A Forensic Analysis of Micromobility Solutions

Jan-Niclas Hilgert, Martin Lambertz, Alina Hakoupian, Anna-Mariya Mateyna

Fraunhofer FKIE, Zanderstr. 5, 53177 Bonn, Germany

Abstract

Over the last years, various kinds of micromobility solutions including bikes and escooters have popped up in many cities across the globe. They provide a new, convenient form of transportation requiring only an easy-to-use application for the renting process, thus making micromobility solutions available to a wide user base. Due to this fact, investigators will positively encounter these applications during the analysis of a suspect's mobile device in the future. Considering their area of application, these apps possibly contain valuable information about a suspect's movements.

This paper aims to shed a light on the question, what exact kind of data these applications store and how it can be accessed during an analysis. We come to the conclusion that all of the micromobility providers we examined, store crucial data on their servers including personal user data, payment information as well as a ride history. In favor of assistance during a forensic analysis of micromobility solutions, we implemented a forensic toolkit for the acquisition of this data making use of the application-specific APIs of the providers. Furthermore, we go a step further and investigate how reliable the acquired movement data is and show whether and how it is possible to create completely spoofed trips.

Keywords:

1. Introduction

2020 offers many choices when it comes to getting from A to B. Taking the car, using public transport, riding the bike or simply walking, are options people could choose from for a long time. Over the last two years, a new possibility of locomotion has been added. Micromobility providers have started their operation of escooters and bikes across major cities all around the world providing users with easy access to this new mode of transportation.

Though the first motorized scooter or 'Autoped' was invented more than a century ago, today's digital environment including the use of applications has opened these vehicles up to the mass. Renting and using micromobility vehicles is simple, does not require any additional license and due to multiple companies in the market, there is usually no lack of vehicles in major cities.

Due to the fact that they can be used by anyone at any time, their applications will also likely be encountered during forensic analyses of suspects' phones in the future. Furthermore, they provide crucial information, namely about the movement of a suspect, including his or her frequently visited spots and routes, which makes it essential to understand what kind of data can be expected during an analysis, but also where and how it can be acquired.

While there has been research regarding security and privacy aspects of micromobility solutions (Vinayaga-Sureshkanth et al., 2020), to the best of our knowledge, there has been no prior work considering the forensic implications of these platforms. In this paper, we close this gap and take a closer look at micromobility applications from a forensic point of view. For this purpose, we are describing the different kinds of data being relevant during a forensic investigation as well as different possible sources of information including the local device storage, data accessible via provided APIs and also at all of the data stored by providers of such applications.

2. Data Sources and Forensic Considerations

In this chapter, we present different kinds of forensic considerations during the analysis of micromobility applications. This includes a categorization of the most crucial data during an analysis as well as other additional information provided by the applications that can be used for forensic or other investigations. Prior to that, we introduce the three different data acquisition categories used in our evaluation.

- Local data stored on the user's phone.
- Data stored within the cloud of the application provider and made available to the user via APIs.
- All information stored by the provider.

Email addresses: `jan-niclas.hilgert@fkie.fraunhofer.de` (Jan-Niclas Hilgert), `martin.lambertz@fkie.fraunhofer.de` (Martin Lambertz), `alina.hakoupian@gmx.de` (Alina Hakoupian), `anna-mariya.mateyna@fkie.fraunhofer.de` (Anna-Mariya Mateyna)

2.1. Data Sources

All of the three aforementioned sources of data have been evaluated by us and are interpreted with regard to the data they contain and provide. The following sections describe the general approach of how this data is obtained.

2.1.1. Local data

Depending on an application's design, its data can be stored locally on the device itself as well as remotely in the cloud of the service provider. Analyzing files stored on the device requires access to the file system, a backup including the data or any other possibility for exporting the data from the application. For this reason, this source is only available when certain requirements are met (e.g. a device with root access). Our evaluation considers only on application-specific data and does not take into account operating system specific or analytics data such as caches or crash logs. Moreover, we focus on iOS as an example for local data in this paper. For Android, we only list where to find the data required to access the cloud data source introduced in the following section. The complete results of our analysis of local Android data is publicly available in the repository associated with our paper (Fraunhofer FKIE, 2020).

