' commands, 73 were predicted as known and 7 as unknown. Among the commands correctly predicted as known, 63 were also correctly classified into their respective classes, whereas 10 were incorrectly classified. Thus, 63 of the 80 known commands were correctly classified. Of the 20 unknown commands, 6 were classified as known and 14 as unknown. The final end-to-end accuracy can then be computed as $(63+14)/100 = 77/100 = 77\%$. In some cases, these output labels on unseen commands were closer to real activities, e.g., the 'Rain Sounds' skill (unseen) was marked as 'Zen Sounds' (seen), and set a timer for one minute (unseen) was marked as set a timer for two minutes (seen). During this experiment, per command, there was between 8 to 80 times more traffic going through the router as background noise compared to the dataset we collected without noise.

Takeaway. We show that our proposed filtering of active and new flows enables us to fingerprint voice commands even in the presence of background traffic from other devices on the same network, as demonstrated through both simulations of background traffic and real users generating traffic while using other devices. Furthermore, we demonstrate that by utilizing

[6] If we trained the activity detection classifier to only focus on the 15 commands we get 92.5% accuracy.

the confidence score of a multi-class classifier, it is possible to establish a threshold that can effectively distinguish 'known' commands from 'unknown' under real-world settings.

7 Discussion

Potential Countermeasures. Voice command fingerprinting relies on encrypted network traffic analysis. While the literature on preventing voice command inference is minimal, there is extensive literature on preventing general traffic analysis, such as website fingerprinting. Most of these defenses should also (with minimal changes) effectively reduce the performance of our inference attack. Our attack, in particular, depends upon the ability of the adversary to distinguish network flows. Any countermeasure where flow information is unavailable to the adversary, such as a home VPN, would be effective against our attack. However, consumers typically do not opt for home-level VPNs as they require additional device configuration [14]. Other popular countermeasures include traffic padding and shaping [11, 27, 32, 35, 62], where the goal is to make all traffic similar to each other and hence reduce the information gained by the adversary. These countermeasures would typically also employ some variation of a VPN-like tunnel to ensure individual connection information is unavailable to the adversary. However, such defenses result in high overheads in terms of bandwidth and latency, ranging from 40% to over 100% or efficacy [35], thus questioning the real-time nature of voice assistants.

Another class of countermeasures relies on injecting adversarial noise to fool classifiers [37, 45]. However, these defenses are not foolproof and can be vulnerable as well [35]. Researchers have also looked at splitting traffic over multiple networks (such as WiFi and cellular) [17, 22]. Such a defense promises little to no overhead but is difficult to adopt as consumer homes usually have one ISP. The most promising and easy-to-apply approach for voice command fingerprinting attacks is similar to the 'k-anonymity' approach proposed by previous research in the context of website fingerprinting [38, 61]. Since most voice commands are neatly grouped based on their network usage (such as simple short commands, streaming services, etc.), it is easy to shape them to match one another with low overheads. Such an approach will only leak which group a command belongs to but not what that command is. We leave the implementation and evaluation of such an approach for voice assistants to future work.

Limitations. Our work focuses on the top three most popular voice assistants, and we leave other off-the-shelf voice assistants from this analysis. Our approach also relies on manual observation to select the endpoints to track the invocation detection. These domains are sometimes region-specific (e.g., have a 'na' keyword in them due to architectural decisions for different regions [1]), meaning they might only be used for devices in the specific region. Outside these regions, devices may contact their region-specific domains. However, Ren et al. [47] showed that despite being in different regions, the device still contacts mostly US-based endpoints. Regardless we expect other region devices to contact these endpoints or their region-specific endpoints, which would serve the same purpose. Our work lacks a comprehensive open-world evaluation for voice commands. We leave such exploration as future work. Lastly, our approach requires isolating traffic from different flows, but this would not be possible if there is a VPN or something similar. However, VPNs for IoT devices are not widespread, and evaluation of voice command fingerprinting attacks under VPN conditions is left as future work.

