Using TCP/UDP Packets to Fingerprint Websites on the Tor Browser

Abraham Adberstein, John Fang, Yasmine Zakout

Fall 2016

1 Introduction

As increasingly more people use the Internet to conduct private affairs such as research, banking, and interpersonal communication, privacy is quickly becoming more of a priority. Not only do users wish to protect against malicious attackers from discovering their private information such as credit card numbers, but also from their ISP or government from spying on their browsing habits. One such way of accomplishing this task is to use Tor, a web browser which obfuscates the user's identity by encrypting outgoing packets numerous times, and then passing them through a circuit of randomly selected nodes while simultaneously decrypting until the packet arrives at the destination. While this approach is good in theory, recent research has shown that the Tor network is not as anonymous as one might think.

In this paper, we research the effectiveness of website fingerprinting attacks against users using the Tor Browser. More specifically, we demonstrate that a fingerprint attack using information gained at the transport layer of the Internet protocol suite achieves moderately-high level of success at 72% accuracy on average. Our code is available on Github [4].

2 Background Information

2.1 The Internet Protocol Suite

The Internet can be conceptually divided into several layers of abstractions, with each layer providing different interfaces and services to the one above it. There are various protocols that implement the interfaces of each layer. The full suite is (from lowest layer to highest) the link layer, network layer, transport layer and finally the application layer. Whenever data is sent across the Internet, it is wrapped in metadata for each of these protocols, and this metadata is added and removed again as the data moves from the source to its destination.

Two layers are of relevance to this research paper, the network layer and the transport layer. The network layer is mainly responsible for sending packets from one router to another, and the Internet Protocol (IP) is the most common protocol used. Above this layer is the transport layer, which is responsible for ensuring that packets move correctly and efficiently across a series of routers to get from the source to the destination. To do this, it must make connections, perform error corrections, control congestion, etc. Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are two well-known protocols for this layer.

2.2 Tor Browser

The Tor Browser is a web browser that is designed to allow a user to browse the Internet anonymously, such that someone else observing your Internet connection or analyzing your traffic should not be able to know what sites you are visiting. The way that Tor works is by using onion routing, which involves sending data to three intermediary nodes in the Tor network between the source and destination. The high level idea behind this scheme is that none of the

nodes along the way is able to see both the endpoint addresses and the content that is being addressed at the same time, since this information is stored at different layers of encryption and the layers are peeled away as the data is passed along the nodes (like an onion).

3 Related Works

Although directly breaking the multiple layers of encryption in a Tor connection seems unrealistic and beyond our scope, it has been demonstrated in the past that a malicious person can still infer a user's browsing patterns using website fingerprinting. There are a handful of previous papers which look into fingerprinting websites accessed with the Tor Browser. These papers measure different features to distinguish websites, and also use different optimizations.

In 2011, Panchenko et al. [1] first demonstrated that a website fingerprinting attack against Tor is possible with a success rate of 55% for a closed world-type attack and 73% for an open world-type attack. In their research, they collect information about IP layer packets such as order, timing, etc. They identify some key features for identification and remove useless data such as acknowledgment packets, and use these features with Weka's SVM supervised learning methods. They also describe two types of attack that an attacker may way to perform: 1) closed world, which tries to identify websites out of a limited set of possibilities, and 2) open world, which tries to determine if any website is one of a given set of websites.

In 2013, Wang et al. [2] performed a different fingerprinting attack, which uses Tor cells rather than IP packets. One of their key insights is that Tor actually uses units called Tor cells to send data, and IP/TCP packets simply wrap around these. Their work achieved better results than Panchenko's work with a 95% accuracy rate for open-world and 91% for closed-world attacks. The classification technique they used was distance based metrics, which uses an SVM classifier to compare how similar traffic instances are.

Similarly, in 2014, Wang [3] demonstrated another attack with a shorter training time. Their design relied on using the k-Nearest Neighbors classifier with weight adjustment on their feature set consisting of attributes of the Tor cells, such as general features, unique packet lengths, packet ordering, concentration of outgoing packets, and bursts. The authors demonstrated that a fingerprinting attack still proved to be quite effective even when several defenses were put into place, including packet padding, order randomization, and traffic morphing. They achieve an 85% success rate in a close-world type attack.