2.1.2. Cloud data

Storing data in the cloud serves various purposes. Data which is frequently updated, such as scooter locations or valid parking zones, is usually requested from the server on demand. For personal data, storing it in the cloud benefits from giving the user access to his or her data from any device. For this purpose, some means of authentication against the server, e.g. by using a token, is or should be required. Thus, obtaining the necessary information for authentication opens up another source of information. Depending on the type of authentication used, this approach may also work when only limited access to the data of a device is given, e.g. by obtaining the SIM card in order to re-register a user.

Authentication. In order to facilitate access to cloud data, it is essential to ascertain the way an application authenticates its users as well as the steps necessary to impersonate a user. For example, authentication can be performed by utilizing tokens stored permanently on a user's device. Depending on their validity, it can be sufficient to acquire the token in order to access all of the relevant data during an investigation. On the other hand, authentication can also be performed by a classic user name and password combination. In these cases, acquiring the device may not be enough, since it is not given that the password is stored on the device itself. In case of limited access to a device's data or an invalid token, it may also be possible to create new tokens for a user. This can for example be done by rerunning the registration process. It is, therefore, also important to assess the ways of generating new authentication means for a user, e.g. new tokens. Since authentication is application-specific, the corresponding chapters will provide more details about the authentication of a certain application.

2.1.3. Complete Data

Providers usually store more data about their users than users are given access to via applications or APIs. Since this data may be of importance for a forensic investigator, our plan was to take a closer look at the complete data stored by providers and compare it to data obtained by the other two means of acquisition. For our research, we wanted to make use of the General Data Protection Regulation (GDPR) established by the European Union in 2018 (European Parliament, Council of the European Union, 2016). The GDPR ensures rules regarding the handling of personal data and also gives users the possibility to request all of the data stored about them from companies. At the time of publishing we have received responses to our inquiries from all four providers. The data received, however, was unfortunately in no way complete. This is obvious, since our API requests return more user-related data than the information provided in their GDPR responses. Even asking them urgently to provide us with any remaining data did not yield sufficient results, which may be caused by legal limbos or other problems in the GDPR process. Since the received GDPR data was only a subset of the information accessible via the API, we have decided to exclude this data source from the evaluation in this paper. For some providers, the only additional information were events tracked by analytics frameworks (e.g. user triggers an event), which are out of the scope of this paper. However, since these frameworks collect a lot of data about user behavior and interaction, they may be an interesting data source for future research in this area.

2.2. Forensic Considerations

The following chapters describe what kind of data can be expected from the aforementioned types of data sources.

2.2.1. User Data

User data includes all of the information related to the user himself. This can be personal data such as the first or given name, phone number or email address as well as application-specific information like user names or credits. This type of data can be used as an indicator, whether all of the other information obtained during the analysis can be linked to a certain suspect.

2.2.2. Payment data

Since providers demand a valid payment method in order to use their services, this kind of data is always existent. For a forensic examiner, credit card details for instance may be linked to a real suspect more reliable than a user-provided first and last name. While payment processes are often handled by third-party providers, it becomes necessary to evaluate how this critical kind of data can be obtained.

2.2.3. Ride History

One of the most meaningful type of data of micromobility applications is the history of a user's rides. Usually a ride refers to a complete rental of a vehicle. The main focus of interest lies on temporal as well as geo information about a certain ride. Temporal information may include start and end time, while geo data contains for example the start and endpoint or even information about the whole track. Together, this provides an investigator with valuable information about the locations a user visited as well as the corresponding points in time. Additional information could include the length and duration of a ride as well as the average speed.

2.2.4. Rental

The process of renting a scooter involves multiple steps, which can be of interest during a forensic analysis:

1. During the first step, the user has to get an overview of the available scooters nearby. On local storage, this data may provide information about the time an application has been used as well as the location a user was interested in or located at.
2. As a second step, a scooter has to be unlocked in order to perform a ride. In some cases, this step can be performed without any physical interaction with the vehicle, thus making it possible to start rides from anywhere corrupting the ride history. In other cases, it may be required to scan some sort of code to prove physical presence near the vehicle.
3. During a ride, data such as speed or location may be recorded and stored on the device or external servers.
4. Finishing a ride can also require physical presence in some cases. Also, applications may provide the possibility to upload pictures of a parked vehicle.

