ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 9, Issue 6, June 2022

Crowd Sourced Mobile Mapping Services

K.Priyadharshini (AP/CSE) 1, K. Mohanapriya, 2. P. Muthulakshmi, 3. S.Sathya,

1, 2, 3 –Department of Computer Science and Engineering,

Bharathiyar Institute of Engineering for Women, Deviyakurichi-636112.

ABSTRACT

Real-time crowd sourced maps like Waze give timely updates on traffic, congestion, accidents and points of interest. In this paper, we tend to demonstrate however lack of robust location authentication permits creation of software-based Sybil devices that expose Crowd sourced map systems to a spread of security and privacy attacks. Our experiments show that one Sybil device with restricted resources will cause disturbance on Waze, news false congestion and accidents and mechanically rerouting user traffic. More importantly, we tend to describe techniques to come up with Sybil devices at scale, making armies of virtual vehicles capable of remotely following precise movements for giant user populations whereas avoiding detection. To defend against Sybil devices, we tend to propose a brand new approach based on co-location edges; etch records that attest to the one-time physical co-location of a combine of devices. Over time , Co-location edges mix to make giant proximity graphs that attest to physical interactions between devices, permitting ascendable Detection of virtual vehicles. We tend to demonstrate the effectively of this approach exploitation large-scale simulations, and the way they will be wont to dramatically scale back the impact of the attacks. we've educated Waze/Google team of our analysis findings. Currently, we are inactive collaboration with Waze team to boost the protection and privacy of their system.

Index Term-Social networks, Location Authentication, Location Privacy, Mapping services.

I.INTRODUCTION

Crowd sourcing is indispensable as a period of time information gathering tool for today's on-line services. Reckon example map and navigation services. each Google Maps and Waze use periodic GPS readings from mobile devices to infer traffic speed and congestion levels on streets and highways. Waze, the foremost fashionable crowd sourced map service, offers users a lot of ways in which to actively share info on accidents, police cars, and even contribute content like piece of writing roads, landmarks, and native fuel costs. This and the ability to move with close users created Waze very popular, with associate calculable fifty million users once it had been non-inheritable by Google for a according $1.3 Billion USD in June 2013. Today, Google integrates hand-picked crowd sourced information (e.g. accidents) from Waze into its own Maps application. Unfortunately, systems that accept crowd sourced information as inherently prone to mischievous or malicious users seeking to disrupt or game the system [1]. as an example, business homeowners can smear competitors by refutation negative reviews on Yelp or Trip Advisor, and Foursquare users will forge their physical locations for discounts [3]. For location-based services, these attacks as doable as a result of there are not any wide deployed tools to manifest the placement of mobile devices. In fact, there are few effective tools these days to spot whether or not the origin of traffic requests as real mobile devices or software package scripts. The goal of our work is to explore the vulnerability of today's crowd sourced mobile apps against Sybil devices, software package scripts that seem to application servers as "virtual mobile devices."1 While one Sybil device will harm mobile apps through misbehavior, larger teams of Sybil devices will overwhelm traditional users and considerably disrupt any crowd sourced mobile app. In this paper, we tend to determine techniques that enable malicious attackers to reliably produce giant populations of Sybil devices victimization software package. Using the context of the Waze crowd sourced map service; we illustrate the powerful Sybil device attack, then develop and evaluate sturdy defenses against them. While our experiments and defenses ar designed with Waze (and crowd sourced maps) in mind, our results generalize to a wide range of mobile apps. With negligible modifications, our

techniques may be applied to services starting from Foursquare and Yelp to Uber, YikYak and Pokemon Go, permitting attackers to cheaply emulate various virtual devices with solid locations to Overwhelm these systems via wrongdoing. wrongdoing will vary from incorrectly getting coupons on Foursquare/Yelp, gaming the new user coupon system in Uber, imposing censorship on YikYak, to cheating within the game play of Pokemon Go. we tend to believe our proposed defenses may be extended to those services in addition. We discuss broader implications of our add Section nine.

A.SYBIL ATTACKS IN WAZE:

I rack the constant when the context of Waze, our experiments reveals variety of potential attacks by Sybil devices. Initial is straightforward event forgery, wherever devices will generate faux events to the Waze server, together with congestion, accidents or police activity which may have an effect on user routes. Second, we tend to describe techniques to reverse engineer mobile app genus APIs, so permitting wrongdoers to form light-weight scripts that effectively emulate an oversized variety of virtual vehicles that interact underneath the management of one attacker. We tend to decision Sybil devices in Waze “ghost riders.” These Sybil's will effectively amplify the officiousness of any attack, and overwhelm contributions from any legitimate users. Finally, we tend to discover a big privacy attack wherever ghost riders will wordlessly and invisibly “follow” and exactly track individual Waze users throughout their day, exactly mapping out their movement to figure, stores, hotels, filling station, and home. We tend to by experimentation confirmed the accuracy of this attack against our own vehicles, quantifying the accuracy of the attack against GPS coordinates. Exaggerated by a military of ghost riders, Associate in Nursing wrongdoer will probably thereabouts of lots of users, all with none risk of detection.

