POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

IoT Forensics Made Easy: Extracting Information from Amazon Echo's Network Traffic with Feature Sniffer

Laurea Magistrale in Computer Science Engineering - Ingegneria Informatica

Author: Marco Marini

Advisor: Prof. Alessandro Enrico Cesare Redondi

Co-advisor: Fabio Palmese

Academic year: 2022-2023

Abstract

Different research works have been recently published showing Machine learning being used to extrapolate information from the encrypted network traffic, but there are challenges to face such as the lack of reliable ground truth and the difficulty in extracting useful features out of network traffic. To address these IoT Forensics challenges, researchers at Politecnico di Milano have developed a tool named Feature-Sniffer that extracts relevant data from the traffic by computing statistically relevant features on the spot. The goal of this work is to test the tool's efficacy in extracting information from the encrypted traffic produced by an Amazon Echo device, while the user interacts with it. We show that this is possible and furthermore we present a model able to identify interactions with Alexa with almost zero error. Combining this model with Feature-Sniffer allows, for future research, to simplify the training dataset building procedure for advanced IoT Forensic pipelines.

1. Introduction

The Internet of Things (IoT) technology is deeply embedded in our everyday life, with well over 13 billion connected devices worldwide, projected to reach 25.4 billion by 2030. The increasing use of IoT technology in everyday life has brought attention to the security and privacy aspect. Moreover, a new branch of digital forensics applied to the IoT, under the name of IoT Forensics, has recently taken importance with the goal of extracting information from such devices, which are so popular mainly thanks to their scalability and affordability. In fact, IoT devices do not require computational power, by forwarding the task directly to their remote servers. This design, however, poses security and privacy risks as the data exchange is vulnerable to attacks and users are often unaware of how their data is managed, used, and preserved. With the vast amount of private and sensitive information exchanged by IoT devices, it is essential to test and identify vulnerabilities as long as developing new techniques to ease IoT forensics analysis. One approach being studied involves using advanced Machine Learning techniques on encrypted network traffic between the device and its servers. In this work, we will explore that kind of technique on Amazon Echo, the world's most used IoT device. In particular, we will use Feature-Sniffer (FS), a tool that extracts in real-time,

from the network traffic, features useful to feed machine learning algorithms. We will analyze FS output to find a way of understanding when an interaction is happening and, after that, the features produced will be used to train models on three different tasks: auto-detect when an interaction is happening, detecting the genre of the speaker (robot or human) and its language (Italian or English).

2. Related Work

In the past few years, researchers have been investigating the possibility of extracting sensitive information from the encrypted traffic exchanged between IoT devices and servers. Many of these studies were focused on Amazon Echo and other similar voice assistant products. In particular, [5] and [3], found out that a neural network built on top of the network traffic exchange of an Amazon Echo device could accurately infer the exact question asked to Alexa with up to a 92% of accuracy. The authors of [2] instead, investigated the possibility of guessing the person who is asking the question to Alexa with an 80% accuracy on a binary classification and a 30% on a classification between 12 speakers. Additionally, [1] studied the possibility of classifying if the question towards Alexa is generated by a human voice or by a text-to-speech (TTS) bot obtaining an AUC value of around 0.95 for Random Forest and Gradient Boosting algorithms.

To mitigate these attacks, [5] and [3] have proposed using adaptive padding. However, this approach slows down the whole communication between the device and the server, which can negatively impact the customer experience. Moreover, transmitting useless data has a non-trivial energy cost, especially in battery-operated IoT devices.

Overall, the research works on this topic have proven that machine learning can produce great results when applied to encrypted data traffic, exploiting informative features such as packet lengths, inter-arrival times, number of packets sent, and so on. Nonetheless, collecting those features in a fast and easy way is difficult due to the lack of clean datasets of network traffic captures and the number of steps that have to be performed, on top of the network captures, before obtaining the final features that will feed the various machine learning models. All the research works above have mainly used robotic voices to produce their training dataset, this approach is very useful when there is the need to obtain a really big dataset in a short amount of time, but, on the other hand, introduces a slight bias between what is being experimented and what happens in the real world. In addition, all the papers we have gone through, exploited different techniques to understand whether an interaction was happening or not. The correct identification of the interactions with the device is a key requirement for this kind of experiment because it determines the precision of our final training dataset.