Finally, it is also important to assess the reliability of the data that can be obtained about a suspect's movements. This includes answering the question, whether it is possible to create rides with false evidence as well how easily this can be achieved.

These forensic considerations will now be applied in the following chapters to gain a detailed insight into the data stored by major micromobility providers, which were chosen due to their availability in our region.

3. Lime

Lime, founded in early 2017, offers electronic scooters and bikes in more than 100 cities across the world (Lime, 2020). Their vehicles are identified by a 6 character, alphanumerical identifier referred to as plate number. However, the identifier does not necessarily correspond to an existing license plate.

Registration and Authentication. For the registration with Lime, two options are available. The first one, which has been the default since its launch, makes use of a mobile phone number used to receive a six digit login code. The second and recently added method provides the possibility of using an email address to receive a 24 character alphanumerical magic link token. After a successful registration, both options yield a bearer token, used for the subsequent authentication of a user against Lime's servers. As soon as a new bearer token is created by re-registering a user, all previous tokens become invalid and cannot be used for further authentication. Thus, access to the suspect's phone number or the most recent bearer token is required in order to authenticate.

The bearer token itself is a JSON web token using without any specified validity. It contains little information, merely a 13 character alphanumerical user token as well as a login count, counting the number of bearer tokens already created for the user.

On iOS devices, Lime makes use of the keychain feature. The keychain can be used by applications in order to store various passwords in an encrypted manner. Along with the account name (email or phone number), the bearer token is stored in the `keychain-2.db` file. For a forensic investigator, this complicates the analysis in cases when no decrypted version of the keychain database is available. In such scenarios, it may be more feasible to obtain a new token provided that access to the suspect's SMS or emails is given.

On Android devices, the bearer token can be found in the shared preferences file `com.limebike_preferences.xml` stored under `/data/data/com.limebike/shared_prefs/`. The token is stored in plaintext in the string entity with the name `authorization_token`.

3.1. User Data

On iOS, Lime uses the `LimeBike.sqlite` database file which can be found in the `Library/Application Support` directory within the application data directory, containing most of the essential information. User data can be found within the `ZPSUSER` table storing attributes such as a user's name (by default set to "Lime Rider"), the phone number, an email address and the user token, which can also be found in the bearer token. The same information can also be obtained by querying Lime's API using the bearer token for authentication.

3.2. Payment Data

Lime supports payment via PayPal, credit card and Apple Pay. Information about a user's payment data is found in the `ZPSPAYMENTMETHOD` table within Lime's SQLite database in which each row represents one payment method with a unique 13 character alphanumerical identifier. For credit cards, the information comprises the last 4 digits, brand as well as the expiry month and year, while for PayPal, it only indicates the type. We have found that even though payment methods are deleted, they could still be present within the database. Querying Lime's API returns the same information, but adds the associated email address to a PayPal account.

3.3. Ride History

For each ride performed using the Lime application, an entry in the table `ZPSTRIP` is created within the database. However, it only contains limited information about a ride including its start and end timestamp (on iOS as an Apple Core Data timestamp, counting seconds since January 1st 2001) and the 13 character alphanumeric identifier for the ride. No geo data is stored within the database.

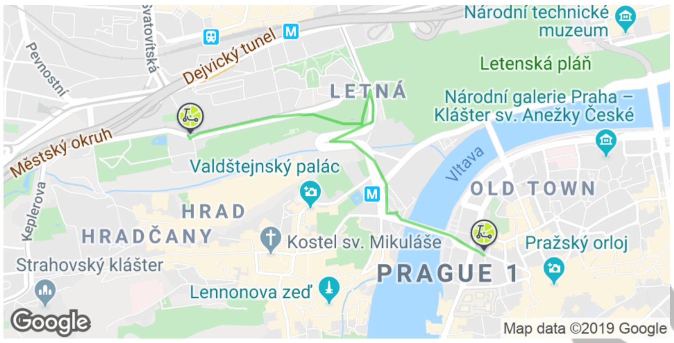


Figure 1: Image of the route of a Lime trip stored on CloudFront server.