B.DEFENSES:

Prior proposals to handle the placement authentication downside have restricted attractiveness, due to reliance on widespread readying of specialized hardware, either as a part of physical infrastructure, i.e., cellular base stations, or as modifications to mobile devices themselves. Instead, we have a tendency to propose a sensible solution that limits the power of Sybil devices to amplify the potential harm incurred by any single wrongdoer. we have a tendency to introduce collocation edges, genuine records that attest to the one-time physical proximity of a combine of mobile devices. The creation of collocation edges is triggered opportunistically by the mapping service, e.g., Waze. Over time, collocation edges mix to make giant proximity graphs, network structures that attest to physical in tractions between devices. Since ghost riders cannot physically act with real devices, they cannot kind direct edges with real devices, solely indirectly through a little variety of real devices operated by the wrongdoer. Thus, the perimeters between associate wrongdoer and therefore the remainder of the network area unit restricted by the quantity of real physical devices she has, notwithstanding what percentage ghost riders area unit beneath her management. This reduces {the downside the matter} of detection ghost riders to a community detection problem on the proximity graph (The graph is seeded by a little variety of trustworthy infrastructure locations). Our paper includes these key contributions:

C.IMPACTS:

• We have a tendency to explore limits and impacts of single device attacks on Waze, e.g., artificial congestion and events. • we have a tendency to describe techniques to form light-weight ghost riders, virtual vehicles emulated by client-side scripts, through reverse engineering of the Waze app’s communication protocol with the server.

• we have a tendency to establish a replacement privacy attack that enables ghost riders to just about follow and track individual Waze users in real time, and describe techniques to supply precise, sturdy location updates.

• we have a tendency to propose and appraise defenses against ghost riders, victimization proximity graphs made with edges representing genuine collocation events between pairs of devices. Since collocation will solely occur between pairs of physical devices, proximity graphs limit the quantity of edges between real devices and ghost riders, so

analytic teams of ghost riders and creating them detectable victimization community detection algorithms .

Before the attack (left), Waze shows the fastest route for the user. After the attack (right), the user gets automatically re-routed by the fake traffic jam.

WAZE BACKGROUND

Waze is the most popular crowd sourced navigation app on smartphones, with more than 50 million users when it was acquired by Google in June 2013 [9]. Waze collects GPS values of users' devices to estimate real-time traffic. It also allows users to report onroad events such as accidents, road closures and police vehicles, as well as editing roads and even updating local fuel prices. Some features, e.g., user reported accidents, have been integrated into Google Maps [10]. Here, we briefly describe the key functionality in Waze as context for our work.

A. Trip Navigation:

Waze's main feature is assist users to find the best route to their destination and turn-by-turn navigation. Waze generates aggregated real-time traffic updates using GPS data from its users, and optimizes user routes both during trip planning and during navigation. If and when traffic congestions is detected, Waze automatically re-routes users towards an alternative. Crowd sourced User Reports. Waze users can generate realtime event reports on their routes to inform others about ongoing incidents. Events range from accidents to road closures, hazards, and even police speed traps. Each report can include a short note with a photo. The event shows up on the map of users driving towards the reported location. As users get close, Waze pops up a window to let the user "say thanks," or report the event is "not there." If multiple users choose "not there", the event will be removed. Waze also merges multiple reports of the same event type at the same location into a single event.

B. Social Function:

To increase user engagement, Waze supports simple social interactions. Users can see avatars and locations of nearby users. Clicking on a user's avatar shows more detailed user information, including nickname, ranking, and traveling speed. Also, users can send messages and chat with nearby users. This social function gives users the sense of a large community. Users can elevate their rankings in the community by contributing and receiving "thanks" from others.

ATTACKING CROWDSOURCED MAPS

In this section, we describe basic attacks to manipulate Waze by generating false road events and fake traffic congestion. Since Waze relies on real-time data for trip planning and route selection, these attacks can influence user's routing decisions. Attackers can attack specific users by forging congestion to force automatic rerouting on their trips. The attack is possible because Waze has no reliable authentication on user reported data, such as their GPS. We first discuss experimental ethics and steps we took to limit impact on real users. Then, we describe basic mechanisms and resources needed to launch attacks, and use controlled experiments on two attacks to understand their feasibility and limits. One attack creates fake road events at arbitrary locations, and the other seeks to generate artificial traffic hotspots to influence user routing.

A.ETHICS

Our experiments seek to understand the feasibility and limits of practical attacks on crowd sourcing maps like Waze. We are very aware of the potential impact to real users from any experiments. We consulted our local IRB and have taken all possible precautions to ensure that our experiments do not negatively impact real Waze users. In

particular, we choose experiment locations where user population density is extremely low (unoccupied roads), and only perform experiments at low-traffic hours, e.g., between 2am and 5am. During experiments, we continuously scan the entire experiment region and neighboring areas, to ensure no other Waze users (except our own accounts) are within miles of the test area. If any Waze users are detected, we immediately terminate all running experiments. Our study received the IRB approval under protocol# COMS-ZH-YA-010-7N. Our work is further motivated by our view of the risks of inaction versus risks posed to users by our study. On one hand, we can and have minimized risk to Waze users during our study, and we believe our experiments have not affected any Waze users. On the other hand, we believe the risk to millions of Waze users from pervasive location tracking (Section 5) is realistic and potentially very damaging. We feel that investigating these attacks and identifying these risks to the broad community at large was the ethically correct course of action. Furthermore, full understanding of the attacks was necessary to design a practical defense.

