The Implementation of a Twitter Stalker

Austin Emery, Mercedes Anderson, and Nickolas Johnson.

***Abstract*—Recent news of data breaches as seen with Equifax and most recently Facebook, have called into question the safety and privacy of user data online. Social media users are quick to post images of themselves, their loved ones, and their food on multiple platforms, and are regularly checking in with statuses through out the day. From a user perspective, it's nothing but harmless fun. From a business perspective, it's a good opportunity for data mining and selling ads. From an attacker perspective, it's a chance for malicious collection of data and invasive research on targets. The threat model presented takes the shape of an attacker that wishes to stalk a Twitter user through out their day, and who constructs maps of where their victim has been in order to see places they frequent. Growing up, adults always warned us to be careful not to tell strangers where we were on the internet, and this project aimed to demonstrate exactly why.**

***Index Terms*—web crawler, Twitter, geo-location.**

I. Introduction

In this paper we will present a proof of concept for an unethical use of a web scrubber allowable by Twitter.

The threat brought on by the recent breach of user data in large social media has driven the need to demonstrate the existing security breaches in other social media applications. The use of the Twitter API can be used in ethical operations; however, it is susceptible to attacks, such as the attack demonstrated in this paper.

The motivation for this work is to educate the computer illiterate on the dangers of sharing geo-locations with applications on the internet.

In order to tackle the issue of lack of data security in the current state of the internet we simulated an attack on Twitter to determine the locations of users with their geo-location turned on while tweeting.

The Twitter Stalker is an unethical application of the Twitter API to collect and map the geo-location of a random user that has tweeted in the location of a specified attacker search. The Twitter Stalker can also be used in ethical ways such as allowing the user an option to show their travel path based on tweets. The ethical use would be for users who operate food trucks, drive promotional vehicles, or organize community events.

In order to go about this simulated attack we utilized the easily accessible documentation and tutorials: [1], [2], [3], and [4]. Part of the goal of this attack was to be able to implement it with using as few APIs as possible and without needing to sign up for any accounts that could track to us. In addition, we wanted to use information and resources that could be found by anyone on the internet. We could not avoiding signing up for a Twitter account in order to have access to the Twitter API, but were able to accomplish the visualization of the geo-location data through mapping without needing any sort of account.

Thanks to the internet for being so easy to take people's data
University of Nevada, Reno.
Manuscript due May 3, 2018.

II. Environment

A. Network and Environment Setup

The network used in this experiment is the Twitter domain. Using Twython, we are able to use Twitters API and search through tweets. To access Twitter's API we had to sign up through Twitter's development page and gather four different keys. This allowed us to use the keys in our program so that we could correctly use the API. Specifically the API allowed for the direct pull of tweets from online into a program for data analysis. In addition, Tweets could only be pulled from up to a week in the past, due to the limits of the standard access API.

Using the latitude and longitude information received from the tweets, we are able to construct a Google Maps URL that traces a path based on where the target was in the past seven days. Google Maps URLs only allow for a certain amount of waypoints in the trip being mapped, so the program had to limit the amount of points to eleven to fit the criteria. Google Maps URLs have the limitation that mobile devices can only work with URLs that have up to three waypoints, while non-mobile platforms can have up to nine waypoints. This means that at most, the attack can collect eleven points, one for an origin, one for a destination and nine for all the stops in between. Due to this limitation, we assumed the attacker was on a desktop, thus able to utilize all eleven points.

B. Installation and Execution

The use of the API required the download and installation of twython, a python wrap around for the Twitter API. There are several options for installation, including cloning the repository or using a third party installer such as pip or easy_install. For further details than what are provided below please see [4].

- git repository:
  $ git clone git://github.com/ryanmcgrath/twython.git
  $ python setup.py install
  A few errors encountered with the source code install were the omission of required modules such as "requests" and "oauth_requests" in the python environment.
- pip:
  $ pip install twython
  The third party installer pip installed all required modules.

- easy_install:
  $ easy_install twython
  The third party installed easy_install installed all required modules.

After the installation of the python wrap around twython, all that was needed was the inclusion keys generated by the Twitter API in the main file.

Once the script was written it was executed as a python file that accessed both the API and the URL builder without any requiring installation. To run the program, enter the line below into a terminal with the program in the current directory.