8 Conclusion

In this work, we show that it is possible to fingerprint different types of voice commands and skills on Amazon Echo (Alexa) ecosystem with high accuracy. We also show that it is also possible to fingerprint voice commands on different platforms, including Google Assistant and Siri, with a similar, if not better, performance. Using a novel time-series feature engineering-based approach, we improve the state-of-the-art performance on existing datasets and further demonstrate the effectiveness of our approach under realistic settings. Using a lightweight machine learning model for invocation detection, we achieved almost perfect accuracy in detecting when the voice assistants were activated without being plagued by a base-rate problem. Furthermore, we show that even when the adversary is out of the local network, we can achieve similar performance on activity detection using invocation detection and a novel method using our flow filtering technique for each voice assistant. Finally, we show that invocation detection models and activity detection models can be combined to create an end-to-end classification system that can predict activities under real-world performance constraints with a high amount of background noise.

Acknowledgement

We thank our anonymous reviewers for their valuable feedback and our shepherd for helping us refine our work. This material is based upon work supported in parts by the National Science Foundation under grant number CNS-1849997 and the Center for Accelerated Real Time Analytics (CARTA) - NCSU Research Site. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] "Alexa Voice Service v20160207 | Alexa Voice Service." [Online]. Available: https://developer.amazon.com/en-US/docs/alexa/alexa-voice-service/api-overview.html

[2] "Voice apps lets you create alexa skills in minutes! no coding required!" [Online]. Available: https://voiceapps.com/

[3] 1st Source Bank, "Amazon.com: 1st Source Bank : Alexa Skills." [Online]. Available: https://www.amazon.com/1st-Source-Bank/dp/B07M8651TJ

[4] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and S. Uluagac, "Peek-a-boo: I see your smart home activities, even encrypted!" in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020, pp. 207–218.

[5] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Multi-classification approaches for classifying mobile app traffic," *Journal of Network and Computer Applications*, vol. 103, pp. 131–145, 2018.

[6] D. Ahmed, A. Das, and F. Zaffar, "Analyzing the Feasibility and Generalizability of Fingerprinting Internet of Things Devices," *Proceedings on Privacy Enhancing Technologies (PETS)*, no. 2, 2022.

[7] H. F. Alan and J. Kaur, "Can android applications be identified using only tcp/ip headers of their launch time traffic?" in *Proceedings of the 9th ACM conference on security & privacy in wireless and mobile networks*, 2016, pp. 61–66.

[8] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan, and N. Feamster, "Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic," *arXiv:1708.05044 [cs]*, August 2017. [Online]. Available: http://arxiv.org/abs/1708.05044

[9] AutoML.org, "AutoML." [Online]. Available: https://www.automl.org/automl/

[10] C. Batyr and M. H. Gunes, "Voice Command Fingerprinting with Locality Sensitive Hashes," in *Proceedings of the Joint Workshop on CPS&IoT Security and Privacy (CPSIOTSEC '20)*, New York, NY, USA, November 2020, pp. 87–92.

[11] X. Cai, R. Nithyanand, and R. Johnson, "CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, Scottsdale Arizona USA, November 2014, pp. 121–130.

[12] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: website fingerprinting attacks and defenses," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Securit (CSS '12)*, 2012, pp. 605–616.

[13] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, New York, NY, USA, August 2016, pp. 785–794.

[14] C. Cimpanu, "Survey reveals users have no clue about router security," Apr 2018. [Online]. Available: https://www.bleepingcomputer.com/news/security/survey-reveals-users-have-no-clue-about-router-security/

[15] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Analyzing android encrypted network traffic to identify user actions," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 114–125, 2015.

[16] B. Copos, K. Levitt, M. Bishop, and J. Rowe, "Is Anybody Home? Inferring Activity From Smart Home Network Traffic," in *IEEE Security and Privacy Workshops (SPW)*, May 2016, pp. 245–251.

