Amazon Echo Security: Machine Learning to Classify Encrypted Traffic

Ryan Blake Jackson
Department of Computer Science
Colorado School of Mines
Golden, Colorado
Email: rbjackso@mines.edu

Tracy Camp
Department of Computer Science
Colorado School of Mines
Golden, Colorado
Email: tcamp@mines.edu

***Abstract*—As smart speakers like the Amazon Echo become more popular, they have given rise to rampant concerns regarding user privacy. This work investigates machine learning techniques to extract ostensibly private information from the TCP traffic moving between an Echo device and Amazons servers, despite the fact that all such traffic is encrypted. Specifically, we investigate a supervised classification problem using six machine learning algorithms and three feature vectors. Our "request type classification" problem seeks to determine what type of user request is being answered by the Echo (again, even though the requests are encrypted). With six classes, we achieve 97% accuracy in this task using random forests.**

I. Introduction

The Amazon Echo and Amazon Echo Dot (shown in Figure 1 and hereafter referred to as "Echo") are smart speakers designed and distributed by Amazon.com. Users interact with the Echo via voice commands that are interpreted and carried out by Amazon's cloud-based intelligent personal assistant service Alexa. Via Alexa, an Echo is capable of voice interaction, music playback, making to-do lists, setting alarms, streaming podcasts, playing audiobooks, acting as a home automation hub, and providing weather, traffic and other real time information [1].

Fig. 1. The Amazon Echo and Echo Dot [2].

As is common for emerging Internet of things (IoT) technologies, the Echo has given rise to privacy concerns. Historically, the level of privacy and security in collecting, processing, and disseminating user information can make or break an IoT innovation, and the consequences for inadequacy in this aspect include non-acceptance of the technology in question, damage to company reputation, and costly law suits [3]. The most prevalent concerns pertain to Amazon's collection and treatment of Echo users' recordings. The Echo's 7 microphones with a consistent Internet connection make it technically possible to continuously stream audio data to Amazon's servers. Thus, articles in popular publications have drawn attention to concerns regarding Amazon's ability to eavesdrop on its users [4] [5] [6]. Amazon has attempted to assuage these concerns by assuring users that, although the Echo is constantly listening for its cue to activate (i.e., its "wake word"), the only data sent to the cloud is the request following the wake word and a few moments of recorded sound immediately prior to the wake word's detection. There is also a physical mute button on the Echo device that disables any form of listening [1].

A similar set of concerns stems from the possibility of malicious third parties obtaining user data, either by gaining access to the recordings saved on Amazon's servers or by intercepting information in transit between the Echo device and the Alexa cloud service [4] [7]. Amazon guards against this potential issue by encrypting all traffic between the Echo and the server [7]. However, as discussed in Section III, the fact that a network flow is encrypted does not necessarily mean that all sensitive information is kept private. This study aims to use machine learning techniques to extract information from encrypted packets captured between the Echo and the Alexa servers.

Specifically, this study investigates a classification problem which we dub "request type classification". The task is to identify what type of request is being answered (e.g. requests for music, weather, information, etc.) given the encrypted packets coming from the Alexa cloud service to the Echo device. These packets are Alexa's response to the user's request. We note that request type classification pertains to user privacy because it could be used to discern user identity information by building large data sets of usage patterns for specific users over time. Even in the absence of ground truthed identity labels, such data would allow for anomaly detection in usage patterns, thus revealing changes in household dynamics.

Section II presents a general overview of concepts necessary to understand this work including machine learning, network communication with the transmission control protocol, and

978-1-5386-5156-8/18/$31.00 ©2018 IEEE

encryption. Section III describes previous studies related to our work. To the best of our knowledge, request type classification with the Echo (or any similar device) has not been previously explored in depth; however, we discuss other popular research domains that bear similarities to this study. Section IV details the methods by which we investigated the request type classification problem and our results. Finally, we offer some concluding remarks and possible directions for future work in Section V.

II. Background

We begin with a brief overview of the supervised classification algorithms that we use in Section IV. We then present a high-level summary of the data paths involved in the Echo's functionality. We also provide a conceptual review of TCP network traffic and encryption as they apply to this work.

A. Machine Learning

Supervised machine learning is the term for all algorithms that reason from externally supplied instances to produce general hypotheses, which then make educated conjectures about previously unseen instances [8]. A machine learning algorithm is said to be a classification algorithm when it seeks to classify an object into a finite set of categories. All machine learning techniques discussed in this work are supervised classification algorithms. In short, supervised classifiers aim to build a concise model of the class label distribution based on features of the classifiable objects. This goal is accomplished via training data for which the true classes are known. The resulting classifier is then used to predict class labels for instances of unknown class, and evaluated by various metrics of efficacy in this task.

The aim of this study is to apply several notable classification algorithms to discern the level of threat that intercepted data pose to Echo user privacy. To that end, we investigate the following six machine learning algorithms: C4.5 decision trees, random forests, linear kernel support vector machines, radial basis function kernel support vector machines, multi-layer perceptron neural networks, and k-nearest neighbors classification. In the following discussion, we briefly describe each of these algorithms and justify their inclusion in our work.

We begin with the decision tree, given its success in several related works. The C4.5 decision tree algorithm generates decision trees using the concept of information entropy in the set of training data. At each node of the tree, C4.5 selects the feature of the data that best splits the training samples into subsets, such that the subsets predominantly represent one class. In other words, the tree is built from the top down, and, at each juncture, the tree splits on the attribute that gives the highest normalized information gain. The algorithm then recurs on the resultant subsets until some stopping criterion is reached [8]. Typical stopping criteria include a maximum depth for the tree or a minimum level of purity at each leaf. New data are classified by following the appropriate path down the tree from root to leaf.