B.BASIC ATTACK:

Generating Fake Events Launching attacks against crowd sourced maps like Waze requires three steps: automate input to mobile devices that run the Waze app; control the device GPS and simulate device movements (e.g., car driving); obtain access to multiple devices. All three are easily achieved using widely available mobile device emulators. Most mobile emulators run a full OS (e.g., Android, iOS) down to the kernel level, and simulate hardware features such as camera, SDCard and GPS. We choose the GenyMotion Android emulator [11] for its performance and reliability. Attackers can automatically control the GenyMotion emulator via Monkeyrunnerd scripts [12]. They can generate user actions such as clicking buttons and typing text, and feed pre-designed GPS sequences to the emulator (through a command line interface) to simulate location positioning and device movement. By controlling the timing of the GPS updates, they can simulate any "movement speed" of the simulated devices. Using these tools, attackers can generate fake events (or alerts) at a given location by setting fake GPS on their virtual devices. This includes any events supported by Waze, including accidents, police, hazards, and road closures. We find that a single emulator can generate any event at arbitrary locations on the map. We validate these using experiments on a variety of unoccupied roads, including highways, local and rural roads (50+ locations, 3 repeated tests each). Note that our experiments only involve data in the Waze system, and do not affect real road vehicles not running the Waze app. Thus "unoccupied" means no vehicles on the road with mobile devices actively running the Waze app. After creation, the fake event stays on the map for about 30 minutes. Any Waze user can report that an event was "not there." We find it takes two consecutive "not there's" (without any "thanks" in between) to delete the event. Thus an attacker can ensure an event persists by occasionally "driving" other virtual devices to the region and "thanking" the original attacker for the event report.

C. CONGESTION AND TRAFFIC ROUTING

A more serious attack targets Waze's real-time trip routing function. Since route selection in Waze relies on predicted trip time, attackers can influence routes by creating "fake" traffic hotspots at specific locations. This can be done by configuring a group of virtual vehicles to travel slowly on a chosen road segment. We use controlled experiments to answer two questions. First, under what conditions can attackers successfully create traffic hotspots? Second, how long can an artificial traffic hotspot last? We select three low-traffic roads in the state of Texas that are representative of three popular road types based on their speed limit—Highway (65 mph), Local (45 mph) and Residential (25 mph). To avoid real users, we choose roads in low population rural areas, and run tests at hours with the lowest traffic volumes (usually 3-5AM). We constantly scan for real users in or nearby the experimental region, and reset/terminate experiments if users come close to an area with ongoing experiments. Across all our experiments, only 2 tests were terminated due to detected presence of real users nearby. Finally, we have examined different road types and hours of the day to ensure they do not introduce bias into our results. Creating Traffic Hotspots. Our experiment shows that it only takes one slow moving car to create traffic congestion, when there are no real Waze

users around. Waze displays a red overlay on the road to indicate traffic congestion (Figure1, right) Different road types have different congestion thresholds, with thresholds strongly correlated to the speed limit. The congestion thresholds for Highway, Local and Residential roads are 40mph, 20mph and 15mph, respectively. To understand if this is generalizable, we repeat our tests on other unoccupied roads in different states and countries. We picked 18 roads in five states in the US (CO, MO, NM, UT, MS) and British Columbia, Canada. In each region, we select three roads with different speed limits (highway, local and residential). We find consistent results: a single virtual vehicle can always generate a traffic hotspot; and the congestion thresholds were consistent across different roads of the same speed limit.

D. OUTVOTING REAL USERS.

Generating traffic hotspot in practical scenarios faces a challenge from real Waze users who drive at normal (non-congested) speeds: attacker's virtual vehicles must "convince" the server there's a stream of slow speed traffic on the road even as real users tell the server otherwise. We need to understand how Waze aggregated multiple inputs to estimate traffic speed. We perform an experiment to infer this aggregation function used by Waze. We create two groups of virtual vehicles: Ns slowdriving cars with speed Ss, and Nf fast-driving cars with speed Sf; and they all pass the target location at the same time. We study the congestion reported by Waze to infer the aggregation function. Note that the server-estimated traffic speed is visible on the map only if we formed a traffic hotspot. We achieve this by setting the speed tuple (Ss, Sf) to (10mph, 30mph) for Highway, (5, 15) for Local and (5, 10) for Residential. As shown in Figure 2, when we vary the ratio of slow cars over fast cars (Ns: Nf), the Waze server produces different final traffic speeds. We observe that Waze does not simply compute an "average" speed over all the cars. Instead, it uses a weighted average with higher weight on the majority cars' speed. We infer an aggregation function as follows:

$$Swaze = Smax \cdot max(Ns, Nf) + Savg \cdot min(Ns, Nf) \; Ns + Nf \text{ where } Savg = \frac{SsNs + Sf\,Nf}{Ns + Nf},$$

Smax is the speed of the group with Nmax cars. As shown in, our function can predict Waze's aggregate traffic speed accurately, for all different types of roads in our test. For validation purposes, we run another set of experiments by raising Sf above the hotspot thresholds (65mph, 30mph and 20mph respectively for the three roads). We can still form traffic hotspots by using more slow-driving cars (Ns > Nf), and our function can still predict the traffic speed on Waze accurately.

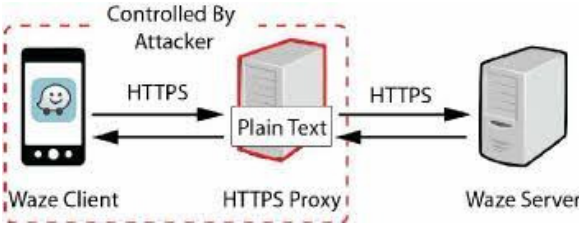
E.LONG-LASTING TRAFFIC CONGESTION.

A traffic hotspot will last for 25-30 minutes if no other cars drive by. Once aggregate speed normalizes, the congestion event is dismissed within 2-5 minutes. To create a long-lasting virtual traffic jam, attackers can simply keep sending slow-driving cars to the congestion area to resist the input from real users. We validate this using a simple, 50-minute long experiment where 3 virtual vehicles create a persistent congestion by driving slowly through an area, and then looping back every 10 minutes. Meanwhile, 2 other virtual cars emulate legitimate drivers that pass by at high speed every 10 minutes. We find the traffic hotspot persists for the entire experiment period.

