International Conference on Advances in Materials, Computing and Communication Technologies
In Association with International Journal of Scientific Research in Science and Technology
Volume 9 | Issue 1 | Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)

Tracking Car Location Using GPS-Agnostic Plate Number Recognition

Angel Jeba Rathna. S[1], Subbulakshmi .T.C[2]
[1]PG Scholar, Department of IT, Francis Xavier Engineering College, Tirunelveli, Tamil Nadu, India
[2]Assistant Professor, Department of IT, Francis Xavier Engineering College, Tirunelveli, Tamil Nadu, India

ABSTRACT

Smart phones nowadays are equipped with GPS chips to enable navigation and location-based services. A malicious app with the access to GPS data can easily track the person who carries the smart phone. People may disable the GPS module and turn it on only when necessary to protect their location privacy. However, in this paper, we demonstrate that an attacker is still able to track a person by using the embedded magnetometer sensor in victim's smart phone, even when the GPS module is disabled all the time. Moreover, this attack neither requests user permissions related to locations for installation, nor does its operation rely on wireless signals like network positioning or suffer from signal propagation loss. Only the angles of a car's turning measured by the magnetometer sensor of a driver's smart phone are utilized. The results show that it is possible for attacker to precisely pinpoint the actual path when the driving path includes 11 turns or more. More simulations are performed to demonstrate the attack with lager selected local areas.

I. INTRODUCTION

Global Positioning Systems (GPS) are widely used in military and civilian applications for navigation and localization. Advancements in mobile technologies further allow GPS to be integrated into mobile devices to provide various Locations based services (LBS). Indeed, more and more emerging smart phone apps like Instagram, Uber, Google Drive, WhatsApp, etc. can access GPS data, and hence concerns have been raised regarding location privacy. Malicious apps with access to the GPS can easily track geographic movement traces of a person, and consequently discover sensitive personal information like home and work addresses, shopping preference, health conditions. To avoid being tracked, one intuitive method is to disable the GPS module on smart phones.

II. MAGETOMETER SENSOR

Magnetometershave extensive applications, mainly in directional path finding for all modes of transportation and the discovery of mineral deposits and other items of interest for mining. As such, they are a mature technology, having seen development in theory and operation. In tablets and smartphones, the magnetometer is a solid state. Hall-effect sensor which detects the Earth's magnetic field and produces a voltage proportional to the strength and polarity of this field along the X, Y, and Z axes of the sensor. This produced voltage is then converted to a digital

signal representing the magnetic field intensity. This attack involves obtaining compass readings via an installed smartphone application, which requires no granted permissions and cannot be disabled. As an example, Figure3 contains raw data from the magnetometer during a sample driving trip. It is clear from observation where directional changes take place within this data, and we discuss generalized methods to isolate turning and other events in Section V. To solve this problem, the attacker can build its own database by taking advantage of an existing OSM project OpenStreetMap (OSM) [3], whose entire database can be downloaded for free. However, an important practical hurdle is that the database of the OSM project is huge (more than 40GB) and it will be very expensive to store a full database. The attacker must remove unnecessary information from the original database for efficiency. Moreover, the OSM database does not provide road intersection angles either, and the attacker has to find a way to calculate these angles and identify and redesign database relations to minimize the storage cost. Due to the limited storage capability of a smart phone, the database should be installed on the attacker's server instead of a smart phone. The malicious app sends collected compass measurements to the attacker whenever Internet connections are available. Again, this process can be camouflaged by normal activities like game score reporting, music rating, and text message exchanges. Lastly, the attacker needs to design an efficient matching algorithm that compares car turn angles with road intersection angles to find the path traversed by a target driver. Assume that n turn angles are identified and the N turn angle can be matched to Ii similar road intersection angles on a real world map. The total number of potential paths is therefore I1 × I2 × • • • × In. Because an individual turn angle. In contrast, the intuitive choice of Google Maps is not suitable for this attack due to its closed source nature and the resultant inability to obtain the actual map data points. As part of the accessibility of OSM, portions of the global database may be downloaded that comprise sections of Earth of any size. Upon download, OSM data appears in the form of a topological data structure having four core elements (data primitives): Nodes are points with a geographic position, stored as latitude and longitude coordinate pairs. These may represent locations of interest such as mountain peaks or other landmarks, but their more common and pertinent use is as constituents of larger objects. Ways are ordered lists of nodes comprising a polyline or a closed loop polygon. Polyline ways define linear features such as streets and rivers, and polygons delineate areas like forests, parks, parking areas and lakes. Relations are ordered lists which can include nodes, ways, and relations (together called "members"), where each member can optionally have a string-defined "role." Relations are used to explain any involvement between nodes and/or ways, such as turn restrictions on roads, routes that span several existing ways and areas with holes. Tags are key- value pairs (both arbitrary strings) storing metadata about the map objects such as name, type, or physical properties.