[17] W. De la Cadena, A. Mitseva, J. Hiller, J. Pennekamp, S. Reuter, J. Filter, T. Engel, K. Wehrle, and A. Panchenko, "Trafficsliver: Fighting website fingerprinting attacks with traffic splitting," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1971–1985.

[18] M. Elena, "AI Voice Assistants: 9 Key Predictions For The Future Of Technology," August 2021. [Online]. Available: https://masterofcode.com/blog/9-key-predictions-for-the-future-of-voice-assistants

[19] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola, "AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data," *arXiv:2003.06505 [cs, stat]*, March 2020. [Online]. Available: http://arxiv.org/abs/2003.06505

[20] T. Group, "tcpdump." [Online]. Available: https://www.tcpdump.org/

[21] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *Proceedings of the 25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 1187–1203.

[22] S. Henri, G. García, P. Serrano, A. Banchs, and P. Thiran, "Protecting against website fingerprinting with multihoming," *Proceedings on Privacy Enhancing Technologies (PETS)*, no. 2, pp. 89–110, 2020.

[23] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naive-bayes classifier," in *Proceedings of the ACM workshop on Cloud Computing Security (CCSW '09)*, 2009, pp. 31–42.

[24] V. Inc., "Jeopardy!" [Online]. Available: https://www.amazon.com/Sony-Pictures-Television-Jeopardy/dp/B019G0M2WS

[25] U. Iqbal, P. N. Bahrami, R. Trimananda, H. Cui, A. Gamero-Garrido, D. Dubois, D. Choffnes, A. Markopoulou, F. Roesner, and Z. Shafiq, "Your Echos are Heard: Tracking, Profiling, and Ad Targeting in the Amazon Smart Speaker Ecosystem," *arXiv:2204.10920 [cs]*, April 2022. [Online]. Available: http://arxiv.org/abs/2204.10920

[26] L. Jessica, "US Voice Assistants and Smart Speakers Forecast 2022," Tech. Rep., 2022. [Online]. Available: https://www.insiderintelligence.com/content/us-voice-assistants-smart-speakers-forecast-2022

[27] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," in *European Symposium on Research in Computer Security*, 2016, pp. 27–46.

[28] B. Katrina, "Infographic: The Most Popular Smart Speakers in the U.S." January 2022. [Online]. Available: https://www.statista.com/chart/23943/share-of-us-adults-who-own-smart-speakers/

[29] S. Kennedy, H. Li, C. Wang, H. Liu, B. Wang, and W. Sun, "I Can Hear Your Alexa: Voice Command Fingerprinting on Smart Home Speakers," in *IEEE Conference on Communications and Network Security (CNS)*, June 2019, pp. 232–240.

[30] D. Kumar, R. Paccagnella, P. Murley, E. Hennenfent, J. Mason, A. Bates, and M. Bailey, "Skill squatting attacks on amazon alexa," in *Proceedings of the 27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Aug. 2018, pp. 33–47.

[31] C. Lentzsch, S. J. Shah, B. Andow, M. Degeling, A. Das, and W. Enck, "Hey alexa, is this skill safe?: Taking a closer look at the alexa skill ecosystem," in *28th Annual Network and Distributed System Security Symposium (NDSS 2021)*, 2021.

[32] D. Lu, S. Bhat, A. Kwon, and S. Devadas, "DynaFlow: An Efficient Website Fingerprinting Defense Based on Dynamically-Adjusting Flows," in *Proceedings of the Workshop on Privacy in the Electronic Society (WPES '18)*, New York, NY, USA, 2018, pp. 109–113.

[33] J. Mao, C. Wang, Y. Guo, G. Xu, S. Cao, X. Zhang, and Z. Bi, "A novel model for voice command fingerprinting using deep learning," *Journal of Information Security and Applications*, vol. 65, p. 103085, March 2022.

[34] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, "AuDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1402–1412, June 2019.