F. IMPACT ON END USERS.

Waze uses real-time traffic data to optimize routes during trip planning. Waze estimates the end-toend trip time and recommends the fastest route. Once on the road, Waze continuously estimates the travel time, and automatically reroutes if the current route becomes congested. An attacker can launch physical attacks by placing fake traffic hotspots on the user's original route. While congestion alone does not trigger rerouting, Waze reroutes the user to a detour when the estimated travel time through the detour is shorter than the current congested route. We also note that Waze data is used by Google Maps, and therefore can potentially impact their 1+ billion users [13]. Our experiment shows that artificial congestion do not appear on Google Maps, but fake events generated on Waze are displayed on Google Maps without verification, including "accidents", "construction" and "objects on road". Finally, event updates are synchronized on both services, with a 2-

minute delay and persist for a similar period of time (e.g., 30 minutes).



Using a HTTPS proxy as man-in-the-middle to intercept traffic between Waze client and server.

SYBIL ATTACKS

So far, we have shown that attackers using emulators can create "virtual vehicles" that manipulate the Waze map. An attacker can generate much higher impact using a large group of virtual vehicles (or Sybil's) under control. In this section, we describe techniques to produce light-weight virtual vehicles in Waze, and explore the scalability of the group-based attacks. We refer to large groups of virtual vehicles as "ghost riders" for two reasons. First, they are easy to create en masse, and can travel in packs to outvote real users to generate more complex events, e.g., persistent traffic congestion. Second, as we show in §5, they can make themselves invisible to nearby vehicles.

A.CREATING SYBIL DEVICES

We start by looking at the limits of the large-scale Sybil attacks on Waze. First, we note user accounts do not pose a challenge to attackers, since account registration can be fully automated. We found that a single-threaded Monkey runner script could automatically register 1000 new accounts in a day. The limiting factor is the scalability of vehicle emulation. Even though emulators like GenyMotion are relatively lightweight, each instance still takes significant computational resources. For example, a MacBookPro with 8G of RAM supports only 10 simultaneous emulator instances. For this, we explore a more scalable approach to client emulation that can increase the number of supported virtual vehicles by orders of magnitude. Specifically, we reverse engineer the communication APIs used by the app, and replace emulators with simple Python scripts that mimic API calls.

B.REVERSE ENGINEERING WAZE APIS.

The Waze app uses HTTPS to communicate with the server, so API details cannot be directly observed by capturing network traffic (TLS/SSL encrypted). However, an attacker can still intercept HTTPS traffic, by setting up a proxy [14] between her phone and Waze server as a man-inthe-middle attack [15], [16]. As shown in Figure 3, an attacker needs to pre-install the proxy server's root Certificate Authorities (CA) to her own phone as a "trusted CA." This allows the proxy to present self-signed certificates to the phone claiming to be the Waze server. The Waze app on the phone will trust the proxy (since the certificate is signed by a "trusted CA"), and establish HTTPS connections with the proxy using proxy's public key. On the proxy side, the attacker can decrypt the traffic using proxy's private key, and then forward traffic from the phone to Waze server through a separate TLS/SSL channel. The proxy then observes traffic to the Waze servers and extracts the API calls from plain text traffic. Hiding API calls using traffic encryption is fundamentally challenging, because the attacker has control over most of the components in the communication process, including phone, the app binary, and the proxy. A known countermeasure is certificate pinning [17], which embeds a copy of the server certificate within the app. When the app makes HTTPS requests, it validates the server-provided certificate with its known copy before establishing connections. However, dedicated attackers can extract and replace the embedded certificate by disassembling the app binary or attaching the app to a debugger [18], [19]. Once we obtain the knowledge of Waze APIs, we can build extrimly lightweight Waze clients using python scripts, allocating one thread for each client. Within each thread, we login to the app using a separate account, and maintain a live session by sending periodic GPS coordinates to the Waze server.

C.POTENTIAL DEFENSES AGAINST SYBIL DEVICES

While attackers can easily create lightweight Sybil devices, it is nontrivial for services providers to

effectively detect and defend against them. Below we discuss possible ways to reliably authenticate mobile devices, and highlight the key challenges to do so. Email Verification. A straight-forward approach is to authenticate a mobile device via an email account. However, attackers may create fake email accounts automatically or purchase them in bulks from black-markets [20]. This approach has limited effect.

1. SMS VERIFICATION.

Two-factor Authentication can be used to verify phone numbers. The latest Waze app already requires SMS verification during account registration. However, attackers can bypass this using disposable phone numbers or temporal SMS services [21].

2. CAPTCHA

Service providers can use CAPTCHAs to test whether a phone is operated by a human user or a computer script. This approach has key limitations too. First, solving CAPTCHAs on smart phones can be distracting and annoying to legitimate users. Second, attackers can leverage crowd sourced CAPTCHA farms to solve CAPTCHAs in real time [22].

3. IMEI VALIDATION

Service providers may also consider validating the unique identifier of the phone such as IMEI. But the challenge is there are already public IMEI databases [23] or fake IMEI generators [24] that can help attackers to spoof the identifier.

4. DEVICE FINGERPRINTING

Researchers have proposed to use motion sensors to fingerprint smart phones [25]. The idea is that Smartphone sensors such as accelerometers and gyroscopes usually have anomalies in their signals due to manufacturing imperfections. Such signal anomalies can be used to uniquely fingerprint the phone. However, a more recent result shows that fingerprinting accuracy would drop quickly for a large number of devices (e.g., 100K) [26]. This technique is still not reliable enough to authenticate mobile devices.