Our second machine learning model, the random forest, is an ensemble method. Machine learning ensembles combine predictions from multiple base classifiers in an attempt to achieve better predictive performance than any individual classifier alone. The intuition behind these ensembles is that each constituent base classifier is trained slightly differently in the same task. We expect occasional errors from individual classifiers, but we hope that their diversity allows the majority to be correct despite a few misclassifications for any given input. This idea is conceptually similar to humans making important decisions by committee rather than leaving them up to one individual, where all committee members share a general goal. Random forests are ensembles of decision trees. Each tree in the forest is trained on a random subset of the training examples drawn with replacement. When training each tree, each node considers only a proper subset of the features. In other words, instead of simply splitting on the feature that provides the highest normalized information gain, the algorithm randomly selects a few features and then splits on the best of those. Random forests are typically more robust to overfitting than individual classifiers. They also generally give error rates that compare favorably to those of other ensemble methods like Adaboost, while being more resilient to noisy data [9]. We include random forests in this study in hopes of improving on the performance of individual decision trees.

Our third machine learning algorithm, the support vector machine, works by creating a linear decision surface in the input space that separates training examples of different classes by as large a margin as possible. A support vector machine can perform non-linear classification by using a kernel function to implicitly map input points to a higher dimensional space before learning the linear decision boundary in this higher dimensional space. In essence, this technique results in a non-linear decision boundary in the input space. In this study, we use both linear kernel support vector machines, which do not map the inputs to a higher dimensional space, and support vector machines with radial basis function kernels. Although single support vector machines can only perform binary classification, we perform multi-class classification by training several support vector machines via the one-vs-one method [10]. Using this method, we train $\frac{n(n-1)}{2}$ hyperplanes to classify n possible classes. Studies have found the one-vs-one method to be more effective in some applications than other available methods (like one-vs-all) for multi-class classification with support vector machines [11]. We include support vector machines in this work for their high generalization ability demonstrated in other domains [12].

Recently, artificial neural networks have become extremely fashionable within the machine learning community and popular culture [13]. Therefore, we include a form of artificial neural network in this study. We use multi-layer perceptrons instead of developing a novel neural network architecture custom tailored to our specific data, as this work is primarily exploratory and concerned with feasibility. Our multi-layer perceptron networks are fully connected feed-forward net-

works trained via backpropagation. Within those constraints, we optimize across several architectures and activation functions as discussed in Section IV.

The final machine learning algorithm that we investigate is the k-nearest neighbors (knn) classifier. In this method, any given input is assigned the majority class of the k training points nearest it based on some measure of distance (e.g., Euclidean or Manhattan). It is known that a simple majority vote between the nearest k points can be flawed when there are large class imbalances within the data; we avoid this issue by collecting a perfectly balanced data set. We include knn in this work for its simplicity, long-standing ubiquity, and the fact that it has been proposed as a basic benchmark against which to evaluate other classification methods [14].

B. Amazon Echo Data Paths

When a user makes a verbal request to the Echo, little of the requisite computation takes place on the Echo's hardware. The Echo device simply listens for the wake word "Alexa" and then streams the audio of the user request to Amazon's servers running the Alexa cloud service. Next, the Alexa service interprets the user's recorded speech and decides how to respond. The response is sent to the Echo device and delivered to the user via the speaker. At the same time, a visual response is delivered from the Alexa service to the user's smartphone, tablet, or computer if it is running the Alexa companion application. For example, in response to the question "Alexa, who is Thomas Jefferson", the Echo might respond "Thomas Jefferson was the third president of the United States of America" audibly through the speaker while simultaneously delivering a link to Jefferson's Wikipedia page within the companion application. This described process is depicted in Figure 2.

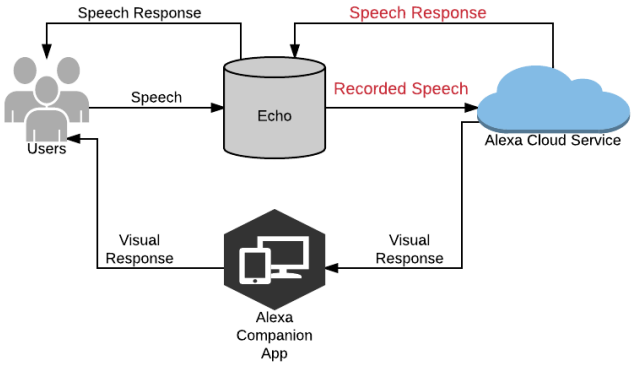

Fig. 2. The flow of data when using the Echo. This study focuses on the paths labeled in red.

This work is specifically interested in the data moving between the Echo device and the servers running the Alexa cloud service. We seek to explore what ostensibly private information an eavesdropping third party could infer by intercepting these data. To that end, our data collection environment essentially amounts to placing a computer between the Echo and the servers that records all traffic passing through on its way to its intended destination (see Figure 3).

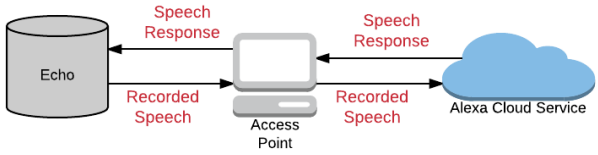

Fig. 3. Our data collection environment places a gateway machine between the Echo and the Alexa cloud service.

C. TCP Network Traffic

The Transmission Control Protocol (TCP) is a set of rules that enables a connection between two computers and governs the delivery of data over the Internet Protocol (IP). All information transferred via TCP is separated into packets, and each packet consists of a header and a payload. The header is information for connection management, while the payload is the actual data that needs to be transferred. When a device needs to send data via TCP, it divides the data into chunks, adds a TCP header to each chunk, encapsulates the chunk and header into an IP datagram, and sends the resultant TCP/IP packet on its way. A specific TCP data exchange begins with a "handshake" between the two participants, e.g., the Echo client and the Alexa server, and ends with a special flag called the "FIN" flag that designates the accompanying data as the last information from the sender. The Echo and the Alexa service communicate with each other via TCP.