III. EXISTING SYSTEM

Nodes are points with a geographic position, stored as latitude and longitude coordinate pairs. Ways are ordered lists of nodes comprising a polyline or a closed loop polygon. Relations are used to explain any involvement between nodes and/or ways, such as turn restrictions on roads, routes that span several existing ways (for instance, a long-distance motorway), and areas with holes. Tags are key-value pairs (both arbitrary strings) storing metadata about the map objects such as name, type, or physical properties. The four events useful to this attack are turns, lane changes, curved roads, and stop times

IV. PROPOSED SYSTEM

Spoof GNSS on Vehicle: GNSS signals could be spoofed to assist a thief stealing a vehicle. Modify Map

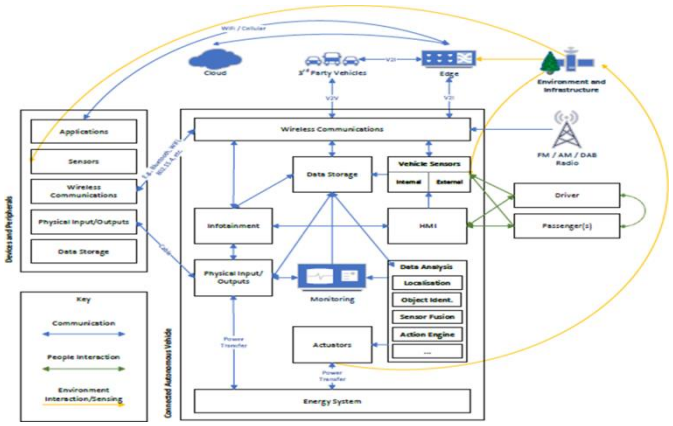
via Update on Cloud: A map update is used to force the vehicle along a route to an arbitrary destination. Replay Retrieval on Device: A thief replays a recorded signal used to retrieve the vehicle. Blind Range Sensor on Vehicle: An adversary seeking to cause a crash could blind the range sensor to prevent a vehicle from knowing its distance from obstacles. DoS Parking Allocator on Cloud: An adversary seeking to induce a traffic jam or freeze the parking garage could DoS the allocator, preventing vehicles from requesting new spaces.

V. PYTHON TECHNOLOGY

Python is an interpreted, high-level, object oriented programming language. Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. Python is a multi-paradigm programming language. They are fully supported, and many of its features support programming and (including by meta programming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution. Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution. It is a logical programming and collector for memory management Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.). Python has a simple syntax similar to the English language. Python has syntax that allows developers to write programs with fewer lines than some other programming languages. Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick. Python can be treated in a procedural way, an object- orientated way or a functional way. Some Python expressions are similar to languages such as C and Java Addition, subtraction, and multiplication are the same, but the Behaviour of division differs. There are two types of divisions in Python. They are floor division and integer division. Python also added the ** operator for exponentiation. From Python 3.5, the new @ infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication. In Python, compares by value, versus Java.. Anonymous functions are implemented using lambda expressions; however, these are limited in that the body can only be one expression. Conditional expressions in Python are written as x if c else y (different in order of operands from the c ? x : y operator common to many other languages).Python makes a distinction between lists and tuples. Lists are written as [1, 2, 3], are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as (1, 2, 3), are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The + operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. Thus, given the variable t initially equal to (1, 2, 3), executing t = t + (4, 5) first evaluates t + (4, 5), which yields (1, 2, 3, 4, 5), which is then assigned back to t, thereby effectively "modifying the contents" of t, while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts. Python features sequence unpacking where multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are

associated in the identical manner to that forming tuple literals and, as a whole, are put on the side of the equal sign in an assignment statement.