5. IP VERIFICATION

Finally, service providers can also check if the device's IP is an actual mobile IP (or a suspicious web proxy). However, attacker can overcome this by routing their traffic through a cellular data plan. We find that authenticating individual mobile devices is very challenging. As long as attackers have full controls on the client side, they could (easily) forge the data needed for authentication. In the later section (§6), we will describe our method to detect groups of Sybil devices.

D.SCALABILITY OF GHOST RIDERS

Ghost riders are fully functional Waze clients and they are highly scalable. Each ghost rider is scripted not only to report GPS to Waze server, but also report fake events using the API. We run 1000 virtual vehicles on a single Linux Dell Server (Quad Core, 2GB RAM), and find that at steady state, 1000 virtual devices only introduces a small overhead: 11% of memory usage, 2% of CPU and 420 Kbps bandwidth. In practice, attackers can easily run tens of thousands of virtual devices on a commodity server. Finally, we experimentally confirm the practical efficacy and scalability of ghost riders. We chose a secluded highway in rural Texas, and used 1000 virtual vehicles (hosted on a single server and single IP) to generate a highly congested traffic hotspot. We perform our experiment in the middle of the night after repeated scans showed no Waze users within miles of our test area. We positioned 1000 ghost riders one after another, and drove them slowly at 15 mph along the highway, looping them back every 15 minutes for an entire hour. The congestion shows up on Waze 5 minutes after our test began, and stayed on the map during the entire test period. No problems were observed during our test, and tests to generate fake events (accidents etc.) also succeeded.

USER TRACKING ATTACK

Next, we describe a powerful new attack on user privacy, where virtual vehicles can track Waze users continuously without risking detection themselves. By exploiting a key social functionality in Waze, attackers can remotely follow (or stalk) any individual user in real time. This is possible with single device emulation, but greatly amplified with

the help of large groups of ghost riders, possibly tracking large user populations simultaneously and putting user (location) privacy at great risk. We start by examining the feasibility (and key enablers) of this attack. We then present a simple but highly effective tracking algorithm that follows individual users in real time, which we have validated using real life experiments (with ourselves as the targets). The only way for Waze users to avoid tracking is to go "invisible" in Waze. However, doing so forfeits the ability to generate reports or message other users. Waze also resets the invisible setting every time the app is opened [27].

A.FEASIBILITY OF USER TRACKING

A key feature in Waze allows users to socialize with others on the road. Each user sees on her screen icons representing the locations of nearby users, and can chat or message with them through the app. Leveraging this feature, an attacker can pinpoint any target who has the Waze app running on her phone. By constantly "refreshing" the app screen (issuing an update query to the server), an attacker can query the victim's GPS location from Waze in real time. To understand this capability, we perform detailed measurements on Waze to evaluate the efficiency and precision of user tracking.

B.TRACKING VIA USER QUERIES

A Waze client periodically requests updates in her nearby area, by issuing an update query with its GPS coordinates and a rectangular "search area." This search area can be set to any location on the map, and does not depend on the requester's own location. The server returns a list of users located in the area, including user ID, nickname, account creation time, GPS coordinates and the GPS timestamp. Thus an attacker can find and "follow" a target user by first locating them at any given location (work, home) and then continuously following them by issuing update queries centered on the target vehicle location, all automated by scripts.

C.OVERCOMING DOWN SAMPLING.

The user query approach faces a down sampling challenge, because Waze responds to each query with an "incomplete" set of users, i.e., up to 20 users per query regardless of the search area size. This down sampled result is necessary to prevent flooding the app screen with too many user icons, but it also limits an attacker's ability to follow a moving target. We find that this down sampling can be overcome by simply repeatedly querying the system until the target is found. We perform query measurements on four test areas (of different sizes between 3×4 mile2 and 24×32 mile2) in the downtown area of Los Angeles (City A, with 10 million residents as of 2015). For each area, we issue 400 queries within 10 seconds, and examine the number of unique users returned by all the queries. Results in Figure 4 show that the number of unique users reported converges after 150-250 queries for the three small search areas ($\leq 12 \times 16$ mile2). For the area of size 24×32 mile2, more than 400 queries are required to reach convergence.

D.TRACKING USERS OVER TIME

Our analysis found that each active Waze app updates its GPS coordinates to the server every 2 minutes, regardless of whether the user is mobile or stationary. Even when running in the background, the Waze app reports GPS values every 5 minutes. As long as the Waze app is open (even running in the background), the user's location is continuously reported to Waze and potential attackers. Clearly, a more conservative approach to managing location data would be helpful here. We note that attackers can perform long-term tracking on a target user (e.g., over months). The attacker needs a persistent ID associated to the target. The "user ID" field in the metadata is insufficient, because it is a random "session" ID assigned upon user login and is released when the user kills the app. However, the "account creation time" can serve as a persistent ID, because a) it remains the same across the user's different login sessions, and b) it is precise down to the second, and is sufficiently to uniquely identify single users in the same geographic area. While Waze can remove the "account creation time" field from metadata, a persistent attacker can overcome this by analyzing the victim's mobility pattern. For example, the attacker can identify a set of locations where the victim has visited frequently or stayed during the past session, mapping to home or workplace. Then the attacker can assign a ghost rider to constantly

monitor those areas, and reidentify the target once her icon shows up in a monitored location, e.g., home.