A more detailed ride history can be queried using Lime's API. Besides the last three characters of the plate number, it contains temporal information such as the start and end time as well as the total duration in minutes for each ride. It also includes geo information, albeit no direct start or end coordinates of a trip are included. Instead, Lime stores a Google polyline for each ride in addition to a rendered PNG file of the performed route. An example for such a PNG file can be seen in Figure 1. On the image itself, the start and endpoint cannot be distinguished. Moreover, it is also not possible to easily extract other coordinates. However, the exact information is stored within the Google polyline. By decoding the polyline (Developers, 2019), it is possible to retrieve all stored GPS coordinates of a trip in correct order as shown in Figure 2. Additionally, the complete distance of the trip, which is calculated using the supplied coordinates, is given in meters. The rendered images, generated by Lime, can also be found in the `Library/RideHistory` directory in which each file name corresponds to the trip ID. Furthermore, the returned data contains the total cost of a trip as well as the corresponding currency used to pay, but no information about the used payment method. The used vehicle can be identified by its type (e.g. scooter) and the last three digits of the plate number. It also includes a label marking group rides.

Almost the identical information about a ride can be obtained by requesting direct information about it using its unique identifier. In addition to the aforementioned

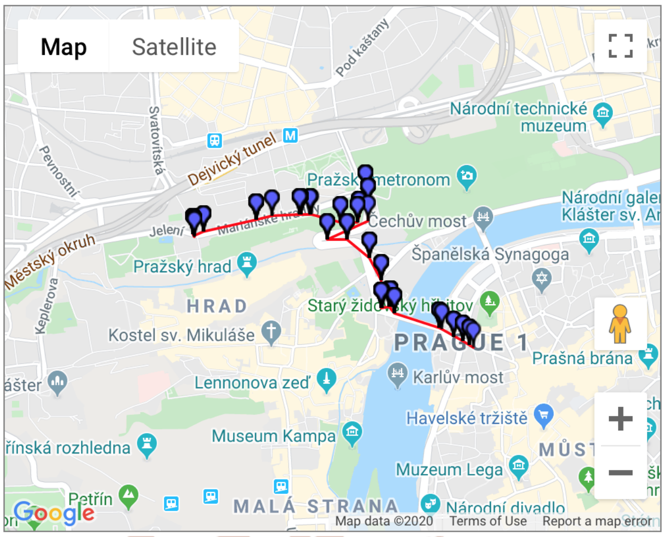


Figure 2: Decoded version of the route of a Lime trip.

data, it contains **current** information about the bike used during the trip. Since Lime provides only the last three letters of a bike's plate number, which is not sufficient to query recent data of a bike, this opens up a way of obtaining for example the recent location of a specific bike used during a ride. It is important to mention that the information about a certain trip requires the respective bearer token of the user who performed the trip.

3.4. Rental

Overview. An overview of all available scooters, can be queried from Lime's API. For each scooter, it contains the last three characters of their plate number, current location, last activity timestamp and possible range in meters.

The SQLite database contains the same information for scooters that have been used during a performed ride. It can be found in the `ZPSBIKE` table.

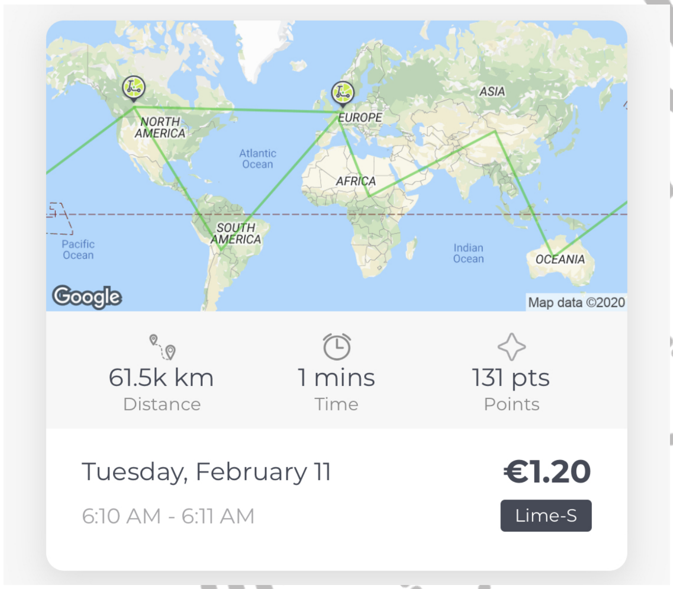
Renting. Lime scooters can be rented either by using their plate number or the QR code printed on their handlebars. Since only the last three numbers of the plate number and no QR code are included in the data that can be queried, it is not possible to start a ride of a scooter without any physical presence. However, the plate numbers and QR code do not change, so makes it possible to reuse them in order to start another ride with the same scooter from anywhere.