VI. ARCHITECTURALDIAGRAM



OPEN STREET MAP

The OSM project [32] was initiated in 2004 as, among other things, open source alternative to such services as Google Maps [3]. Its central database and web services are hosted at the University College London campus, and the servers and interfaces built to create and share OSM data are largely developed and administered by volunteers. Additionally, the OSM data itself is collected by volunteers worldwide, and likewise accessible in an open source manner for all interested. In contrast, the intuitive choice of Google Maps is not suitable for this attack due to its closed source nature and the resultant inability to obtain the actual map data points. As part of the accessibility of OSM, portions of the global database may be downloaded that comprise sections of Earth of any size. Upon download, OSM data appears in the form of a topological data structure having four core elements (data primitives):

- Nodes are points with a geographic position, stored as latitude and longitude coordinate pairs. These may represent locations of interest such as mountain peaks or other landmarks, but their more common and pertinent use is as constituents of larger objects. • Ways are ordered lists of nodes comprising a polyline or a closed loop polygon. Polyline ways define linear features such as streets and rivers, and polygons delineate areas like forests, parks, parking areas and lakes.
- Relations are ordered lists which can include nodes, ways, and relations (together called "members"), where each member can optionally have a string- defined "role." Relations are used to explain any involvement between nodes and/or ways, such as turn restrictions on roads, routes that span several existing ways (for instance, a long-distance motorway), and areas with holes. • Tags are key-value pairs (both arbitrary strings) storing metadata about the map objects such as name, type, or physical properties. Tags are not free-standing, but are always attached to a node, way, or relation. Tags are written in OSM documentation as key = value, where key selects from a broad class of features, and value describes that feature.

ROUTE MATCHING

As previously mentioned, the attacker in this scenario wishes to determine which intersections have been traversed by the driver. To start, the attacker queries the database for all intersection turn angles which match the turns made by the driver, assembling a list of angles for each turn. Next, any connections between adjacent turns are identified, providing several candidate routes. Finally, limiting conditions,

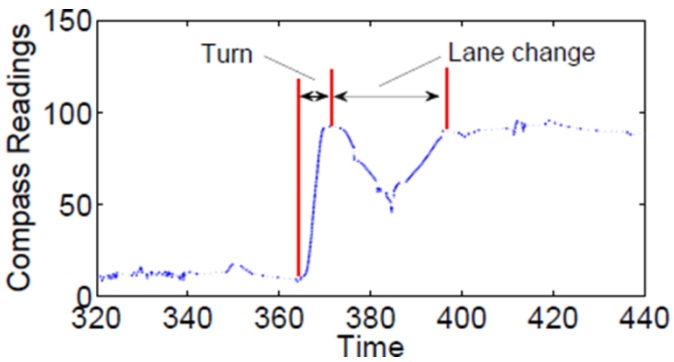


Figure 3.11 Difference between a turn and a lane changing

turning directions and speed limits, are applied to filter out logically impossible routes to a small number of resulting possible paths (ideally just one).

TABLE 3.1

THE MATRIX OF THE ROUTES.

Routes \ Turns	T_1	T_2	...	
1	$A_{1,1}$	$A_{1,2}$	...	
2	$A_{2,1}$	$A_{2,2}$	...	
...	...	...	...	
j	$A_{j,1}$	$A_{j,2}$	...	
...	...	...	...	