D. Encryption

Encryption is the practice of encoding data in such a way that only authorized parties can understand it. All communications between the Echo and Amazon's servers are encrypted. In other words, the data payloads of the transmitted TCP packets are purposefully obfuscated such that an eavesdropping third party cannot glean any meaningful information by inspecting them. Thus, deep packet inspection (DPI), i.e., the practice of inspecting the data payloads of network packets, is not useful here. We therefore use shallow packet inspection (SPI), i.e., the practice of examining packet headers and statistical information regarding traffic patterns, to generate feature data for our machine learning classifiers. Without encryption, it would be trivial to discern all information traveling between the Echo and Alexa. This work aims to discover what information we can extract despite the encryption.

III. Related Work

To the best of our knowledge, this study is the first to use machine learning to glean information from Echo traffic. One relatively popular research area that is related to our work is termed the *identification problem*. This problem consists of classifying encrypted web traffic. The interest in this problem stems largely from its utility for surveillance, network management, and security applications. For example, a campus network might want to prevent online games from

hogging available bandwidth. The first step in doing so is to separate game traffic from other traffic, even if the traffic is encrypted. In this section, we discuss several papers that have addressed versions of this problem, their relationships to our research, and characteristics that differentiate our work from this preexisting body of knowledge. We also briefly describe existing work that concerns the Echo device, though this previous research does not consider the associated network traffic.

Li and Moore [15] investigated real-time encrypted traffic classification for network monitoring and intrusion detection. This study used individual TCP flows as the basic object of classification, where flows were bi-directional sessions between two participants uniquely identified by the host-IP address, client-IP address, host port, client port, and timestamp of the first packet. Li and Moore classified these flows into 10 general categories (such as mail and games) with 99.8% accuracy using a C4.5 decision tree. The data analyzed were two consecutive days of real TCP network traffic from a research facility of roughly 1000 employees. The input features for the decision tree were derived from packet headers.

Li and Moore's high accuracy is partially attributable to favorable conditions not present in our Echo study. First, although [15] states that the approach does not *rely* on port numbers for classification, the client and server port numbers were part of the input feature vectors. In fact, the server port number was the most discriminative individual feature used. We cannot consider port numbers to differentiate between our classes of Echo communications, because the client and server ports do not change with the class. Additionally, the 10 classes used by Li and Moore separate a wide range of traffic into 10 general bins. We posit that the differences between these general classes are more pronounced than the differences between different requests to Alexa. Finally, though it is convenient to consider TCP flows as defined by Li and Moore as fundamental objects, there is not one TCP flow per Echo request to Alexa.

The authors of [16] use a hybrid of the k-means clustering algorithm and the k-nearest neighbors geometric classifier to categorize 12 million flows from two Internet edge routers. The classes in this work encompass general purpose network functionality. Again, we expect that these classes are generally more disparate than ours. The authors' hybrid algorithm achieves classification accuracy between 94.0% and 99.9%, depending on the traffic class. This performance is better than the 83% average accuracy achieved by k-means clustering alone. Furthermore, although the hybrid algorithm is less accurate on average than the k-nearest neighbors algorithm, which achieved an overall accuracy of 99.1%, the hybrid algorithm is faster and can classify traffic in real-time. Also, this technique uses only the packet headers to gather features, so it is invariant to encryption. To emphasize this point, the authors show that their algorithm is as accurate at classifying unencrypted Bit Torrent peer-to-peer traffic as it is at classifying encrypted Bit Torrent peer-to-peer traffic. The 17 features used in [16] are associated with (1) the amount and rate of data transfer in each direction and (2) the protocol used. We note that the authors of [16] do not use the port numbers that Li and Moore found valuable.

Much of the literature regarding the identification problem builds upon the work of Moore, Zuev, and Crogan [17]. This paper describes a comprehensive set of 248 features that can be collected from TCP flows (with or without encryption) and then used for classification. We cannot use their features in our study because of the definition of a flow. TCP flows have clearly defined starts and stops; however, as mentioned previously, each Alexa request does not necessarily constitute a single flow. Nonetheless, we draw inspiration from [17] when selecting our own features.

The author of [18] used a subset of the features outlined in [17] to compare several different machine learning techniques in the task of classifying TCP flows by web application for the 16 most commonly used web services at the Air Force Institute of Technology. In this work, the J48 decision tree (which is an open source Java implementation of the C4.5 decision tree algorithm) and AdaBoost+J48 were the two most successful algorithms; both of these algorithms had a 98% classification accuracy. The other machine learning algorithms evaluated in [18] follow: support vector machine, Naive Bayes classifier, and Naive Bayes tree. The success of the decision tree in both [15] and [18] motivates our investigation of this method, despite the differences between our data and the data used in these studies.

A narrower version of the identification problem is investigated in [19]. This study investigates AdaBoost, support vector machines, Naive Bayes, C4.5 decision trees, and RIPPER (Repeated Incremental Pruning to Produce Error Reduction, a depth-first rule induction based algorithm) in the tasks of identifying encrypted Skype or SSH traffic in large traces. The authors of [19] found that the C4.5 decision tree was the best algorithm. They achieved detection rates greater than 80%, with false positive rates less than 10%, for detecting both SSH and Skype in four sets of data from different networks. That is, the training data came from a different network than the testing data.

A variant of the identification problem is presented in [20]. This study attempts to distinguish between roughly 100,000 different web pages based solely on information that is available to a third party intercepting packets from users that accessed the websites via an encrypted channel. The available information is essentially HTTP object count and size. The authors discussed that a relatively straightforward algorithm could identify many of the websites with low false positive rates, barring significant padding to obfuscate the actual website content.

An open source tool called "Pacumen" aims to solve the identification problem through machine learning in a way that requires less training data and is easier for network administrators to use than the previously discussed academic approaches [21]. Rather than classifying TCP flows as done by the previously discussed studies, the authors in this work classify temporal windows of traffic. Each time window could be

the start, end, or middle of one or more flows. We can extract similar features in our work with the Echo traffic. The authors of [21] found top performance with decision trees, though their custom version of a decision tree outperformed the more common C4.5 tree on some of their data sets. The classes in this study are similar to those in [18], though these authors introduce the browser as a differentiator between classes. In addition to introducing and evaluating their Pacumen tool, the authors provide an example of a typical feature set for machine learning in this domain. We draw inspiration from this typical feature set in selecting our features.