A graphical view of the tracking result in Los Angeles downtov
(City A). Blue dots are GPS points captured by the attacker an
the red dots are those missed by the attacker.

E.STEALTH MODE.

We note that attackers remain invisible to their targets, because queries on any specific geographic area can be done by Sybil's operating "remotely," i.e. claiming to be in a differment city, state or country. Attackers can enable their "invisible" option to hide from other nearby users. Finally, disabling these features still does not make the attacker visible. Waze only updates each user's "nearby" screen every 2 minutes (while sending its own GPS update to the servers). Thus a tracker can "pop into" the target's region, query for the target, and then move out of the target's observable range, all before the target can update and detect it.

F.REAL-TIME INDIVIDUAL USER TRACKING

To build a detailed trace of a target user's movements, an attacker first bootstraps by identifying the target's icon on the map. This can be done by identifying the target's icon while confirming her physical presence at a time and location. The attacker centers its search area on the victim's location, and issues a large number of queries (using Sybil accounts) until it captures the next GPS report from the target. If the target is moving, the attacker moves the search area along the target's direction of movement and repeats the process to get updates.

EXPERIMENTS

To evaluate its effectiveness, we performed experiments by tracking one of our own Android Smartphone's and one of our virtual devices. Tracking was effective in both cases, but we experimented more with tracking our virtual device, since we could have it travel to any location. Using the OSRM tool [28], we generate detailed GPS traces of two driving trips, one in downtown area of Los Angeles (City A), and one along the interstate highway-101 (Highway B). The target device uses a realistic driving speed based on average traffic speeds estimated by Google Maps during the experiment. The attacker used 20 virtual devices to query Waze simultaneously in a rectangular search area of size 6 × 8 mile2 . This should be sufficient to track the GPS update of a fast-driving car (up to 160 mph). Both experiments were during morning hours, and we logged both the network traffic of the target phone and query data retrieved by the attacker.

DEFENSES

In this section, we propose defense mechanisms to significantly limit the magnitude and impact of these attacks. While individual devices can inflict limited damage, an attacker's ability to control a large number of virtual vehicles at low cost elevates the severity of the attack in both quantity and quality. Our priority, then, is to restrict the number of ghost riders available to each attacker, thus increasing the cost per "vehicle" and reducing potential damage. The most intuitive approach is performing strong location authentication, so that attackers must use real devices physically located at the actual locations reported. This would make ghost riders as expensive to operate as real devices. Unfortunately, existing methods for location authentication do not extend well to our context. Some proposals solely rely on trusted infrastructures (e.g., wireless access points) to verify the physical presence of devices in close proximity [29], [30]. However, this requires large scale retrofitting of cellular cell towers or installation of new hardware, neither of which is practical at large geographic scales. Others propose to embed tamperproof location hardware on mobile devices [31], [32], which incurs high cost per user, and is only effective if enforced across all devices. For our purposes, we need a scalable approach that works with current hardware, without incurring costs on mobile users or the map service (Waze).

A. SYBIL DETECTION VIA PROXIMITY GRAPH

Instead of optimizing per-device location authentication, our proposed defense is a Sybil detection mechanism based on the novel concept of proximity graph. Specifically, we leverage physical proximity between real devices to create collocation edges, which act as secure attestations of shared physical presence. In a proximity graph, nodes are Waze devices (uniquely identified by an account username and password on the server side). They perform secure peer-to-peer location authentication with the Waze app running in the background. An edge is established if the proximity authentication is successful. Because Sybil devices are scripted software, they are highly unlikely to come into physical proximity with real devices. A Sybil device can only form collocation edges with other Sybil devices (with coordination by the attacker) or the attacker's own physical devices. The resulting graph should have only very few (or no) edges between virtual devices and real users (other than the attacker). Leveraging prior work on Sybil detection in social networks, groups of Sybil's can be characterized by the few "attack edges" connecting them to the rest of the graph, making them identifiable through community-detection algorithms [33]. We use a very small number of trusted nodes only to bootstrap trust in the graph. We assume a small number of infrastructure access points are known to Waze servers, e.g., hotels and public Wi-Fi networks associated with physical locations stored in IPlocation databases (used for Geolocation by Apple and Google). Any Waze device that communicates with the Waze server under their IPs (and reports a GPS location consistent with the IP) automatically creates a new collocation edge to the trusted node.

B. PEER-BASED PROXIMITY AUTHENTICATION

To build the proximity graph, we first need a reliable method to verify the physical collocation of mobile devices. We cannot rely on GPS reports since attackers can forge arbitrary GPS coordinates, or Bluetooth based device ranging [34] because the coverage is too short (Tracking via User Queries. A Waze client periodically requests updates in her nearby area, by issuing an update query with its GPS coordinates and a rectangular "search area." This search area can be set to any location on the map, and does not depend on the requester's own location. The server returns a list of users located in the area, including user ID, nickname, account creation time, GPS coordinates and the GPS timestamp. Thus an attacker can find and "follow" a target user by first locating them at any given location (work, home) and then continuously following them by issuing update queries centered on the target vehicle location, all automated by scripts. Overcoming Down sampling. The user query approach faces a down sampling challenge, because Waze responds to each query with an "incomplete" set of users, i.e., up to 20 users per query regardless of the search area size. This downsampled result is necessary to prevent flooding the app screen with too many user icons, but it also limits an attacker's ability to follow a moving target. We find that this down sampling can be overcome by simply repeatedly querying the system until the target is found. We perform query measurements on four test areas (of different sizes between 3×4 mile2 and 24×32 mile2) in the downtown area of Los Angeles (City A, with 10 million residents as of 2015). For each area, we issue 400 queries within 10 seconds, and examine the number of unique users returned by all the queries. Results show that the number of unique users reported converges after 150-250 queries for the three small search areas ($\leq 12 \times 16$ mile2). For the area of size 24×32 mile2 , more than 400 queries are required to reach convergence. [6] discussed about a method, In vehicular ad hoc networks (VANETs), because of the nonexistence of end-to-end connections, it is essential that nodes take advantage of connection opportunities to forward messages to make end-to-end messaging possible. [8] discussed about a system, In this proposal, a neural network approach is proposed for energy conservation routing in a wireless sensor network. Our designed neural network system has been successfully applied to our scheme of energy conservation. Neural network is applied to predict Most Significant Node and selecting the Group Head amongst the association of sensor nodes in the network.