VII. COUNTERMEASURES DISCUSSION

The nature of this side-channel attack attracts several trivial solutions but only one that is robust. Unfortunately, it is also the most logistically difficult as it requires fundamental modifications to Android and iOS. Specifically, motion and orientation sensors which as previously discussed are not bound to permissions should be, so that users have the ability to know an app is using them. This will mesh well with the recent permissions upgrade found in Android 6.0, which enables users to choose for each app among the permissions required by the app developer and requires the developer to

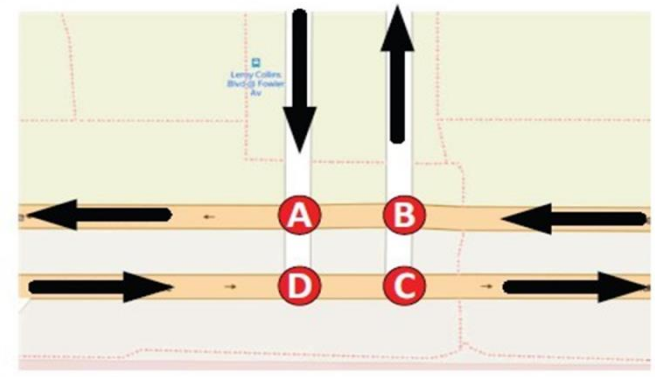


Figure 13 Four nodes in an intersection

handle being denied each such permission. Then, if this attack is hidden in, say, a media player app, and the user does not know of any reason that the magnetometer sensor would be focal to the operation of that app, the user can disallow that permission and thereby protect against information leakage.

This is ultimately contingent upon the inclusion of sensor permissions in future versions of Android (and all of these updates in iOS), however, and furthermore that alone is not sufficient for a comprehensive solution. It will simply limit the attack space to include apps which have a good reason for using orientation or motion sensors, as well as a data connection. The obvious example here is games, which are both multitudinous and often ignored by users in terms of security. The definitive distinction for preventing this type of side- channel attack is therefore that for car safety users should not access their phones while driving, so the apps in question should not be active. Rather, they will be in the background, if they are running at all, and so this prompts a separation within Android OS of background permissions and foreground permissions. To protect against this attack, users could simply disable use of sensors and data while the app is in the background, leaving them available when the app is in the foreground so it remains usable then. While driving, the phone will be left in standby as required for safety, and the data necessary for this attack will not be retrievable.

The ability to choose different permissions based on whether or not the app is being actively used will afford users powerful control over their app security. Far from simply preventing this attack, it can prevent other sorts of side-channel attacks dependent on collecting data by means invisible to the user, by giving the user the option to restrict the abilities of apps outside of active use. This functionality can benefit users in many scenarios other than in response to the attack detailed in this research. Some individuals may enjoy informing their social media contacts that they are “checking in” at a special location, but an app providing this capability needs not continue accessing location in the background between uses, for example. This distinction between foreground and background processes belongs loosely

to the concept of context-aware security solutions, but is coarse enough to be usable and understandable by the general public.

In the absence of sensor permissions and a foreground/ background permission distinction, maintaining location privacy in the presence of this attack is clumsy at best. The user can place the phone in a case with a magnetic closure mechanism on its lid, which will provide enough magnetic interference to overpower the small amount of magnetism in Earth's magnetic field, and thereby disrupt the magnetometer into providing false readings. Further, a case constructed of ferromagnetic materials will have a similarly deleterious effect on the magnetometer. These options impair usability outside of the driving environment, however, and mass-produced metallic phone cases are predominantly aluminum, a non-ferromagnetic metal. Otherwise, a user may move the phone at some point during the drive, violating our assumption regarding relative immobility of the phone and confusing the attack, but this is hardly a robust defense.

VIII. REFERENCES

[1]. Dan Song; Ratnasingham Tharmarasa; Thiagalingam Kirubarajan; Xavier N. Fernando,2018, "Multi-Vehicle Tracking With Road Maps and Car-Following Models", IEEE Transactions on Intelligent Transportation Systems, vol: 19, no: 5,pp. 1375 – 1386.

[2]. Alessandro Rucco; Giuseppe Notarstefano; John Hauser,2015, "An Efficient Minimum-Time Trajectory Generation Strategy for Two-Track Car Vehicles", IEEE Transactions on Control Systems Technology, vol: 23, no: 4,pp. 1505 – 1519.