The authors of [22] examined the Echo as a potential source of forensically relevant digital information. They concluded that the Echo serves primarily as a conduit for interfacing with other services and were unable to obtain meaningful information from the Echo device itself. Furthermore, the only information discovered on the tablet running the Alexa companion application was timestamps for the commands given to Alexa and the responses to user commands. The authors of [22] did not investigate the network traffic between the Echo and the Alexa service, which is the focus of our work.

IV. Request Type Classification

This section describes our efforts to identify the type of request being answered by Alexa using the encrypted packets sent from the Alexa cloud service to the Echo device in response to the user's request. We begin our discussion by describing our data set, and then cover our machine learning processes and their results.

A. Our Data

We collected the data used in this problem via the data capture environment described in Section II-B. An extensive search did not discover any similar data sets available for use in this research. We collected the following six classes of data for this classification problem: information, quotes, weather, directions, music, and unintelligible. We collected 130 examples of each class for a total of 780 packet capture files. The information class is comprised of questions of the form "Alexa, who is Thomas Jefferson?" We used 130 different names to add diversity to the data set and avoid any effects of possible server-side caching. Each user request in the quotes class is "Alexa, give me a quote." The 130 responses from the Alexa servers are all unique. The weather class questions have the from "Alexa, what is the weather in Paris?" with 130 different places cited. The music class is comprised of 130 different song samples. User requests in the music class have the form "Alexa, play me a sample of Hey Jude by The Beatles" with 130 different pairs of song title and artist. The user requests for directions have the form "Alexa, get me directions to the Home Depot." When collecting data in the directions class, we noticed a perceptible difference in response time based on our proximity to the location in question. For example, when collecting data in Golden, Colorado, the request "Alexa, get me directions to the Colorado School of Mines" would receive a reply markedly more quickly than "Alexa, get me directions to Anchorage, Alaska". To avoid this variation in response time and collect data in keeping with what we believe to be the typical use case for the direction functionality, we relegated our direction requests to places within roughly a 30 minute drive from our data collection environment. While we were not aware of 130 different such places, over half of our requests contain unique locations and we varied their order to avoid server-side caching. We also note that traffic information changes over time, so two direction responses from Alexa to the same location are likely different. Thus, we believe that server-side caching was not taking place. The user requests in the unintelligible class are 130 unique, linguistically invalid sentences including nonsense words. Alexa's responses to the unintelligible requests, though varied, are along the lines of "Sorry, I didn't quite catch that."

We use 80% of each class (104 examples) for training our machine learning models and 20% of each class (26 examples) for testing the models. Our train/test splits are computed randomly, so the individual data points that end up in a given data set (training and testing) vary across different train/test splits. However, we always use stratified train/test splits which means that the proportion of each class in the training data is always the same as the proportion of that class in the testing data. Since we collected the same number of examples of each class, the classes are always represented equally in the training and testing data.

B. Feature Extraction

Inspecting our data revealed that each request response pair between the user and the Echo does not necessarily correspond to a TCP flow as defined by [15], [16], [17], and [18]. Though the correspondence is TCP traffic, it is not punctuated by the starts and stops that create a flow. We also did not observe a strictly one-to-one correspondence between requests and flows when collecting traffic bidirectionally. Thus, the tools discussed in [17] are not useful. We therefore turn to other feature vectors. We extract single feature vectors from entire packet capture files, where each file corresponds to one request/response. In this study, we investigate three different sets of features: "tcptrace", "histogram", and "combined". Sections IV-B1, IV-B2, and IV-B3 describe these three types of feature vectors.

Some machine learning algorithms, e.g., support vector machines, assume input data features with zero mean and unit variance. To accommodate such algorithms on all three of our feature sets, we fit a transform on the training data that standardizes each feature to zero mean and unit variance, and then apply that transform to both the training and testing data sets.

1) tcptrace Feature Vector: Our first feature vector is called "tcptrace" because the feature extraction code leverages a tool called "tcptrace" to create statistical features from packet capture files [23]. We manipulate these features into a numerical vector form that is amenable to typical machine learning algorithms. We believe that these tcptrace features

are similar to the "typical" feature set described in [21], with additional information. Of the 35 tcptrace features, 13 were useless in our study because they were constant for all collected data. The 13 discarded values follow: sack pkts sent, dsack pkts sent, max sack blks/ack, zwnd probe pkts, zwnd probe bytes, SYN/FIN pkts sent, urgent data pkts, urgent data bytes, zero win adv, stream length, missed data, truncated data, and truncated packets. A full list of tcptrace features with descriptions is available at [23].

2) Histogram Feature Vector: We call the second feature vector that we create from each packet capture file a "histogram" feature vector. The first half of the features in this vector come from a normalized histogram of packet sizes. We create this histogram based on a parameter-specified number of bins that evenly divide the interval between the size of the smallest packet in our data set and the size of the largest packet in our data set. Then, for each packet capture file, the histogram is created by calculating the number of packets that fall into each bin. Finally, the histogram is normalized to remove any information regarding the total number of packets in the packet capture file. Each feature in our vector, then, corresponds to the value in one bin of the histogram. The second half of the features in the vector come from another normalized histogram calculated in the same manner, but with packet interarrival times instead of packet sizes. Both histograms use the same number of bins. For example, given 15 bins, the resultant feature vector would contain 30 features, 15 for the packet size histogram and 15 for the packet interarrival time histogram.

The rationale behind this histogram feature vector is twofold. First, we wanted features that captured packet size information, especially relative frequencies of different packet sizes, since this type of information is not present in the tcptrace features. Second, we wanted a feature vector that contained no information regarding the total number of packets or the total amount of information transmitted in a packet capture file. Having such a feature vector allows us to verify whether classification is possible without knowing the number of packets or number of bytes in a request/response pair.