Our paper's contribution differs from prior research in that our attack takes place on the transport layer, as opposed to others which run at the network layer level or using Tor cells. The information available varies by layer, so our research seeks to demonstrate that TOR's weaknesses can be exploited in more ways than past research has shown.

4 Procedure

4.1 Experimental Setup

We ran our attack on a VMware 8.5.2 virtual machine running 64-bit Kali Linux Version 2016.2. We collected data with a script written with Python 3.5.2. For the machine learning algorithms, we used libraries from scikit-learn, which was part of Anaconda 4.2.0.

4.2 Data Collection

In order to collect our data, we wrote a Python sniffer script. This script uses pyshark, a Python wrapper around the network protocol analyzer TShark, and Selenium, a suite for automating web browsers. The Selenium module automates requests

to the Tor Browser drivers. The script uses one thread to open an instance of the Tor Browser and requests a specific URL, while another thread runs TShark in the background to capture the packets that were received during this time. Every time a request is made to a corresponding website we restarted the Tor Browser and the sniffer to keep the data collection clean. From the packets, we extracted out the protocol, time of receiving, packet length, destination IP address, and destination port. These parameters served as the measurements to fingerprint each website.

We selected most of our websites from the Alexa Top 500 Global Sites list [5], similar to what past research papers have done. We think this is a realistic model because these websites are the most used websites, and thus may reflect the type of websites an average user might visit. We also included some other websites that we picked ourselves, in order to see if the attack is also successful for websites that aren't as large as the ones on Alexa Top 500 presumably are. The full list of websites we looked into can be found in the appendix. Overall, we surveyed 30 websites and took 40 samples per website. A full list can be found in the appendix.

We did minimal optimizations on the data. The one change we made was to get rid of bad samples. When opening Tor, a request can sometimes fail which is noticeable when the packet trace is extremely short relative to all the others from the same website. In order to even out the number of samples used for each website, we took 37 good samples from each website since one website produced at most 3 bad traces out of 40.

4.3 Machine Learning

We used several machine learning techniques to classify the data in order to demonstrate the similarities between traces of individual websites, with the goal of eventually fingerprinting them.

We wrote another Python script that would take the output of the data collection scripts, and record the number of unique connections (a destination IP address and destination port pair), the time deltas between each packet, and the frequency of each packet size for each sample recording. We chose these features because these are the ones that will vary depending on the amount of data that the website must transfer.

Finally, the script evaluated the similarity of the traces using machine learning and classification techniques from the open source Python library, scikit-learn. We used this library to split the raw data into "training sets" and "test sets". The classifiers were fit with the training set, and then it would be asked to classify the test set. This classification was repeated 10 times in order to vary the training and test sets. We then averaged the results from the 10 trials and used these samples to create a confusion matrix and show their accuracy.

In total, we tried out three different machine learning techniques: Linear Support Vector Machines (SVM), k-Nearest Neighbors, and Naive Bayes method. We selected these particular techniques because we knew they were used in the past research papers, so this led us to believe that they would also fit our attack, which is very similar in nature. We try all three techniques to decrease the likelihood that our attack failed simply because we selected a bad classification technique.

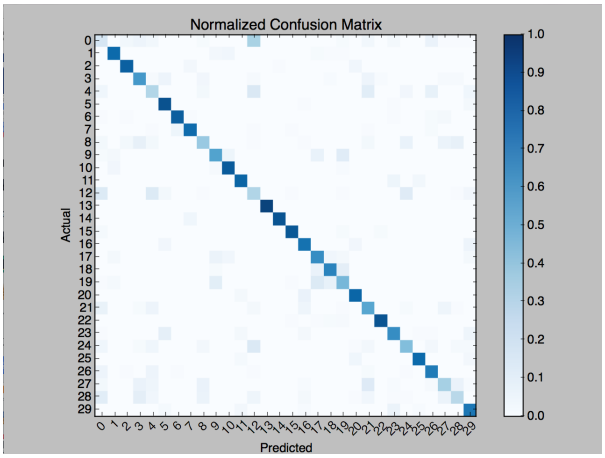
5 Results

5.1 Data

Below, we present the results of our attacks using three different machine learning techniques in the form of a confusion matrix. The numbers on the axes correspond to the numbers on the websites in the appendix. We do not report all of our numbers in this paper due to space constraints and simply summarize the findings, although the exact data can be found at the aforementioned Github link under the "data" directory.