To recap, there is not a standard approach to this use case when it comes to the features collection and training data construction and, often, this step requires a quite considerable effort and might also cause a relevant delay between the reception of the traffic capture and the prediction of the model trained. For this reason, our aim has been to try to standardize and ease this process, designing a pipeline of the data manipulation combining Feature-Sniffer with a model that automatically identifies the interactions with the Echo device.

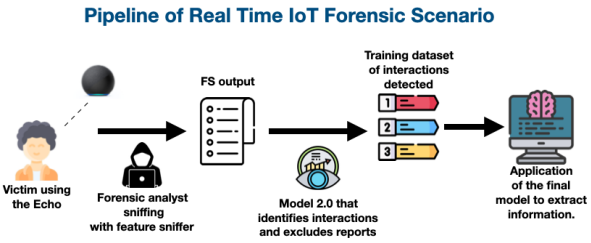

Figure 1: Data manipulation pipeline described above.

3. Data Collection

The first step we had to face has been the data collection phase; in order to produce interactions, some human voices were needed.

Since there were multiple classification tasks involved, we desired a dataset almost equally distributed between robots, humans, English and Italian speakers. We also tried to collect voices from users of different ages and equally spread between male and females, in order to have a quite acceptable variability of our training sample.

To collect the voices, we developed a simple web

form using HTML, Flask, and Python that users could access from their mobile phones or personal computers. The form contained 20 questions and users were required to submit three vocal recordings for each question, resulting in a total of 60 recordings per user.
In the end, we were able to collect the voices of 22 people for Italian questions and 9 for English ones for a total of 1860 recordings, which were examined to ensure their correctness. After the examination phase, around 1-2% of the dataset was removed because the recordings were not triggering Alexa. However, since the form asked for three audio samples per question, we always had at least one recording that worked fine.
For the robot voices, three different text-to-speech (TTS) APIs were used: Google TTS, pyttsx3 (based on Python), and Amazon Polly. Google and Python TTS produce more mechanical voices that are easy to identify as robotic, while Amazon Polly produces either less realistic or super realistic voices using the flag `"neural"`. In the Italian samples, both flags have been used, while in the English one, we used only neural voices. Since Polly offers many English different voices, but way fewer Italian ones, we used Google and Python TTS only to increase the Italian sample of robotic voices.

Final Dataset

	Number of Recordings
Human	1860
Robot	1380
Italian	1800
English	1440
TOTAL	3240

Table 1: Final dataset dimensions.

After the data collection and examination phase, we set up an Access Point with FS installed and running. Then, we connected the Amazon Echo to the WiFi network and, with a Python script to reproduce one audio every minute, we recorded all the interactions with the device. The one-minute pause was needed for Alexa to elaborate and reply to the questions since some of them needed up to a 40 seconds reply.

4. Interaction Detection (IDA)

Feature-Sniffer's output, as better explained in its presentation paper (see [4]), is composed by a csv having, in the columns, the features computed while, in the rows, the different time windows of traffic.

TS	Dev	CntTcpDLpckSz	SumTcpDLpckSz	MuTcpDLpckSz	MdnTcpDLpckSz
1655301092.670	f0:f0:a4:c:bc:26	37	2442	66.0000	66.0000
1655301093.670	f0:f0:a4:c:bc:26	45	3012	66.9333	66.0000
1655301094.670	f0:f0:a4:c:bc:26	98	79087	807.0102	1102.0000
1655301096.670	f0:f0:a4:c:bc:26	4	264	66.0000	66.0000
1655301097.670	f0:f0:a4:c:bc:26	11	11351	1031.9091	1466.0000