We consider the histogram feature vector both with and without ACK packets. Discarding the ACK packets is motivated by the idea that we are interested only in the sizes of the data payloads, and including ACK packets would skew the packet size histogram towards smaller sizes, which could obfuscate subtler information in the bins containing larger packet sizes. However, we also consider the feature vector with ACK packets just in case they provide some valuable information.

We also seek to optimize the number of bins for this feature vector. Table I shows how a random forest classifier performs with various bin sizes. We see that, for every number of bins under 200, more bins yields better results. However, we see drastically diminishing returns after 15 bins, especially without ACK packets. To keep computational costs to a reasonable level, we used 15 bins for our histogram feature vectors in the remainder of our work. This decision reduced the experimental runtimes on our hardware by durations up to hours, which allowed us to cover more exploratory ground than would otherwise have been feasible. We provide the histogram features without ACK packets for the remainder of this thesis because discarding ACK packets gave better performance on our chosen number of bins (15).

TABLE I
MEAN ACCURACY FOR A 400-TREE RANDOM FOREST USING THE HISTOGRAM FEATURE VECTORS WITH VARIOUS BIN SIZES BOTH WITH AND WITHOUT ACK PACKETS. AVERAGED OVER 100 RANDOM STRATIFIED TRAIN/TEST SPLITS.

Bins	Mean Accuracy with ACK Packets	Mean Accuracy without ACK Packets
5	85.16	82.01
10	90.59	88.00
15	92.34	93.21
25	92.40	93.25
50	92.97	93.30
100	93.92	93.96
200	92.72	93.87

3) Combined Feature Vector: Our third and final feature vector, which we dub "combined", is the tcptrace features combined with the histogram features. We included the combined vector in this study in hopes that each constituent feature vector would contribute some unique information allowing for better classification performance than either of the two constituent feature vectors alone.

C. Results

We noticed significant variation in our results that depended on how the data were divided into training and testing sets. For that reason, and because several of our machine learning algorithms include some stochasticity, presenting results based on a single trial of each algorithm with a single set of training data and a single set of testing data could be misleading. Thus, instead, we present the aggregated results of 100 trials with random stratified train/test splits. In each trial, we used stratified 3-fold cross validation to tune salient model hyper-parameters via a grid search over possible hyper-parameter values. We do not tune hyper-parameters on testing data, as doing so would overestimate model efficacy. We note, however, that the chosen hyper-parameters are not always the same in each trial. We believe our chosen method provides a more thorough and accurate evaluation of our models than would be possible with the more common single-trial approach.

We leveraged the algorithmic implementations of Scikit-learn [24] for the six machine learning models evaluated. Table II presents our accuracy results for each of our three feature vectors with the different machine learning algorithms considered. We treat accuracy as our metric of interest for several reasons. First, our data set is perfectly balanced, i.e., no one class appears more often than any other (we collected 130 examples per class). Therefore, metrics that account for class imbalances (e.g., F1-score) are unnecessary. We did, however, evaluate F1-score and found the result extremely close to, or identical to, accuracy. Second, no one type of prediction error

TABLE II
ACCURACY RESULTS FOR 100 TRIALS WITH DIFFERENT TRAIN/TEST SPLITS FOR OUR SIX MACHINE LEARNING ALGORITHMS ON OUR THREE DIFFERENT FEATURE VECTORS. HIGHEST ACCURACY RESULTS ARE IN BOLD.

Feature Vector	Model	Mean Accuracy	Standard Deviation	Median Accuracy
tcptrace	Decision Tree	95.69	1.58	95.51
	Random Forest	**96.87**	1.34	**96.79**
	SVM (linear)	94.01	1.68	94.23
	SVM (RBF)	94.03	1.69	94.23
	Neural Net (MLP)	94.24	1.57	94.23
	KNN	94.08	1.84	94.23
Histogram	Decision Tree	87.05	2.29	87.18
	Random Forest	**93.23**	1.82	**93.27**
	SVM (linear)	86.89	2.31	87.18
	SVM (RBF)	87.91	2.31	87.82
	Neural Net (MLP)	88.15	2.36	88.46
	KNN	88.87	2.09	89.10
Combined	Decision Tree	95.42	1.68	95.51
	Random Forest	**97.01**	1.14	**97.44**
	SVM (linear)	96.12	1.47	96.15
	SVM (RBF)	95.56	1.23	95.51
	Neural Net (MLP)	95.07	1.44	95.19
	KNN	95.72	1.71	95.51

is more costly than any other. Thus, we treat all errors as equally costly and have no need for weighting. For the sake of thoroughness, we present confusion matrices for our highest performing algorithm for each feature vector in Figures 4, 5, and 6.

Table II shows that the random forest is the best performing classifier for our three feature vectors. We note that, for each feature vector, the difference between the random forest mean accuracy and the mean accuracy of the second best algorithm is statistically significant with $p < 0.0001$. Furthermore, the combined feature set provides the best result, while the histogram feature vector performs the worst. The difference in the accuracy means for the random forest using tcptrace features versus histogram features is statistically significant with $p < 0.0001$, as is the difference in accuracy means for random forest using the combined feature vector versus the histogram features. However, the difference in accuracy means for the random forest using the combined feature vector versus the tcptrace features gives $p = 0.427$, and the 95% confidence interval for the difference is -0.4869% to 0.2069%. These statistics indicate that using a random forest with the tcptrace features alone is essentially equivalent to using a random forest with the combined features.

The confusion matrices in Figures 4, 5, and 6 provide details on the best performing classification algorithm (random forest) for each feature vector. The most often misclassified class is information. Regardless of the feature vector used, the information class is often mistaken for weather. Furthermore, when misclassifying, the classifier tends to erroneously predict information for examples belonging to all of the other classes except music. The music class is always classified correctly, and samples from other classes are rarely mistaken for music. Intuitively, this result makes sense because the musical output is fundamentally different in nature from the speech output of the other five classes. In Section V, we discuss ideas for adding other classes that are similar to the music class.