[35] N. Mathews, J. K. Holland, S. E. Oh, M. S. Rahman, N. Hopper, and M. Wright, "Sok: A critical evaluation of efficient website fingerprinting defenses," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE Computer Society, 2023, pp. 344–361.

[36] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT," in *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 2177–2184.

[37] M. Nasr, A. Bahramali, and A. Houmansadr, "Defeating DNN-Based Traffic Analysis Systems in Real-Time With Blind Adversarial Perturbations," in *Proceedings of the 30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2705–2722.

[38] R. Nithyanand, X. Cai, and R. Johnson, "Glove: A Bespoke Website Fingerprinting Defense," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, Scottsdale Arizona USA, November 2014, pp. 131–134.

[39] T. OConnor, R. Mohamed, M. Miettinen, W. Enck, B. Reaves, and A.-R. Sadeghi, "HomeSnitch: behavior transparency and control for smart home IoT devices," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '19)*, 2019, pp. 128–138.

[40] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, "Website Fingerprinting at Internet Scale," in *Proceedings 2016 Network and Distributed System Security Symposium*, San Diego, CA, 2016.

[41] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th ACM workshop on Privacy in the electronic society (WPES '11)*, 2011, pp. 103–114.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[43] R. Perdisci, T. Papastergiou, O. Alrawi, and M. Antonakakis, "IoTFinder: Efficient Large-Scale Identification of IoT Devices via Passive DNS Traffic Analysis," in *IEEE European Symposium on Security and Privacy (EuroS&P)*, September 2020, pp. 474–489.

[44] E. Petagna, G. Laurenza, C. Ciccotelli, and L. Querzoni, "Peel the onion: Recognition of android apps behind the tor network," in *International Conference on Information Security Practice and Experience*. Springer, 2019, pp. 95–112.

[45] M. S. Rahman, M. Imani, N. Mathews, and M. Wright, "Mockingbird: Defending Against Deep-Learning-Based Website Fingerprinting Attacks With Adversarial Traces," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1594–1609, 2021.

[46] A. Reed and M. Kranch, "Identifying https-protected netflix videos in real-time," in *Proceedings of the 7th ACM on Conference on Data and Application Security and Privacy*, 2017, pp. 361–368.

[47] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi, "Information Exposure From Consumer IoT Devices: A Multidimensional, Network-Informed Measurement Approach," in *Proceedings of the Internet Measurement Conference (IMC '19)*, October 2019, pp. 267–279.

[48] A. Sabir, E. Lafontaine, and A. Das, "Hey alexa, who am i talking to?: Analyzing users' perception and awareness regarding third-party alexa skills," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. Association for Computing Machinery, 2022.

[49] "Intriguing Amazon Alexa Statistics You Need to Know in 2022," Safe at Last, February 2022. [Online]. Available: https://safeatlast.co/blog/amazon-alexa-statistics/

[50] S. J. Saidi, A. M. Mandalari, R. Kolcun, H. Haddadi, D. J. Dubois, D. Choffnes, G. Smaragdakis, and A. Feldmann, "A haystack full of needles: Scalable detection of iot devices in the wild," in *Proceedings of the ACM Internet Measurement Conference (IMC '20)*, 2020, pp. 87–100.

[51] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, and J. Qian, "Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic," in *10th USENIX Workshop on Offensive Technologies (WOOT 16)*, 2016.

[52] T. S. Saponas, J. Lester, C. Hartung, S. Agarwal, and T. Kohno, "Devices that tell on you: Privacy trends in consumer ubiquitous computing," in *Proceedings of the 16th USENIX Security Symposium (USENIX Security 07)*. Boston, MA: USENIX Association, Aug. 2007.

[53] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*, New York, NY, USA, 2018, pp. 1928–1943.

[54] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, August 2019.

[55] D. Song, "Timing analysis of keystrokes and ssh timing attacks," in *Proceedings of the 10th USENIX Security Symposium (USENIX Security 01)*, 2001.

[56] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic," in *IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 439–454.