Figure 2: Piece of a csv produced by FS

For our experiments we have used only windows of 0.5 and 1 second. Of course, larger is the window, lighter will be the output file, but also it will be less informative with respect to a smaller window, which provides an higher level of detail. Since we had, on average, one minute of silence between an interaction and the next one, in the csv produced by FS were present a lot of useless windows including the traffic produced when no one interacted with the Echo. To train models on the classification tasks chosen it was necessary to identify the windows containing an interaction. In order to do so, we developed, in python, IDA (Interaction Detection Algorithm), an algorithm that, using different hyper parameters, is able to isolate the windows of traffic belonging to an interaction with the home assistant. Precedent researches to identify interactions used different parameters, like the inter arrival time between one packet and the successive one. However, using FS, this approach is not viable since this information is not available because features are computed on the range of time defined by the window chosen. Therefore, we couldn't obtain the time between a packet and another, we only had, for example, the average inter arrival time of the entire window. For this reason, it was necessary to find a new and reliable way to isolate interactions.
IDA exploits a feature produced by FS: the number of packets sent in uplink. Usually, if someone is speaking with the Echo, then the device has to send the encoded recording of the question to the server, this implies having a quite large uplink flow. By studying the CDF of this feature during a silence period, we were able to determine a threshold useful to understand whether

an interaction is happening or not. Basically, if a window of traffic has the feature's value over the threshold, this window, with an high probability, can be considered as part of an interaction. It is important to highlight that, being the questions of different length, an interaction is generally composed by more consecutive windows.
In the end, the threshold values used are 22 for the 0.5 second window and 40 for the 1 second window.
However, not all the windows over the threshold are interactions, there are some silence windows which present anomalous values. For this reason, in order to produce a training dataset as clean as possible, while developing IDA, we exploited the fact the audios were reproduced each 60 seconds. Thanks to this information, it was possible to approximately obtain the position of each interaction. By doing so, IDA computes a dynamic acceptability range of fall in for interactions and it is able to discard windows of silence over the threshold which fall in the non acceptable range of time.

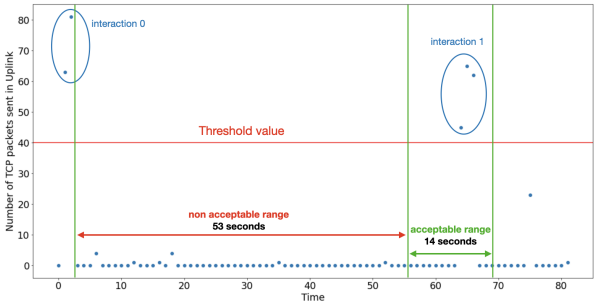

Figure 3: IDA's mechanism.

This is only a simplified explanation of IDA's functioning. The algorithm is more complex than that, there are in fact other variables that might alter the correct classification of interactions and we had to make adjustments and introduce other parameters in order to maximize its performance (everything is detailed explained in the thesis work). Anyhow, what matters is the result obtained. To verify the correctness of IDA's output we wrote a script that evaluates the length of the interactions detected and their time distance. Are expected more or less 60 seconds between an interaction and the next one and, on average, their lengths should be around 2 to 4 windows. Thanks to this script, which spotted anomalous interactions by exploiting those two characteristics, we were able to drastically reduce the examination time and extract information about IDA's performance. After a parameters tuning phase no more anomalous interactions have been spotted and we obtained our final training set of traffic windows. Finally, IDA computes the features of an interaction by using statistical operators (average, variance, standard deviation and sum) to combine the features' values of the different windows belonging to the same interaction.

Final Dataset with IDA

	# of Interactions
Data Collection	3240
Data Cleaning	3143
Detected by IDA	3075

Table 2: Final dataset with IDA.

As can be seen, we lost some interactions through the different phases, this loss is mainly due to questions that did not trigger Alexa and, partially, might be interactions with anomalous values that IDA has not detected, anyway we are speaking of only a really small percentage of the starting dataset.

5. Experimental Results

In the next subsections are reported the results obtained in each classification task by the models trained on the dataset produced by IDA.