[3]. Dongchul Kim;Kichun Jo;Minchul Lee; Myoungho Sunwoo,2018, "L-Shape Model Switching-Based Precise Motion Tracking of Moving Vehicles Using Laser Scanners", "IEEE Transactions on Intelligent Transportation Systems", vol: 19, no: 2,pp. 598 – 612.

[4]. Chih-Lyang Hwang,2016, "Comparison of Path Tracking Control of a Car-Like Mobile Robot With and Without Motor Dynamics", IEEE/ASME Transactions on Mechatronics,vol: 21, no: 4,pp. 1801 –1811.

[5]. Keyvan Majd;Mohammad Razeghi-Jahromi;Abdollah Homaifar,2020, "A stable analytical solution method for car- like robot trajectory tracking and optimization", IEEE/CAA Journal of Automatica Sinica, vol: 7, no: 1,pp. 39 – 47.

[6]. Abdullahi Daniyan; Sangarapillai Lambotharan;Anastasios Deligiannis;Yu Gong; Wen-Hua Chen,2018, "Bayesian Multiple Extended Target Tracking Using Labeled Random Finite Sets and Splines", IEEE Transactions on Signal Processing, vol: 66, no: 22,pp. 6076 – 6091.

[7]. Kunpeng Dai;Yafei Wang;Qinghui Ji;Haiping Du;Chengliang Yin,2019, "Multiple Vehicle Tracking Based on Labeled Multiple Bernoulli Filter Using Pre-Clustered Laser Range Finder Data", IEEE Transactions on Vehicular Technology, vol: 68, no: 11,pp. 10382 – 10393.

[8]. Sangjin Lee;James McBride,2019, "Extended Object Tracking via Positive and Negative Information Fusion", IEEE Transactions on Signal Processing, vol: 67, no: 7,pp. 1812 – 1823.

[9]. Zi Li;Qingqi Pei;Ian Markwood;Yao Liu;Miao Pan;Hongning Li,2018, "Location Privacy Violation via GPS- Agnostic Smart Phone Car Tracking", IEEE Transactions on Vehicular Technology, vol: 67, no: 6,pp. 5042 – 5053.

[10]. Shengping Zhang;Yuankai Qi;Feng Jiang;Xiangyuan Lan;Pong C. Yuen;Huiyu Zhou,2018, "Point-to-Set Distance Metric Learning on Deep Representations for Visual Tracking", IEEE Transactions on Intelligent Transportation Systems, vol: 19, no: 1,pp. 187 – 198.

[11]. Jingjing Chen;Yi Ruan;Lingling Guo;Huijuan Lu,2020, "BCVehis: A Blockchain-Based Service Prototype of Vehicle History Tracking for Used-Car Trades in China", IEEE Access, vol: 8,pp. 214842 – 214851.

[12]. Jingwei Cao;Chuanxue Song;Silun Peng;Shixin Song;Xu Zhang;Feng Xiao,2020, "Trajectory Tracking Control Algorithm for Autonomous Vehicle Considering Cornering Characteristics", IEEE Access, vol: 8,pp. 59470 – 59484.

[13]. Xing Sun;Nelson H. C. Yung;Edmund Y. Lam,2016, "Unsupervised Tracking With the Doubly Stochastic Dirichlet Process Mixture Model", IEEE Transactions on Intelligent Transportation Systems, vol: 17, no: 9,pp. 2594 – 2599.

[14]. Gültekin Gündüz ; Tankut Acarman,2019, "Efficient Multi-Object Tracking by Strong Associations on Temporal Window", IEEE Transactions on Intelligent Vehicles, vol: 4, no: 3,pp. 447 – 455.

[15]. Dan Song;Ratnasingham Tharmarasa;Gongjian Zhou;Mihai C. Florea;Nicolas Duclos-Hindie;Thiagalingam Kirubarajan,2019, "Multi-Vehicle Tracking Using Microscopic Traffic Models", IEEE Transactions on Intelligent Transportation Systems, vol: 20, no: 1,pp. 149 – 161.