5.2 Linear Support Vector Machines

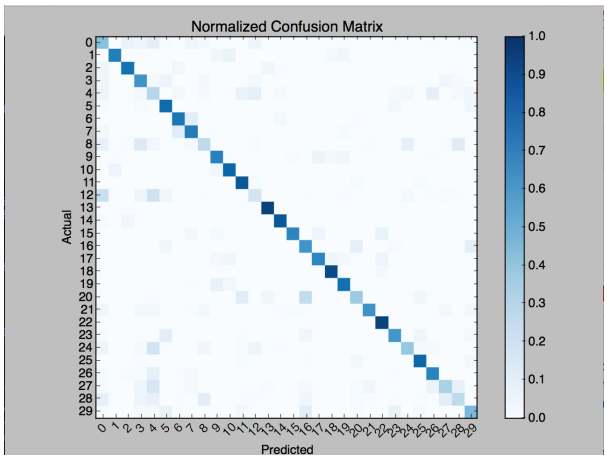
Support Vector Machines is a classification technique that technique works by plotting points of data in space based on its features, and computes the hyperplane that divides the two data sets such that the distance between the points and the divider is maximized. In order to classify an unknown data point, it will also plot this point in the space, and check which side of the dividers it is on.



Overall, the success rate for any particular website was around 72% on average, with a low of 22% and a high of 100%.

5.3 k-Nearest Neighbors

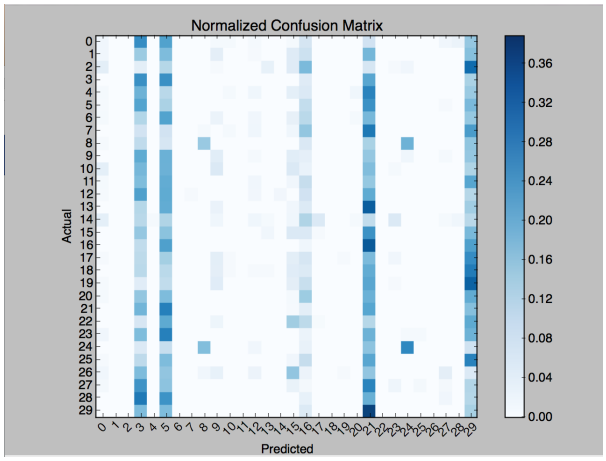
The k-Nearest Neighbors classification algorithm is a simple machine learning algorithm that takes in a feature vector to classify and outputs the class the algorithm thinks it belongs to. It works by computing the distances between a test feature vector and the feature vectors of known training data. Of the k nearest vectors, the class that has the most nearest feature vectors is determined to be the class of the test vector. In this project, we select k = 3.



Overall, the success rate for any particular website was around 69% on average, with a low of 23% and a high of 100%.

5.4 Naive Bayes

Naive Bayes is a family of classification algorithms which assumes that the values of each feature are independent of the values of other features, and each feature contributes independently to the identity of the class. It uses the Bayes probability model in conjunction with a decision rule to create a function to test the data.



Overall, the success rate for any particular website was around 5% on average, with a low of 0% and a

high of 28%.

6 Analysis

From our results, we can see that our best attack had a fairly good success rate. The classification technique that produced the best results for this type of attack was linear SVM with 72% success rate. k-Nearest Neighbors was not far behind with a 68% success rate. Finally, Naive Bayes had the lowest success rate at 5% which is only slightly higher than randomly guessing, which would have a success rate of 3% on average.