5.1. Interaction Detection Task

The purpose of this task was to develop a model that could classify individual windows of traffic as part of an interaction or not. Differently from the other tasks, the training dataset used consisted of features related to the individual windows of traffic rather than to the whole interaction. Since we got way more windows of silence, to prevent an unbalanced training set that would have resulted in a model predicting only silence, part of the silence windows have been removed. The resulting dataset was composed

of 13,968 training windows, half of which were part of interactions, and half were silent. Several classification models have been compared, including Decision Tree, Random Forest, Gradient Boosting, Naive Bayes, K-Nearest Neighbor, and Support Vector Machine. Using windows of one second, all the models achieved an accuracy of over 95% on the test set, with Random Forest and Gradient Boosting reaching an accuracy of 99% and an AUROC value of 0.999. Regarding the 0.5-second window, the results obtained are really similar.

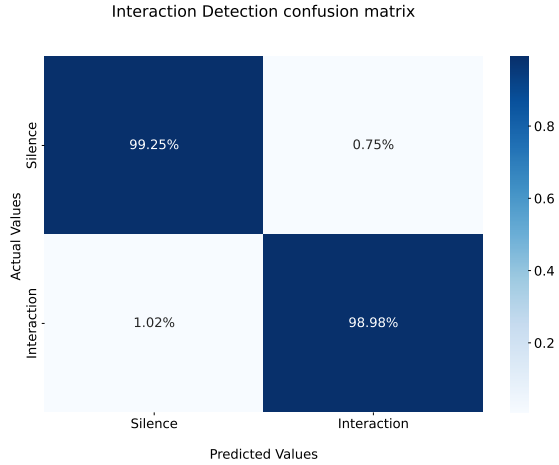

Figure 4: Interaction detection Gradient Boosting confusion matrix.

The results obtained are extremely good, but it is important to highlight that we are working on windows and not on interactions, therefore even one wrongly classified window could compromise the whole interaction classification by propagating the error. For this reason, it was really important to reach this kind of result.

5.2. Genre Classification Task

The aim of this task is to verify if it is possible to understand whether the question for Alexa was produced by a human or a robotic voice.
Different from the previous task, in this case, the training dataset is composed of the features related to the whole interaction. The best model revealed to be Gradient Boosting, which obtained an accuracy of 74% and an AUROC value of 0.79 for the 1-second window dataset. Like for all the tasks, we didn't notice a relevant difference in terms of results with respect to the 0.5-second window dataset.

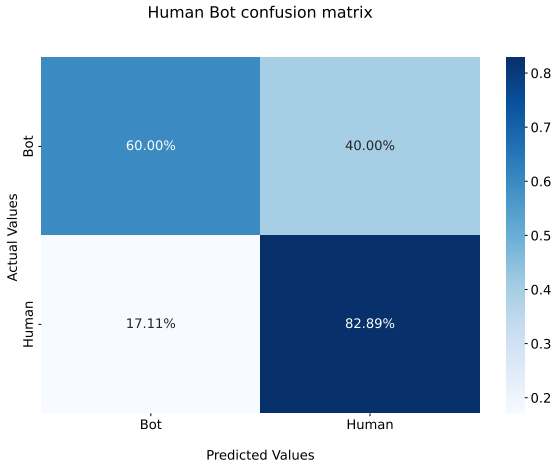

Figure 5: Genre detection Gradient Boosting confusion matrix.

This time results are not astonishing, anyway, it is important to underline that the task was particularly difficult since we have used neural robotic voices (which are almost indistinguishable from human ones) and the training set was composed by 54 different voices. Working with so many voices inevitably introduces a meaningful variability lowering the performances. Anyhow the overall performance, quite distant from random guessing, indicates that the information we looked for can be partially extracted with the use of FS.