$ python stalker.py

III. The Twitter Stalker

A. Threat Model

1) Assumptions: It was assumed that:

- The user who's account was selected, was operating on a public account.
- The user who's account was selected, had the location turned on.
- The attacker only had access to a standard Twitter account and its API.
- The attacker only had access to Twitter user information that is public.
- The attacker only had access to the bare resources located on the internet, and did not spend money for additional resources through accounts or other means.
- The attacker is using the program on a desktop, due to URL way point limits on mobile.

2) Method: In order to appropriately 'stalk' a random user the program followed the flow presented in Figure 1. The attack begins by specifying a location to search tweets in, then scans 150 Tweets that are pulled from the Twitter API until a user with geo-locations enabled is found. If no such user is found, then a default user is provided for the purposes of showing off the full capabilities of the project. Once a user is specified, the program then searches that user's tweets specifically for geo-locations. Tweets up to a week old are looked at, and up to the eleven most recent are grabbed to be mapped. Once the latitude and longitude coordinates have been found from the pulled Tweets, they are converted into strings with the appropriate delimiters and format for Google Maps URLs and given to a web browser to open for viewing. There were three cases for generating maps in this way:

1) Case 1: There is only one location in the array.
   In this case the map will open with only one point on the map, with the one location being the origin and the destination.
2) Case 2: There are two locations in the array.
   In this case the map will open with the first point being the origin and the last point being the destination.
3) Case 3: There are more than two locations in the array.
   In this case the map will open with the first point being the origin, the last point being the destination, and all other points being waypoints.

Even in cases where there was only one location point, that still gives valuable information about where a user is located.

In order to make sure that the authors of the paper *were* in accordance with Twtiter's requirements concerning data, no data is stored passed the construction of the URL, and no twitter user names are linked to the URL. This was done to ensure that even in a simulated attack scenario, innocent internet users will remain safe from invasive stalking such as this attack.

3) Results: The results shown below are different search fields our program used. The different results come from different locations that were searched for. If a user was not found with the given search, our program defaults to a user that is from the Reno area, seems to use Twitter often and has their location turned on, their map can be seen in Figure 2. We were able to grab information from separate areas with varying results on each search. Figure 7 demonstrates that users can be followed across state lines, while other maps such as Figure 5 show what is most likely weekly commutes and errand running. Figures 3, 4, and 6 are additional maps gathered from scanning users in other specified locations all over the world. The points shown on the map each represent a different tweet from the past seven days. The starting point on the map was the first tweet that was available for us to use and the destination was the most recent tweet that was able to use. There is a constraint on the URL length, so we were only able to use up to 11 sets of latitude and longitude, with the first set being the origin and the last set being the destination.

B. Defense Model

A natural defense is already in affect against the simulated attack, as Twitter requires that users enable the location on their own, instead of auto enrolling users in the feature. In addition, Twitter's API developer guidelines explicitly states that developers need to be careful when working with sensitive user information, and that they are not to store user location information. However, there doesn't appear to be any checks or monitoring on Twitter's part to ensure that no location information is being misused.

IV. Conclusion

We showed exactly what we wanted, as we were able to implement an invasive online attack using only publicly available information that people have the option of turning off, but either do not have the knowledge or concern to do so. This was done with relative ease, and deliberately made more broad and random in order to respect the privacy of strangers. Imagine what could be done by people with access to the pro Twitter API, which has more data available to developers, and with the intent to do harm. Twitter informs developers not to store people's location, but does not prevent you from doing so. There are ethical reason to allow developers access to user's locations, such as personalized advertisements and emergency alerts. In addition, if Twitter was to police developers, we could expect to see a drop in innovation. So it is hard to say whether or not allowing developers access to this private information is the right thing to do. Twitter's standard settings of not allowing user location is, for the time being, the best solution. An interesting study would be to see if people

changed their behavior / how people would react if the map of their locations was then messaged to them privately. The authors had originally planned on doing this, but then after seeing just how exact and invasive the attack was determined to not keep records that could link users to their data in order to maintain their privacy.

ACKNOWLEDGMENT

The authors would like to acknowledge:

- Austin Emery for Twitter API setup and implementation, coordinate to string parsing, and writing.
- Mercedes Anderson for Google Map URL construction research and implementation, and writing.
- Nickolas Johnson for collected Tweet validation, and writing.

In addition the authors would like to acknowledge that programming in Python is a completely different beast than programming in C++, and are shocked that this was the first project they'd ever implemented in the language. It's amazing.