C. TRACKING USERS OVER TIME

Our analysis found that each active Waze app updates its GPS coordinates to the server every 2

minutes, regardless of whether the user is mobile or stationary. Even when running in the background, the Waze app reports GPS values every 5 minutes. As long as the Waze app is open (even running in the background), the user's location is continuously reported to Waze and potential attackers. Clearly, a more conservative approach to managing location data would be helpful here. We note that attackers can perform long-term tracking on a target user (e.g., over months). The attacker needs a persistent ID associated to the target. The "user ID" field in the metadata is insufficient, because it is a random "session" ID assigned upon user login and is released when the user kills the app. However, the "account creation time" can serve as a persistent ID, because a) it remains the same across the user's different login sessions, and b) it is precise down to the second, and is sufficiently to uniquely identify single users in the same geographic area. While Waze can remove the "account creation time" field from metadata, a persistent attacker can overcome this by analyzing the victim's mobility pattern. For example, the attacker can identify a set of locations where the victim has visited frequently or stayed during the past session, mapping to home or workplace. Then the attacker can assign a ghost rider to constantly monitor those areas, and reidentify the target once her icon shows up in a monitored location, e.g., home. Stealth Mode. We note that attackers remain invisible to their targets, because queries on any specific geographic area can be done by Sybil's operating "remotely," i.e. claiming to be in a different city, state or country. Attackers can enable their "invisible" option to hide from other nearby users. Finally, disabling these features still does not make the attacker visible. Waze only updates each user's "nearby" screen every 2 minutes (while sending its own GPS update to the servers). Thus a tracker can "pop into" the target's region, query for the target, and then move out of the target's observable range, all before the target can update and detect it.

PEER-BASED PROXIMITY AUTHENTICATION

To build the proximity graph, we first need a reliable method to verify the physical collocation of mobile devices. We cannot rely on GPS reports since attackers can forge arbitrary GPS coordinates, or Bluetooth based device ranging [34] because the coverage is too short (<10 meters) for vehicles. Instead, we conside a challenge-based proximity authentication method, which leverages the limited transmission range of Wi-Fi radios.

- Wi-Fi Tethering Challenge.
- Constructing Proximity Graphs.

GRAPH-BASED SYBIL DETECTION

We apply graph-based Sybil detection algorithms to detect Sybil's in Waze proximity graph. Graph-based Sybil detectors [33], [36]– [42] were originally proposed in social networks. They all rely on the key assumption that Sybil's have difficulty to form edges with real users, which results in a sparse cut between the Sybil and nonSybil regions in the social graph. Because of the limited number of "attack edges" between Sybil's and non-Sybil's, a random walk from non-Sybil region has a higher landing probability to land on a non-Sybil node than a Sybil node. Although this assumption may not always hold in online social networks [43], it holds well for the proximity graph. In online social networks, Sybil's may build "attack edges" by befriending with real users (e.g., using attractive female photos) [43]. However, in a proximity graph, building an attack edge requires physical collocations. With the Wi-Fi authentication, it's difficult to build attack edges using software simulations alone in a massive, automated manner (e.g., for tens of thousands of Sybil devices). In addition, the authentication is done in the background without human involvement, which further eliminates the chance for Sybil's to trick real users to add edges.

COUNTERMEASURE EVALUATION

We use simulations to evaluate the effectiveness of our proposed defense. We focus on evaluating the feasibility and cost for attackerrs to maintain a large number of Sybil's after the Sybil detection is in place. We quantify the cost by the number of attack edges a Sybil must establish with real users. In practice, this translates into the effort taken to physically drive around and use physical devices (with Wi-Fi radios) per Sybil to complete proximity authentication. In the following, we first describe our

simulation setup, and then present the key findings and their implications on Waze.

II.RESULTS

Our evaluation primarily focuses on Sybil Rank, and we briefly discuss the results of Sybil SCAR in the end.

ACCURACY OF SYBIL DETECTION

We assume the attacker seeks to embed 1000 Sybil's into the proximity graph. We use either single- or multi-gateway approaches to build attack edges on the proximity graph by connecting Sybil's to randomly chosen normal users. We then add edges between Sybil nodes, following the power-law distribution and producing an average weighted degree of either 5 or 10 (to emulate different Sybil sub graph density). We randomly select 10 trusted nodes to bootstrap trust for Sybil Rank and run it on the proximity graph. We repeat each experiment 50 times.