5.3. Language Classification Task

This task focused on identifying the language of the speaker. In our use case, we worked with two languages: English and Italian.
While the human-or-bot task analyzed FS' ability to extract information regarding who asked the question, in this case we are analyzing its capability to obtain information about the content of the question. Under this aspect results are more promising, the best model has been Random Forest, which showed an accuracy of 85% and an AUROC value of 0.92. Even in this case is important to underline that we worked with 40 different questions (20 in Italian and 20 in English) and that some of the questions produced different answers every time they were asked (example: "Echo, pick a number"). This means that, even though the training dataset was characterized by a really large variability, the model learned the data pretty well. The reason could be probably attributed to the downlink flow which, more or less, remains constant

depending only by the question and not by the speaker.

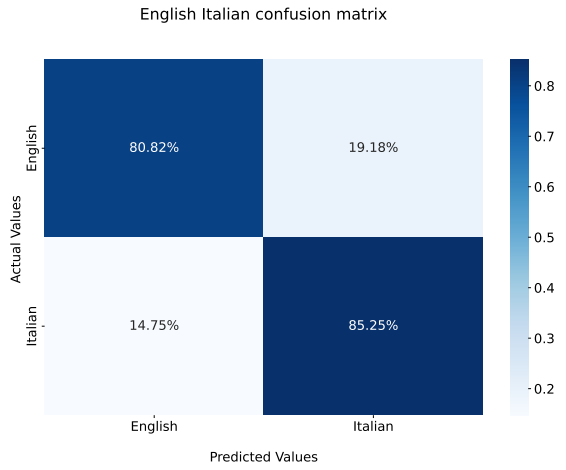

Figure 6: Language detection Random Forest confusion matrix.

	Accuracy	AUROC
Bot-Human	74%	0.791
Eng-Ita	84%	0.919
Interaction	99.8%	0.999

Table 3: Results for all the tasks studied.

6. Conclusions

A brief recap of the obtained results is reported in table 3.
To summarize what we have done: in this work we have first collected a dataset of voices of 31 different people, then we have used those voices to generate interactions with an Amazon Echo device; after that, thanks to Feature Sniffer, features related to the traffic produced have been computed; we have analyzed those features and developed an algorithm to exactly isolate the windows belonging to an interaction. This step has been necessary in order to be able to construct the training dataset taken as input by the different machine learning models. In the end, we have shown that Feature Sniffer can be used with success to extract information out of an encrypted traffic. Of course, it depends from the task and we have some limitations, like the fact that the features produced are finite, predefined and, moreover, are averaged with respect to a time range loosing granularity of information. In particular, FS use is more promising when related to the analysis of the traffic exchange content rather than the speaker's identity. The major advantages of this tool are: the possibility to extract in a quick and on demand way the features necessary to perform machine learning, the reduced size of the files produced, thanks to the direct use of csv files, avoiding pcap, and the capability to perform a live features extraction. Finally, we have developed a model that can be integrated with FS to automate the creation of a clean and precise dataset of interactions with an Amazon speaker. Thanks to this model, future researches on Amazon Echo devices' traffic could be focused mainly on the machine learning part, ignoring the training dataset construction and feature extraction problems.

References

[1] Alessandro Arensi. Machine learning classification of amazon echo dot users via encrypted voice traffic. 2021.

[2] Tiffany Kalin, Kerri Stone, and Tracy Camp. Amaze: Recognizing speakers with amazon's echo dot device. In *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 494–503. IEEE, 2019.

[3] Sean Kennedy, Haipeng Li, Chenggang Wang, Hao Liu, Boyang Wang, and Wenhai Sun. I can hear your alexa: Voice command fingerprinting on smart home speakers. In *2019 IEEE Conference on Communications and Network Security (CNS)*, pages 232–240. IEEE, 2019.

[4] Fabio Palmese, Alessandro EC Redondi, and Matteo Cesana. Feature-sniffer: Enabling iot forensics in openwrt based wi-fi access points. *arXiv preprint arXiv:2302.06991*, 2023.

[5] Chenggang Wang, Sean Kennedy, Haipeng Li, King Hudson, Gowtham Atluri, Xuetao Wei, Wenhai Sun, and Boyang 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*, pages 254–265, 2020.