REFERENCES

[1] Google Maps URL Builder Documentation, *https://developers.google.com/maps/documentation/urls/guide*
[2] Twitter API Documentation, *https://developer.twitter.com/en/docs/geo/places-near-location/api-reference/get-geo-search*
[3] Social Metrics, *http://social-metrics.org/downloading-tweets-by-a-list-of-users-take2/*
[4] Twython Documentation, *https://twython.readthedocs.io/en/latest/usage/install.html*

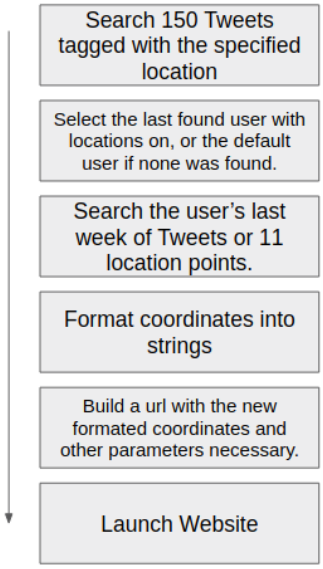

Fig. 1. The steps and process the program took in order to determine a user to track and make a map of.

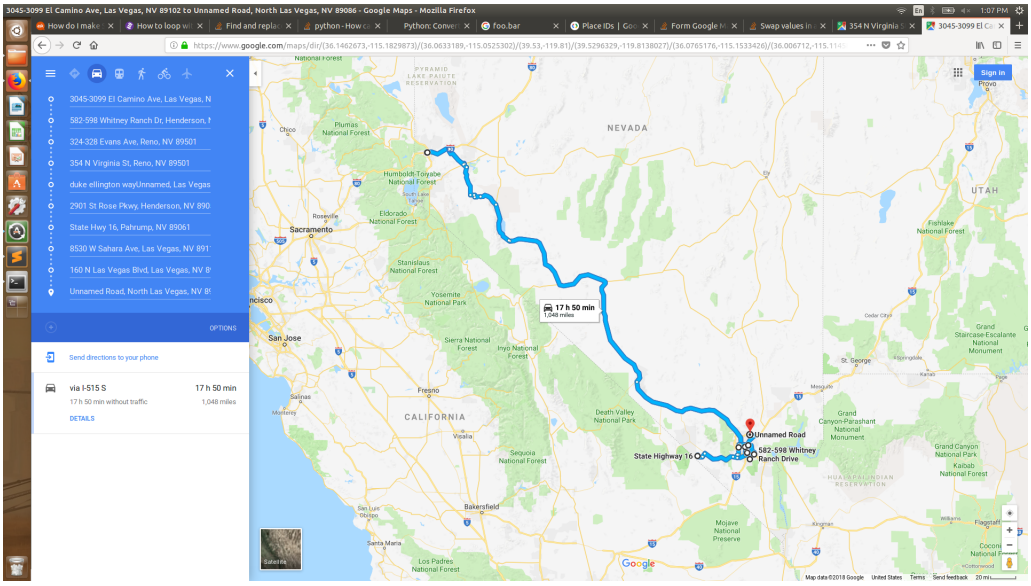

Fig. 2. This figure shows our default search results. The default search occurs when there are no users, in the search, that have their location turned on. This user was defaulted as they have shown to tweet often, and move locations regularly.