[57] R. Trimananda, J. Varmarken, A. Markopoulou, and B. Demsky, "Packet-level signatures for smart home devices," in *Proceedings of Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, 2020.

[58] tshark, "tshark." [Online]. Available: https://tshark.dev/

[59] T. van Ede, R. Bortolameotti, A. Continella, J. Ren, D. J. Dubois, M. Lindorfer, D. Choffnes, M. van Steen, and A. Peter, "Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic," in *Network and Distributed System Security Symposium (NDSS)*, vol. 27, 2020.

[60] C. Wang, S. Kennedy, H. Li, K. Hudson, G. Atluri, X. Wei, W. Sun, and B. Wang, "Fingerprinting encrypted voice traffic on smart speakers with deep learning," in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec 20)*, New York, NY, USA, July 2020, pp. 254–265.

[61] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proceedings of the 23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 143–157.

[62] T. Wang, I. Goldberg *et al.*, "Walkie-talkie: An efficient defense against passive website fingerprinting attacks." in *Proceedings of the 26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1375–1390.

[63] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson, "Uncovering spoken phrases in encrypted voice over ip conversations," *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 4, pp. 1–30, 2010.

[64] C. V. Wright, L. Ballard, F. Monrose, and G. M. Masson, "Language identification of encrypted VOIP traffic: Alejandra y roberto or alice and bob?" in *Proceedings of the 16th USENIX Security Symposium (USENIX Security 07)*, vol. 3, 2007, pp. 43–54.

[65] Y. Xu, T. Wang, Q. Li, Q. Gong, Y. Chen, and Y. Jiang, "A Multi-tab Website Fingerprinting Attack," in *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC '18)*, December 2018, pp. 327–341.

Appendix

A Full Feature List

Table 6: List of all features considered with their respective groups.