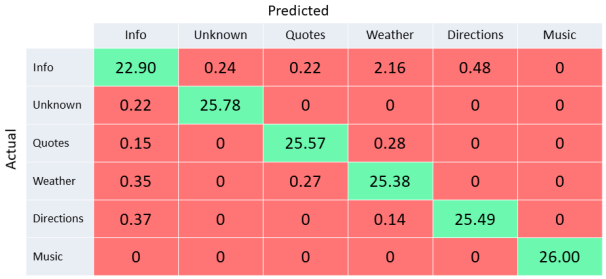

Fig. 4. Confusion matrix averaged over 100 different train/test splits for random forest using tcptrace feature vectors.

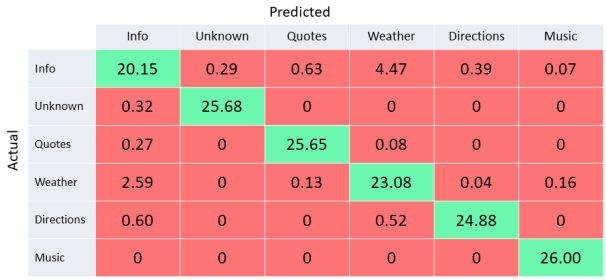

Fig. 5. Confusion matrix averaged over 100 different train/test splits for random forest using histogram feature vectors.

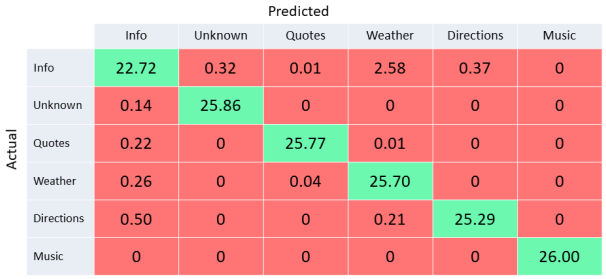

Fig. 6. Confusion matrix averaged over 100 different train/test splits for random forest using the combined feature vectors.

D. Machine Learning Algorithm Hyper-parameters

For the decision tree, the only hyper-parameter we tune is whether to use Gini impurity or information gain as the splitting criterion. We require full purity in all leaves as the stopping criterion. With the separate tcptrace and histogram feature vectors, we found that Gini impurity and information gain are essentially equivalent. The combined feature vectors showed a higher preference for information gain, with 69 trials selecting information gain and only 31 trials selecting Gini impurity.

For the random forest classifier, we always use 400 trees. We believe that 400 trees is sufficient because our results did not differ in any statistically significant way with 500 trees. We use

full leaf purity as the stopping criterion in building the decision trees that comprise the random forest. We again use cross validation to choose between Gini impurity and entropy as the splitting criterion for each trial. The separate tcptrace and histogram feature vectors showed a slight preference for Gini impurity, while the combined feature vector showed a slight preference for entropy. In regards to the number of features to consider at each node, we test both the square root of the total number of features and the log base 2 of the total number of features. We found the square root of the total number of features is selected more often for all three of our feature vectors.

For the support vector machine with linear kernel function, the only parameter that we tune is the penalty parameter of the error term, often dubbed "C". The parameter C governs the trade-off between finding a hyperplane that correctly classifies as many training points as possible, and finding a hyperplane that generally has a large margin between separate classes, even if that means allowing some misclassifications in training. High values of C prioritize correctly classifying all training points, but, if C is too large, the learned hyperplane may overfit the training data by being too sensitive to outliers during training. Low values of C prioritize a large-margin hyperplane, even if some training points are misclassified. However, if C is too small, the learned hyperplane will needlessly misclassify many points (even with linearly separable data) because misclassifications are not sufficiently penalized. We test the values 0.1, 0.5, 1.0, 5.0, and 10.0 for C. We found the best cross validation performance on tcptrace features with higher values of C, with $C = 5$ or $C = 10$ being selected in 30 and 28 trials respectively. There was a preference for $C = 1$ and $C = 0.5$ on the histogram features, with those values being chosen in 34 and 37 trials respectively. Lower values of C proved to be better for the combined feature set, with $C = 0.1$ and $C = 0.5$ chosen in 53 and 38 trials respectively. Perhaps greater outlier presence in the combined features made the support vector machine more prone to overfitting with high values of C.

For the support vector machine with a radial basis kernel function, we tune the kernel coefficient gamma in addition to the penalty parameter C. For all feature vectors, the most often optimal value for gamma is 1 over the number of features. There is also a pronounced tendency toward higher values of C (5.0 and 10.0) for all feature vectors.

It was infeasible for us to explore the entire hyper-parameter space for neural networks in this study due to the numerous architectural choices and various other quantitative hyper-parameters. We chose to restrict our tuning to three multi-layer perceptron architectures: 1 hidden layer with 100 nodes, 1 hidden layer with 300 nodes, and 2 hidden layers with 100 nodes each. We did evaluate both the logistic sigmoid and rectified linear unit activation functions. The rest of the hyper-parameters are set to the Scikit-learn defaults [24]. We found that the different hidden layer architectures do not seem to matter. The rectified linear unit function is chosen as the superior activation function in the vast majority of trials for all feature vectors.

For k-nearest neighbors classification, the hyper-parameters that we tune are k (the number of neighbors to consider), whether to weight the neighbors based on their distance to the point being classified, and whether to use the Manhattan or Euclidean distance for this weighting. The tcptrace and histogram feature vectors both result in a preference for weighting points by the Manhattan distance, while the combined feature vector showed a preference for considering the neighbors unweighted. For the tcptrace features, $k = 3$ and $k = 5$ are the preferred values, chosen in 37 and 24 trials respectively. Cross validation tends to select $k = 5$ for the other two feature vectors (in 46 trials for the histogram features and 50 trials for the combined features). The second most common value of k for both the histogram features and the combined features was $k = 3$, selected in 20 cross validation trials for the histogram features and 35 trials for the combined features.