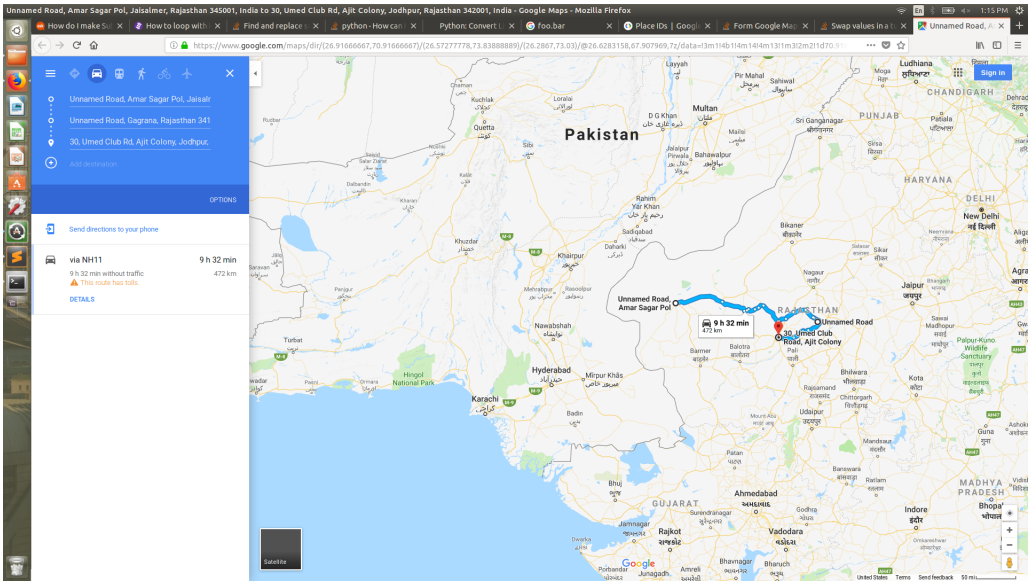

Fig. 3. This figure shows our search result for ”india”.

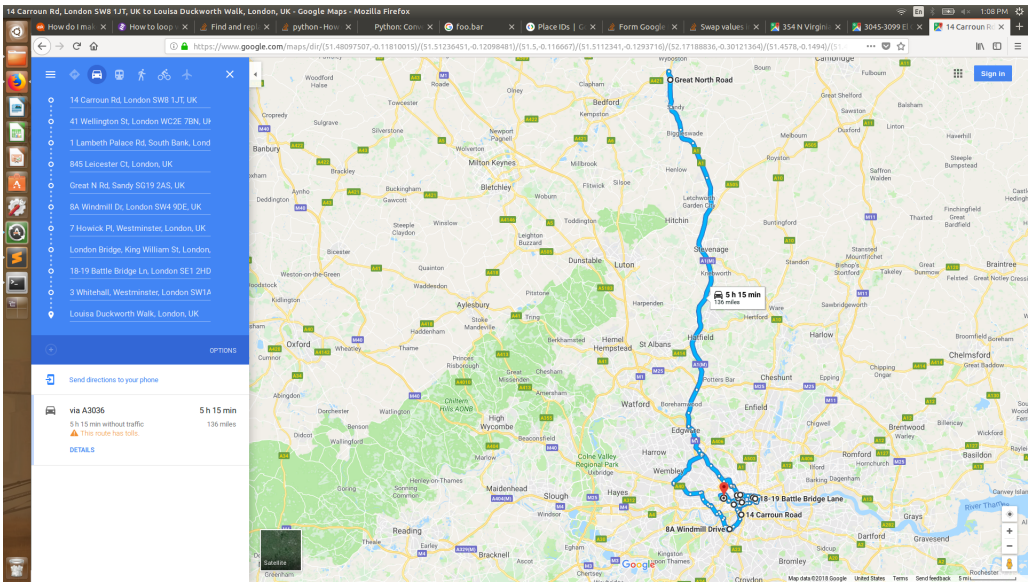

Fig. 4. This figure shows our search result for ”london”.

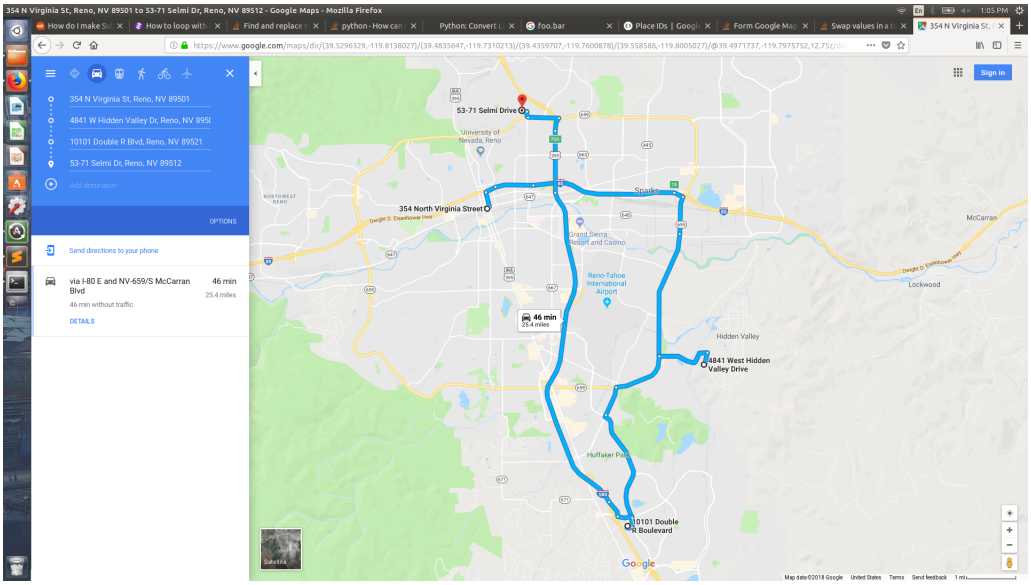

Fig. 5. This figure shows our search result for ”reno”.