Total Percentage	Packet Length	Unique Packet Length	Flow Packet Count	Burst Packet Count	Inter-Packet Delay
out_icmp_percentage	in_75per_len	in_len_uniquelen	out_mean_flownumpkts	in_max_burstnumpkts	in_mean_interpktdelay
out_tcprst_percentage	out_mean_len	in_median_uniquelen	in_min_flownumpkts	in_median_burstnumpkts	out_min_interpktdelay
out_tcppsh_percentage	in_max_len	out_len_uniquelen	out_std_flownumpkts	out_min_burstnumpkts	out_median_interpktdelay
out_tcpfin_percentage	in_mean_len	out_90per_uniquelen	out_min_flownumpkts	in_90per_burstnumpkts	in_90per_interpktdelay
in_tcp_percentage	out_min_len	out_min_uniquelen	in_75per_flownumpkts	in_std_burstnumpkts	in_std_interpktdelay
in_udp_percentage	in_25per_len	out_median_uniquelen	out_75per_flownumpkts	in_10per_burstnumpkts	out_90per_interpktdelay
out_tcpack_percentage	out_max_len	in_25per_uniquelen	out_max_flownumpkts	in_75per_burstnumpkts	out_max_interpktdelay
out_percentage	in_10per_len	out_mean_uniquelen	out_median_flownumpkts	out_10per_burstnumpkts	in_75per_interpktdelay
out_tcp_percentage	in_median_len	in_90per_uniquelen	in_10per_flownumpkts	out_mean_burstnumpkts	in_median_interpktdelay
in_percentage	out_25per_len	in_max_uniquelen	in_median_flownumpkts	in_mean_burstnumpkts	out_mean_interpktdelay
in_tcpfin_percentage	out_median_len	in_75per_uniquelen	in_90per_flownumpkts	out_max_burstnumpkts	out_75per_interpktdelay
in_tcppsh_percentage	in_min_len	out_max_uniquelen	out_25per_flownumpkts	out_90per_burstnumpkts	in_25per_interpktdelay
out_udp_percentage	in_len_len	in_10per_uniquelen	in_mean_flownumpkts	in_min_burstnumpkts	out_25per_interpktdelay
in_tcprst_percentage	in_90per_len	out_25per_uniquelen	in_max_flownumpkts	out_median_burstnumpkts	in_min_interpktdelay
in_tcpurg_percentage	out_10per_len	out_10per_uniquelen	out_10per_flownumpkts	out_std_burstnumpkts	out_10per_interpktdelay
out_dns_percentage	out_90per_len	out_75per_uniquelen	in_25per_flownumpkts	in_25per_burstnumpkts	in_max_interpktdelay
out_tcpurg_percentage	out_len_len	in_mean_uniquelen	out_90per_flownumpkts	out_75per_burstnumpkts	in_10per_interpktdelay
out_tcpsyn_percentage	in_std_len	out_std_uniquelen	in_std_flownumpkts	out_25per_burstnumpkts	out_std_interpktdelay
in_tcpack_percentage	out_75per_len	in_min_uniquelen	**Flow Length**	**Burst Length**	**Inter-Burst Delays**
in_dns_percentage	out_std_len	in_std_uniquelen	in_75per_flowbytes	out_90per_burstbytes	in_25per_interburstdelay
in_tcpsyn_percentage	**Burst Time**	**Flow Time**	out_10per_flowbytes	in_90per_burstbytes	out_25per_interburstdelay
in_icmp_percentage	out_75per_bursttime	out_25per_flowtime	in_mean_flowbytes	in_25per_burstbytes	out_median_interburstdelay
Packet Lengths	in_min_bursttime	in_25per_flowtime	out_90per_flowbytes	in_mean_burstbytes	out_75per_interburstdelay
out_tcp_dict_packetlens	out_90per_bursttime	in_std_flowtime	out_std_flowbytes	out_median_burstbytes	out_mean_interburstdelay
out_all_dict_packetlens	in_25per_bursttime	in_10per_flowtime	in_25per_flowbytes	in_min_burstbytes	out_10per_interburstdelay
in_udp_dict_packetlens	in_median_bursttime	out_median_flowtime	in_90per_flowbytes	out_25per_burstbytes	in_median_interburstdelay
out_udp_dict_packetlens	out_max_bursttime	out_10per_flowtime	in_10per_flowbytes	out_max_burstbytes	in_90per_interburstdelay
in_tcp_dict_packetlens	out_10per_bursttime	out_std_flowtime	out_75per_flowbytes	in_75per_burstbytes	in_75per_interburstdelay
in_all_dict_packetlens	out_25per_bursttime	in_min_flowtime	out_mean_flowbytes	out_mean_burstbytes	out_min_interburstdelay
External Counts	out_mean_bursttime	out_min_flowtime	in_min_flowbytes	out_std_burstbytes	out_std_interburstdelay
unique_hostname_tld+1_extcount	in_std_bursttime	out_75per_flowtime	in_std_flowbytes	out_10per_burstbytes	in_min_interburstdelay
unique_hostname_extcount	in_10per_bursttime	in_75per_flowtime	out_min_flowbytes	in_max_burstbytes	in_std_interburstdelay
unique_ip_3octet_extcount	out_std_bursttime	in_mean_flowtime	out_max_flowbytes	in_std_burstbytes	in_max_interburstdelay
ratio_extport443_extcount	in_mean_bursttime	in_median_flowtime	out_25per_flowbytes	in_10per_burstbytes	in_10per_interburstdelay
unique_ip_extcount	in_max_bursttime	in_max_flowtime	in_max_flowbytes	out_min_burstbytes	out_max_interburstdelay
unique_extport_extcount	in_90per_bursttime	out_max_flowtime	out_median_flowbytes	out_75per_burstbytes	out_90per_interburstdelay
Protocols	out_min_bursttime	in_90per_flowtime	in_median_flowbytes	in_median_burstbytes	in_mean_interburstdelay
in_protos_dict_protocols	in_75per_bursttime	out_90per_flowtime	**Hostname**	**External Port**	**IP**
out_protocols_dict_protocols	out_median_bursttime	out_mean_flowtime	udp_dict_hostname	udp_dict_extport	udp_dict_ip
in_protocols_dict_protocols	**Req Reply Packet Lengths**	**Total Bytes**	all_dict_hostname	tcp_dict_extport	all_dict_ip
out_protos_dict_protocols	all_dict_reqreplylens	out_totalbytes	tcp_dict_hostname	all_dict_extport	tcp_dict_ip
Total Packets	udp_dict_reqreplylens	in_totalbytes			
out_totalpkts	tcp_dict_reqreplylens				
in_totalpkts					