E. Relative Importance of Features

Once trained, random forest classifiers can give information about the relative discriminative power of each input feature by calculating the mean decrease impurity for each feature. The impurity decrease at any given node for the feature on which it splits is the impurity of the data that arrived at the node minus the summed impurities of all child nodes. The mean decrease impurity for any given feature is the average impurity decrease over all nodes within the forest that split on that feature, weighted by the amount of data that reaches each relevant node. The greater a feature's mean decrease impurity, the more discriminative that feature is. Due to the various stochastic elements of our random forest training process, the mean decrease impurities for each feature are not consistent across different trials or different train/test splits. Despite this inconsistency, we can observe trends that exist regardless of the stochasticity.

The most important tcptrace features are those pertaining to the window advertisement. Using only the window advertisement features, the random forest algorithm achieves a mean accuracy of 91.92%. In TCP communication, the window advertisement is basically one party telling the other how much data it is willing to receive per unit time. Since our data for this request type classification problem consists of packets coming from Amazon's servers to the Echo, the window advertisements specify how much data the server is willing to receive from the Echo. One reason why the music class is so easy to identify is that its window sizes are typically about ten times smaller than the window sizes in the other five classes. Based on changes in the source IP address, we hypothesize that the servers responsible for streaming music to the Echo are different from the servers that receive, interpret, and respond to Echo voice recordings. We consistently observe a change in server IP address when the response shifts from speech (e.g., "Here is a sample of No One by Alicia Keys") to music. The music streaming servers are likely not expected or equipped to receive much data from the Echo, so they advertise a small window. It is not clear why the window advertisements

differentiate the five non-music classes so well. Since we collected our data one class at a time, it is possible that each class has different window advertisement characteristics because the server was under a different load when each class was collected. We do not believe this hypothesis to be the case, however, both because of significant variability in average window advertisement within classes (1013 bytes to 20576 bytes within the information class), and because of similar average window advertisements across classes (many window advertisements of roughly 4500 bytes appear in all classes except music and directions).

A metric that quantifies the amount of data observed is also typically in the top five tcptrace features, including unique bytes sent, actual data packets, and actual data bytes. If we use only features quantifying the amount of data (i.e., total packets, unique bytes sent, actual data packets, and actual data bytes), the random forest algorithm achieves a mean accuracy of 86.92%. In other words, while features quantifying the amount of data are useful, they are not sufficient to achieve peak performance. Though no feature had a mean decrease impurity of zero, the features pertaining to data and packet retransmission are the least important.

For the histogram features, the packet size features are typically more important than the packet interarrival time features as indicated by greater impurity decreases. With only the interarrival time histogram, the mean random forest accuracy is 43.85%. Manuel inspection of the interarrival time histogram data shows that the music class generally has longer packet interarrival times than the other five classes. Although we were unable to differentiate between the five non-music classes by manually inspecting the raw interarrival time data, the random forest was able to distinguish them with higher accuracy than random guessing; that is, if the algorithm had correctly classified only music and then guessed between the five non-music possibilities for the other five classes, we would expect only 33.33% accuracy. On the other hand, when the random forest algorithm only uses the packet size histogram, the mean random forest accuracy is 92.53% with a standard deviation of 2.01%. The difference between this result and the 93.23% result in II (which includes the whole histogram feature vector) is statistically significant with a 95% confidence interval of -1.23% to -0.17%. However, the closeness of these two results indicates that the histogram feature vector performs almost as well without considering the interarrival times. We do not see any discernible pattern to the impurity decrease for the individual features (bins) within each histogram.

As one might expect from the results discussed previously, the mean decrease impurity calculations for the combined feature vectors indicate that the most important tcptrace features are the most important features overall. Similar to the separate tcptrace and histogram feature sets, the least important features in the combined feature vector are the tcptrace features related to data retransmission and the packet interarrival time histogram features.

V. Conclusions and Future Work

The goal of our research was to extract ostensibly private information from the encrypted TCP traffic flowing between the Amazon Echo and the Alexa cloud service. Because the transmissions are encrypted, we must rely on shallow packet inspection techniques. Specifically, our request type classification problem consists of classifying the encrypted information coming from the Alexa cloud service to the Echo device by the type of user request that is being answered. This task is conceptually similar to the well-studied "identification problem" on general purpose network traffic. We designed a data capture environment and collected six classes of data. Our classes are information, quotes, weather, directions, music, and unintelligible. We extracted three different feature vectors from the captured packets in this research. While all three feature vectors proved to be valuable, we found the combined feature vector yielded slightly better request type classification performance than the tcptrace feature vector. We tested six machine learning algorithms after tuning their hyper-parameters via 3-fold cross validation, and found that the random forest algorithm performed the best in this application, achieving 97.01% accuracy averaged across 100 different train/test splits using the combined feature vectors. We use our random forests to report on the relative importance of our different features based on their mean impurity decreases. The most important features concern the window advertisements. Overall, we believe that this result on the Amazon Echo constitutes a credible threat to user privacy.

A. Improving Generalizability Across Networks and Users

In a real eavesdropping scenario, the eavesdropper would likely not have, nor be able to obtain, labeled training data of the people using the Echo on the target's home network. Instead, the eavesdropper would need to collect a labeled training data set on their own network(s), train a machine learning model with it, and then use this model to classify data from the target's network. Therefore, to present a real threat, our techniques must perform well when applied to a previously unseen network. Our techniques must also generalize between Echo users; the way that one person asks for information might be sufficiently different from the way that another person asks for the same information to meaningfully impact Alexa's response, at least in timing if not in content. Our results from informal, preliminary experiments on this topic necessitate further research, but do not negate the threat that exists to Echo user privacy. Specifically, we would like to train machine learning models on data from many different networks and users, and then evaluate the models on new networks and users. We believe our techniques have the potential to generalize well across networks and users.