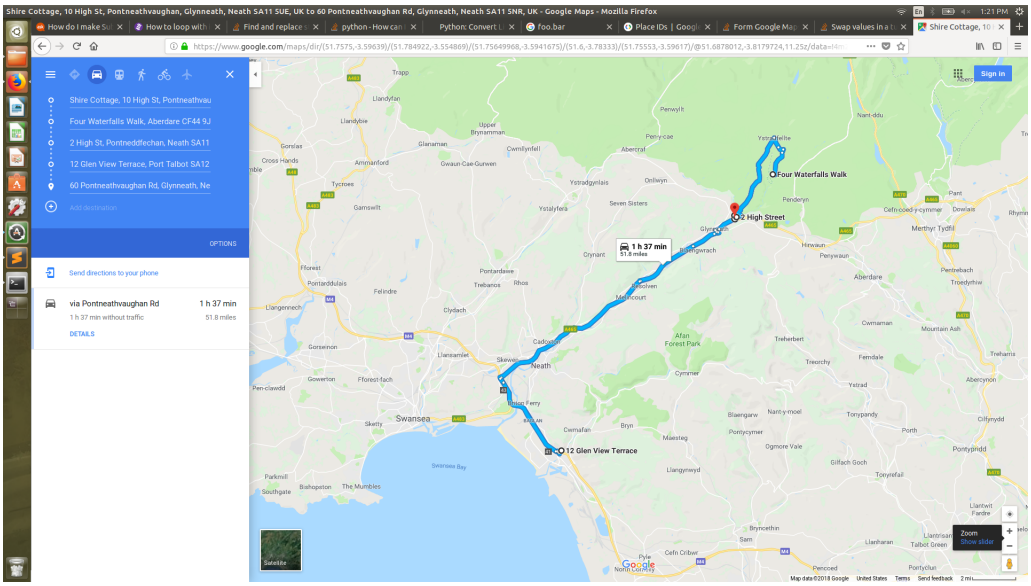

Fig. 6. This figure shows our search result for ”wales”.

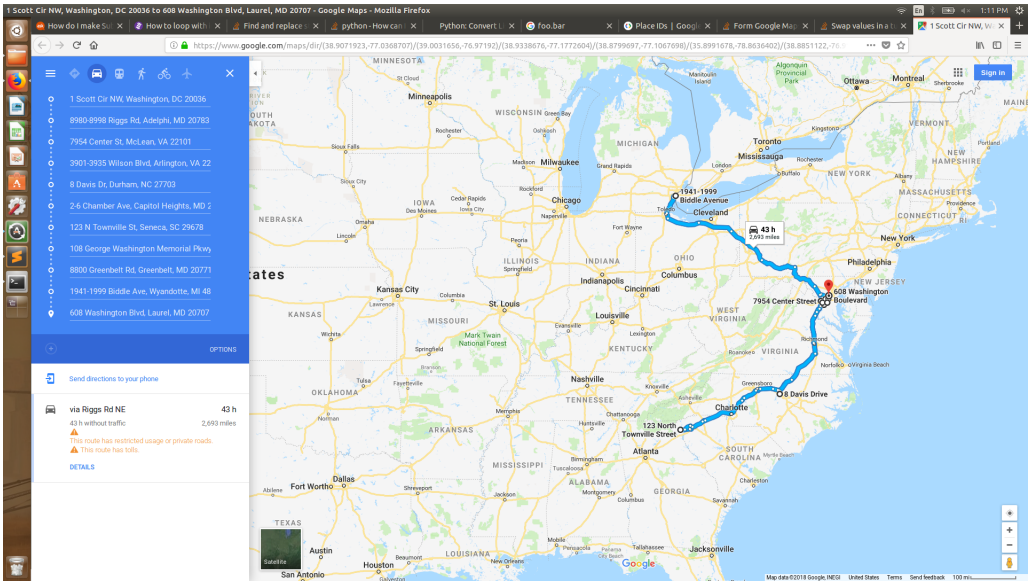

Fig. 7. This figure shows our search result for ”washington dc”.