B Traffic to Invocation Detection endpoints

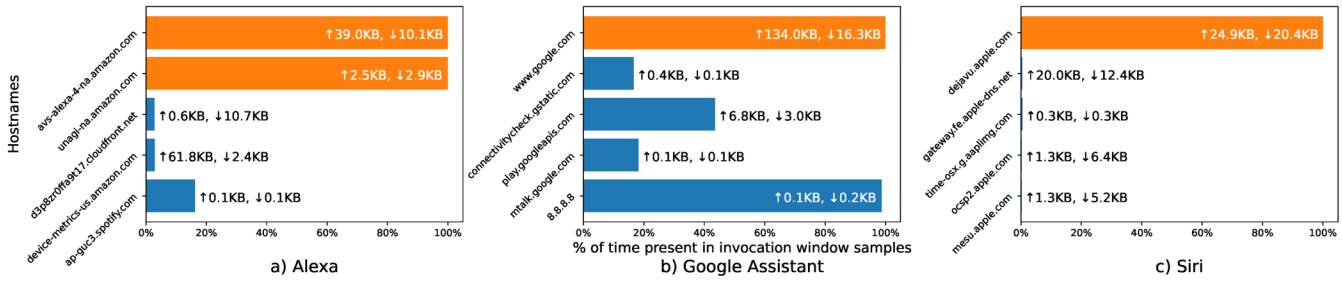

Figure 8: Presence of different domains as a ratio of total samples and the average incoming and outgoing traffic sizes. Domains we selected for detecting invocation are colored orange and have 100% presence rate in all invocation samples.

C Feature Rankings

Table 7: Top 10 features from ranking utilizing multiple methods on the Deep VC Fingerprinting [60] dataset. The rankings show features that existing works do not consider are important such as flow-based features.

ANOVA		Mutual Information		RF Feature Importance	
feature name	**F-statistic**	**feature name**	**score**	**feature name**	**importance**
out_protocols_dict	8336	in_90per_uniquelen	2.867	in_max_flowbytes	0.032
in_protocols_dict	5710	in_max_flowbytes	2.424	in_totalbytes	0.026
in_75per_len	4273	in_totalbytes	2.041	in_std_flowbytes	0.023
in_max_flowtime	3109	in_std_uniquelen	2.025	in_max_interburstdelay	0.021
out_max_flowtime	3072	in_std_flowbytes	1.895	in_90per_uniquelen	0.018
in_mean_len	3019	in_max_interburstdelay	1.829	in_90per_flowbytes	0.018
in_std_flowtime	2881	in_mean_uniquelen	1.785	in_mean_flowbytes	0.018
out_std_flowtime	2853	in_90per_flowbytes	1.696	out_max_flowbytes	0.017
in_mean_burstbytes	2544	in_max_flowtime	1.649	in_max_flowtime	0.015
in_90per_flowtime	2423	in_mean_flowbytes	1.649	in_std_uniquelen	0.015

Table 8: Feature ranking based on mutual information across the three platforms combined into feature groups. Features with '*' are multi-valued features that are encoded at train time. Refer to Table 6 for details about features and groups.