B. Building Data Sets of Usage Patterns

Our success with request type classification in Section IV opens the door for several avenues of future research. We would like to collect large data sets of people using the Echo in their homes over several weeks or months. We would

then explore feature extraction and machine learning on the collected usage patterns. We could investigate how well the usage patterns identify specific households. A trivial example would be that household A asks for weather at 8:00 every morning, while household B routinely asks for music at that time. Applying clustering techniques to this data set could reveal interesting behavioral groups of households. Applying anomaly detection techniques to this data set could allow us to detect changes in household dynamics (e.g., divorce or a child moving away to college), which would also constitute a privacy violation. We would also like to investigate how personal usage patterns could identify individuals within a household.

The first step in investigating the topics in the previous paragraph would be to expand the set of request type classes to be as comprehensive as possible. Our six classes cover a large portion of the Echo's functionality, but they are not exhaustive. We could add, for example, classes for shopping and listening to podcasts. We believe that the podcast data might be similar to the music data, which would test our classification process well. We could also include a catchall "other" class for any overlooked request types that do not belong in another class. With these added classes, we would collect data that represents the typical Echo user.

C. Exploring Similar Devices

The consumer options for virtual assistant smart speakers are rapidly increasing as many companies work to earn a share of the emerging smart speaker market. The Google Home is a product similar to the Echo that is already on the market [25]. Samsung and Apple are poised to release their own smart speakers in 2018 [26] [27]. With so many similar products competing for essentially the same customers, differences in security could play a significant role in consumer choices. We believe that a comprehensive study comparing how vulnerable these products are, both to the techniques developed in this study and to other potential threats to user privacy, could be valuable to conscientious consumers.

REFERENCES

[1] "Amazon Echo product page," https://www.amazon.com/dp/B00X4WHP5E, access date: April 18th, 2018.

[2] M. Hughes, "The Amazon Echo and Echo Dot are coming to the UK and Germany," https://thenextweb.com/gaming/2016/09/14/the-amazon-echo-and-echo-dot-are-coming-to-the-uk-and-germany/#.tnw_dz0yAwD8, 2016, access date: April 18th, 2018.

[3] J. H. Ziegeldorf, O. G. Morchon, and K. Wehrle, "Privacy in the Internet of things: Threats and challenges," *Security and Communication Networks*, vol. 7, no. 12, pp. 2728–2742, 2014.

[4] S. Baral, "Amazon Echo privacy: Is Alexa listening to everything you say?" https://mic.com/articles/162865/amazon-echo-privacy-is-alexa-listening-to-everything-you-say#.1mt6eZ2WS, access date: April 18th, 2018.

[5] C. Davies, "How private is Amazon Echo?" https://www.slashgear.com/how-private-is-amazon-echo-07354486, 2014, access date: April 18th, 2018.

[6] S. Machkovech, "Amazon announces Echo, a $199 voice-driven home assistant," https://arstechnica.com/gadgets/2014/11/amazon-announces-echo-a-199-voice-driven-home-assistant, 2014, access date: April 18th, 2018.

[7] T. Moynihan, "Alexa and Google Home record what you say, but what happens to that data?" https://www.wired.com/2016/12/alexa-and-google-record-your-voice/, 2016, access date: April 18th, 2018.

[8] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," *Informatica*, vol. 31, pp. 249–268, 2007.

[9] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[10] S. Knerr, L. Personnaz, and G. Dreyfus, "Single-layer learning revisited: a stepwise procedure for building and training a neural network," *Neurocomputing*, vol. 68, pp. 41–50, 1990.

[11] M. Pal, "Multiclass approaches for support vector machine based land cover classification," *Computing Research Repository (CoRR)*, vol. abs/0802.2411, 2008. [Online]. Available: http://arxiv.org/abs/0802.2411

[12] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[13] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Springer, 2009.

[14] D. Coomans and D. Massart, "Alternative k-nearest neighbour rules in supervised pattern recognition: Part 1. k-nearest neighbour classification by using alternative voting rules," *Analytica Chimica Acta*, vol. 136, pp. 15–27, 1982.

[15] W. Li and A. W. Moore, "A machine learning approach for efficient traffic classification," *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 2007.

[16] R. Bar-Yanai, M. Langberg, D. Peleg, and L. Roditty, "Realtime classification for encrypted traffic," *Proceedings of the 9th International Conference on Experimental Algorithms (SEA'10)*, pp. 373–385, 2010.

[17] A. W. Moore, D. Zuev, and M. L. Crogan, "Discriminators for use in flow-based classification," Queen Mary and Westfield College, Department of Computer Science, Tech. Rep., 2005.

[18] W. C. Barto, "Classification of encrypted web traffic using machine learning algorithms," Department of the Air Force, Air University, 2013, http://www.dtic.mil/get-tr-doc/pdf?AD=ADA585816 Access date: April 18th, 2018.

[19] R. Alshammari and A. N. Zincir-Heywood, "Machine learning based encrypted traffic classification: Identifying SSH and Skype," *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009.

[20] Q. Sun, D. Simon, Y. Wang, W. Russell, V. Padmanabhan, and L. Qiu, "Statistical identification of encrypted web browsing traffic," *Proceedings of the IEEE Symposium on Security and Privacy*, 2002.

[21] B. Neimczyk and P. Rao, "Identification over encrypted channels," *BlackHat USA*, 2014.

[22] C. C. L. C. for Digital Investigation, "Amazon Echo forensics," https://lcdiblog.champlain.edu/wp-content/uploads/sites/11/2016/05/EDITED_Amazon_Echo_Report-1.pdf, 2016, access date: April 18th, 2018.

[23] S. Ostermann, "tcptrace," http://www.tcptrace.org/, 2003, access date: April 18th, 2018.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[25] A. Gebhart, "Google home review," https://www.cnet.com/products/google-home/review/, 2017, access date: April 18th, 2018.

[26] Z. Hall, "Homepod: Everything we know about the Apple smart speaker so far," https://9to5mac.com/2017/11/14/homepod-siri-speaker-launch-details/, 2017, access date: April 18th, 2018.

[27] M. Gurman, "Samsung targets first half of 2018 for smart speaker," https://www.bloomberg.com/news/articles/2017-12-14/samsung-is-said-to-target-first-half-of-2018-for-smart-speaker, 2017, access date: April 18th, 2018.