BROADER IMPLICATIONS

While our experiments and defenses have focused strictly on Waze, our results are applicable to a wider range of mobile applications that rely on geolocation for user-contributed content and metadata. Examples include location based check-in services (Foursquare, Yelp), mobile navigation systems (Waze, Moovit), crowd sourced taxi services (Uber, Lyft), mobile dating apps (Tinder, Bumble), anonymous mobile communities (Yik Yak, Whisper) and location-based gaming apps (Pokemon Go). These systems face two common challenges exposing them to potential attacks. First, our efforts show that it is difficult for app developers to build a truly secure channel between the app and the server. There are numerous avenues for an attacker to reverseengineer and mimic an app's API calls, thereby creating "cheap" virtual devices and launching Sybil attack . [4] discussed about a system,the effective incentive scheme is proposed to stimulate the forwarding cooperation of nodes in VANETs. In a coalitional game model, every relevant node cooperates in forwarding messages as required by the routing protocol. This scheme is extended with constrained storage space. A lightweight approach is also proposed to stimulate the cooperation..

RELATED WORK

SECURITY IN LOCATION-BASED SERVICES

Location-based services face various threats, ranging from rogue users reporting fake GPS, [53], to malicious parties compromising user privacy [54]. A related study on Waze [55] demonstrated that small-scale attacks can create traffic jams or track user icons, with up to 15 mobile emulators. Our work differs in two key aspects. First, we show that it's possible to reverse engineer its APIs, enabling light-weight Sybil devices (simple scripts) to replace full-stack emulators. This increase the scale of potential attacks by orders of magnitude, to thousands of Waze clients per commodity laptop. The impact of thousands of virtual vehicles is qualitatively different from 10-15 mobile simulators. Second, as possible defenses, [2] discussed about creating Obstacles to Screened networks. In today's technological world, millions of individuals are subject to privacy threats. Companies are hired not only to watch what you visit online, but to infiltrate the information and send advertising based on your browsing history. In contrast, we propose a novel proximity graph approach to detect and constrain the impact of virtual devices. Researchers have proposed to preserve user location privacy against map services such as Waze and Google. Earlier studies apply location cloaking by adding noise to the GPS reports [56]. Recent work use zero-knowledge [57] and differential privacy [58] to preserve the location privacy of individual users. Our work differs by focusing on the attacks against the map services

III.CONCLUSION

We describe our efforts to identify and study a range of attacks on crowd sourced map services. We identify a range of single and multi-user attacks, and describe techniques to build and control groups of virtual vehicles (ghost riders) to amplify these attacks. Our work shows that today's mapping services are highly vulnerable to software agents controlled by malicious users, and both the stability of these services and the privacy of millions of users are at stake. While our study and experiments focus

on the Waze system, we believe the large majority of our results can be generalized to crowd sourced apps as a group. We propose and validate a suite of techniques that help services build proximity graphs and use them to effectively detect Sybil devices. Throughout this work, we have taken active steps to isolate our experiments and prevent any negative consequence on real Waze users. We also proactively informed Waze team of theses attacks, and worked with them to mitigate the threat.

REFERENCE

[1] N. Stefano itch, A. Alshamsi, M. Cebrian, and I. Rahwan, "Error and attack tolerance of collective problem solving: The darpa shredder challenge," EPJ Data Science, vol. 3, no. 1, pp. 1–27, 2014.

[2] Christo Ananth, P.Muppidathi, S.Muthuselvi, P.Mathumitha, M.Mohaideen Fathima, M.Muthulakshmi, "Creating Obstacles to Screened networks", International Journal of Advanced Research in Biology, Ecology, Science and Technology (IJARBEST), Volume 1,Issue 4,July 2015, pp:10-14.

[3] Z. Zhang, L. Zhou, X. Zhao, G. Wang, Y. Su, M. Metzger, H. Zheng, and B. Y. Zhao, "On the validity of geosocial mobility traces," in Proc.of HotNets, 2013.

[4] Christo Ananth, M.Muthamil Jothi, A.Nancy, V.Manjula, R.Muthu Veni, S.Kavya, "Efficient message forwarding in MANETs", International Journal of Advanced Research in Management, Architecture, Technology and Engineering (IJARMATE), Volume 1,Issue 1, August 2015,pp:6-9.

[5] S. Cheng, "Uber's terrifying "ghost drivers" are freaking out passengers in china," Quartz, September 2016.

[6] Christo Ananth, Kavya.S., Karthika.K., Lakshmi Priya.G., Mary Varsha Peter, Priya.M., "CGT Method of Message forwarding", International Journal of Advanced Research in Management, Architecture, Technology and Engineering (IJARMATE), Volume 1, Issue 1, August 2015,pp:10-15.

[7] M. Wehner, "How to cheat at Pokémon go and catch any Pokémon you want without leaving your couch," Daily Dot, July 2016.

[8] Christo Ananth, A.Nasrin Banu, M.Manju, S.Nilofer, S.Mageshwari, A.Peratchi Selvi, "Efficient Energy Management Routing in WSN", International Journal of Advanced Research in Management, Architecture, Technology and Engineering (IJARMATE), Volume 1, Issue 1, August 2015,pp:16-19.

[9] V. Goel, "Maps that live and breathe with data," The New York Times, June 2013.

[10] Google, "Google maps and Waze, outsmarting traffic together," Google Official Blog, June 2013.

[11]"GenyMotionEmulator,"http://www.genymotion.com.

[12]"Monkeyrunner,"https://developer.android.com/studio/test/monkeyrunner/index.html.

[13] B. Reed, "Google maps becomes Google's second 1 billion-download hit," Yahoo! News, June 2014.

[14] "Charles Proxy," http://www.charlesproxy.com.

[15] D. Sounthiraraj, J. Sahs, G. Greenwood, Z. Lin, and L. Khan, "Smvhunter: Large scale, automated detection of ssl/tls man-in-the-middle vulnerabilities in android apps," in Proc. of NDSS, 2014.