We cannot directly compare our numbers with past papers, as they used more websites than us and more websites would likely decrease the accuracy of the classification. Our attack is likely to be less effective than Tor cells even with additional websites, and we are unsure if it would still be better than Panchenko's network layer attack. From our readings and the nature of transport layer-level attacks, we are able to offer a few explanations for factors that would affect our results and why our attacks are less effective than attacks using Tor cells.

One reason why our project is likely to be less effective than past attacks is due to the fact that our attack was chosen to be on the transport layer. Intuitively, since Tor actually communicates using Tor cell units, and Tor cells are wrapped in a TCP/UDP packet, measuring the packets is likely to provide less information than a Tor cell. The actual data is hidden behind another layer of abstraction, and the TCP/UDP packets may hide important information depending on how the Tor cells are packed into the packets. Thus, this is one likely explanation for why our accuracy rate is not quite as high relatively.

Furthermore, we did not have the time to make certain optimizations that would have removed extraneous and useless information. For example, since TCP needs to maintain a reliable connection, it needs to send acknowledgment packets every so often. These packets do not actually provide any meaningful data and only serves to make the different URLs look more similar, which is why past research papers made sure to take out these packets before analysis. Since we did not do this, it is expected that this would also decrease our accuracy.

Additionally, our classifier does not assign weights to the various parameters that we used to analyze our data. Tor pads packets to provide a uniform size when transmitting, and while the related work included packet size as a parameter in their classifiers as well, they weighted it much less heavily than other parameters. For example, in the 2014 paper by Wang et. al, packet length was given the minimum weight possible in their classification algorithm. Our script simply gives the machine learning algorithm all the data without specifying this type of information, so it is very possible our classification weighs information that is not quite as useful heavier than is appropriate.

7 Conclusion and Future Work

In this paper, we have demonstrated that a website fingerprinting attack on websites visited with the Tor browser is able to identify websites with some level of success, even on an unoptimized attack at the transport layer. We collected the TCP packets received by Tor, and use a few of the features for machine learning classification. We tried out a variety of techniques, including SVM, k-NN and Naive Bayes. These techniques showed us that linear SVM is the most optimal for this type of attack, and it can classify one of our websites accurately with a success rate of 72% on average.

From our results, we conclude that the transport layer is another area vulnerable for attacks against people using Tor for anonymous browsing. Although our attack isn't quite as effective as the optimized attacks using Tor cells, the results are still significantly better than random guessing and could potentially be further exploited.

Some things that we think would be worth additional investigation in the future is how much our attacks can improve by applying aforementioned op-

timizations that other research papers included but that we didn't. These include removing unnecessary data, and looking into which features are more distinct so that they can be weighted more heavily in the machine learning classifiers.

A Top 20 Websites from Alexa

7	Google	google.com
29	YouTube	www.youtube.com
6	Facebook	www.facebook.com
2	Baidu	www.baidu.com
14	Live	www.live.com
25	Vk	www.vk.com
11	Instagram	www.instagram.com
13	Linkedin	www.linkedin.com
20	Reddit	www.reddit.com
5	Ebay	www.ebay.com
23	Tumblr	www.tumblr.com
16	Microsoft	www.microsoft.com
22	Stackoverflow	www.stackoverflow.com
19	Pornhub	www.pornhub.com
10	imgur	www.imgur.com
1	Apple	www.apple.com
17	Netflix	www.netflix.com
15	Mailru	www.mailru.com
9	IMBD	www.imdb.com
18	Office	www.office.com

B Other 10 Websites

26	Wiki 1	www.tinyurl.com/glolqhu
27	Wiki 2	www.tinyurl.com/6ubun3j
28	Yahoo	www.yahoo.com
12	Inverarte	www.inverarteartgallery.com
0	Amazon	www.amazon.com
24	Utexas	www.utexas.edu
8	IGN	www.ign.com
21	Sparknotes	www.tinyurl.com/metz9
4	Defcon	defcon.ru
3	CNN	www.cnn.com

References

[1] 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, 2011.*

[2] T. Wang and I. Goldberg. *Improved Website Fingerprinting on Tor*

[3] 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, 2014*

[4] https://github.com/adle29/secreta

[5] http://www.alexa.com/topsites