	Across Voice Assistants			Across different Alexa command types		
#	**Alexa**	**Siri**	**Google Assistant**	**Simple Commands**	**Streaming**	**Skills**
1	(18.33) Total Percentage	(43.78) Flow Time	(37.39) Flow Time	(18.77) Total Percentage	(17.19) Flow Length	(19.19) Flow Length
2	(13.73) Flow Length	(31.33) Flow Length	(29.01) Flow Length	(14.73) Flow Length	(16.85) Total Percentage	(17.98) Total Percentage
3	(11.71) Pkt Length	(24.20) Flow Pkt Count	(22.51) Flow Pkt Count	(12.68) Pkt Length	(14.31) Hostname*	(16.84) Pkt Length
4	(10.74) Flow Pkt Count	(14.19) Pkt Length	(13.77) Inter-Pkt Delay	(11.47) Flow Pkt Count	(13.78) Pkt Length	(13.82) Flow Pkt Count
5	(8.84) Inter-Burst Delay	(13.88) Uniq Pkt Length	(12.03) Pkt Length	(9.63) Burst Length	(13.53) Inter-Pkt Delay	(12.85) Inter-Pkt Delay
6	(8.77) Burst Length	(12.59) Inter-Pkt Delay	(9.58) Inter-Burst Delay	(9.23) Inter-Burst Delay	(11.53) Inter-Burst Delay	(12.58) Uniq Pkt Length
7	(8.64) Flow Time	(11.26) Burst Length	(8.56) Burst Length	(8.80) Flow Time	(11.48) Flow Pkt Count	(12.56) Burst Length
8	(7.87) Inter-Pkt Delay	(10.18) Total Percentage	(8.14) Uniq Pkt Length	(8.08) Inter-Pkt Delay	(10.83) Uniq Pkt Length	(12.26) Flow Time
9	(6.32) Burst Pkt Count	(8.58) Inter-Burst Delay	(7.51) Total Percentage	(6.59) Burst Pkt Count	(10.43) Burst Length	(10.62) Inter-Burst Delay
10	(4.66) Protocols*	(8.13) Burst Time	(5.68) Burst Time	(5.43) Uniq Pkt Length	(10.30) IP*	(8.56) Burst Time
11	(4.34) Uniq Pkt Length	(6.79) Protocols*	(3.90) Total Bytes	(5.01) Protocols*	(9.57) Burst Time	(7.25) Protocols*
12	(3.51) Burst Time	(4.52) Burst Pkt Count	(3.67) Burst Pkt Count	(3.90) Burst Time	(8.53) Flow Time	(6.63) Burst Pkt Count
13	(3.43) Total Bytes	(3.45) Total Bytes	(3.11) Protocols*	(3.62) Total Bytes	(8.09) Burst Pkt Count	(4.25) Hostname*
14	(2.26) Total Pkts	(2.55) Total Pkts	(2.95) Total Pkts	(2.48) Hostname*	(6.99) Protocols*	(3.60) Total Bytes
15	(2.04) Hostname*	(2.51) External Counts	(2.37) Hostname*	(2.38) Total Pkts	(3.56) External Counts	(3.25) IP*
16	(1.85) IP*	(2.31) Hostname*	(1.90) IP*	(2.11) IP*	(2.74) Total Bytes	(2.80) Total Pkts
17	(1.16) External Counts	(2.09) IP*	(1.38) Pkt Lengths*	(1.35) External Counts	(2.14) Total Pkts	(1.64) External Counts
18	(0.90) Pkt Lengths*	(1.20) Pkt Lengths*	(0.87) External Counts	(0.79) Pkt Lengths*	(1.42) Pkt Lengths*	(0.72) Pkt Lengths*
19	(0.12) Req Reply Pairs*	(0.22) Req Reply Pairs*	(0.32) Req Reply Pairs*	(0.10) Req Reply Pairs*	(0.53) Req Reply Pairs*	(0.44) Req Reply Pairs*
20	(0.04) External Port*	(0.02) External Port*	(0.03) External Port*	(0.04) External Port*	(0.02) External Port*	(0.00) External Port*