Riding. By specifying the plate number of a Lime scooter, it is possible to gain detailed information about it via Lime's API including its current GPS location. This is even possible during the ride, enabling us to track any scooter while it is being used. During our tests, this approach returned the actual GPS coordinates of a scooter. However, we have also found that in some cases mobile devices constantly sent GPS data to Lime's server. This data is also used to create the rendered overview maps and the polyline of a route after a trip has been finished.

Finishing. Ending a Lime ride can be performed via the application and is thus possible from any location and without any interaction with the scooter. After the ride is finished, it is possible to upload a photo of the parked scooter—however this is optional. Though the path to these images is unfortunately stored nowhere in the accessible data, it is important to consider that these images may exist.

3.4.1. Reliability

As previously described, it is possible to send custom GPS data to Lime's API during a ride enabling us to create any false route as soon as a ride has been started. By providing only the last three characters of a scooter's plate number, Lime, however, prevents the option to start a ride with any scooter at any location—unless its plate number (or QR code) was previously obtained. However, also the start and end point of a ride can be spoofed, making it possible to create a completely faked ride without containing the real location in any of the stored GPS data. Figure 3 illustrates this method of spoofing a ride by showing a trip around the world taken out of Lime's application.



61.5k km
Distance

1 mins
Time

131 pts
Points

Tuesday, February 11
6:10 AM - 6:11 AM

€1.20
Lime-S

Figure 3: 61.500 km Lime trip around the world in 1 minute.

4. TIER

According to their website, Berlin-based company TIER Mobility, founded in 2018, is Europe's leading company for shared micromobility services. It solely operates escooters, which can be found in more than 40 cities across twelve markets worldwide (Mobility, 2020b). Each of their scooters is identified by a six-digit number and can be rented via their application. With *myTier*, TIER provides a unique service for renting a personal scooter, which is not shared with other customers (Mobility, 2020a). Nevertheless, a myTier scooter is still part of TIER's ecosystem benefiting from their existing infrastructure, e.g. by enabling users to grant access to their scooter to family or friends via the TIER application. According to our evaluation, which

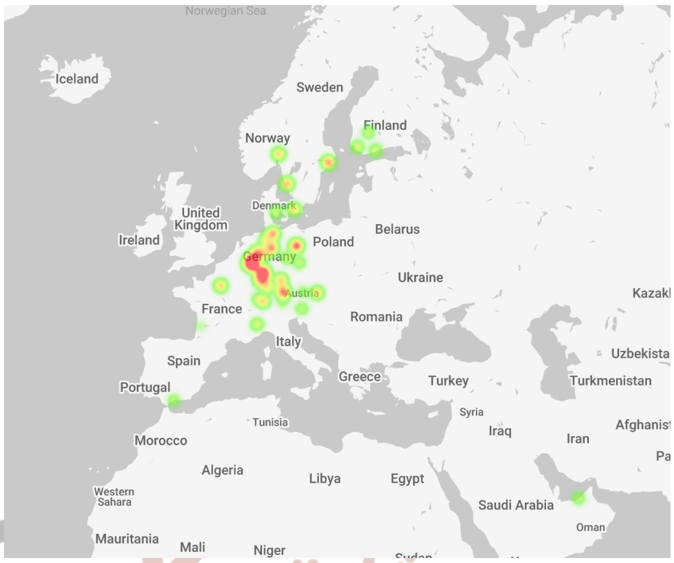


Figure 4: Distribution of TIER scooters (February 2020).

involved crawling all available TIER scooters, TIER Mobility is operating roughly 24.000 scooters as of February 2020. As illustrated in Figure 4, they are mainly spread across major cities in central Europe, while some can also be found in the United Arab Emirates. Around 7.000 of the crawled scooters are used for the myTier service leaving 17.000 scooters available to potential users.

Registration and Authentication. TIER requires a mobile phone number for the registration process, which is performed using Google's Identity Toolkit (Google Developers, 2019). After a six digit code has been sent and verified, an access token used for authorization against TIER's services as well as a refresh token is sent back. Unlike Lime for example, the access token has a limited validity of 3600 seconds until it expires. After that, the refresh token has to be used to create a new access token. For a forensic examiner, this means that an acquired access token, for example by sniffing network traffic, may be invalid when used for further analysis. Thus, it is better to aim for the refresh token. The access token used is a JSON web token. Apart from its expiration time, it contains the time it was issued as well as a 28 alphanumeric user ID and the corresponding phone number.

On iOS, the access as well as the refresh token can be found within a Firebase authentication `plist` file stored in the encrypted keychain. Thus, the user's passcode is required to access the information.

On Android, the authentication is also implemented via Firebase. The same two tokens are stored in the preferences file `com.google.firebase.auth.api.Store.<TOKEN>.xml`, where `<TOKEN>` is a token which can be found in the file `com.google.android.gms.appid.xml`. Both files reside in the folder /data/data/com.tier.app/shared_prefs/. The access token is stored under the key access_token, the refresh token under refresh_token. Both can be found in the JSON ducoment stored in the com.google.firebase.auth.GET-TOKEN_RESPONSE.<USERID> entity (<USERID> is defined in the same file).

4.1. User Data

On iOS, the file app.tier.sharing.plist, stored in the Library/Preferences directory of the application's data folder, contains information about the user. It includes the last known city the application was used in as well as the user's access token, which can be decoded to extract the aforementioned information. In most cases however, it will already be expired due to its limited validity. Using a valid access token, it is possible to request user data from TIER's back end. The returned data contains a unique user ID along with the ID stored in the token. Personal information includes the full name, phone number and email address, which cannot be changed after the registration process. Additional information can be queried, by sending a request containing a valid access token to the Google Identity Kit API. The information includes the last login as well as the registration time of the user.

4.2. Payment Data

Payment for TIER can be performed via PayPal or credit card. On iOS an entry for payments via PayPal can be extracted from the keychain. All of the payment data of a user can be obtained by querying TIER's API using the corresponding access token. The data contains information for each saved payment method including the date it was added, the provider used for the payment (Stripe or Braintree) as well as additional information. For credit cards, this includes the first four digits of the credit card number as well as the brand, for PayPal it stores the email address of the PayPal account.

4.3. Ride History

In late February 2020, TIER introduced the feature to get an overview of one's ride history. It includes information about the used scooter, time stamps as well as the total cost of the ride. Also included is a link to a PDF file of the invoice for the ride. On iOS, invoices which have been downloaded by a user can be found within the Documents folder. Any invoice can also be requested from TIER's servers using the corresponding invoice ID stored in the ride history. Apart from the user's full name, an invoice contains even fewer information than displayed in the application. Unfortunately, both the application and the invoice lack information about any geo data of a ride. To close this gap, it is possible to query a different API endpoint, delivering more detailed information about each ride performed. This includes the user's start and end location, indicating the locations sent by the application when a scooter was rented and returned respectively. Furthermore, it also contains the vehicle start and end location, marking the locations sent by the scooter's own GPS during the start and end of a ride.

4.4. Rental

Overview. Available scooters can easily be queried via the API by specifying a location and a radius. The returned information includes detailed information about nearby scooters such as their battery level, six-digit vehicle code or a timestamp of their last location update. Using the application instead, it is not possible to display scooters which are too far from the mobile's current GPS location.

Renting. Starting a trip and renting a TIER scooter only requires its six-digit vehicle code, which can easily be obtained as previously described. Thus, it is possible to rent an arbitrary scooter from any location without physical presence.

Riding. TIER provides an API endpoint to obtain detailed information about a specific scooter, including its current location, by using its six-digit vehicle code. This request is also possible for scooters, which are currently used and rented by any other user. This makes it possible to track any TIER scooter and get detailed information about its location in real time similar to Lime. Since no information is sent from the user's device, the received location corresponds to the scooter's own GPS data. An example of a tracked scooter in Gothenburg, illustrating the intervals between each GPS updates can be seen in Figure 5.

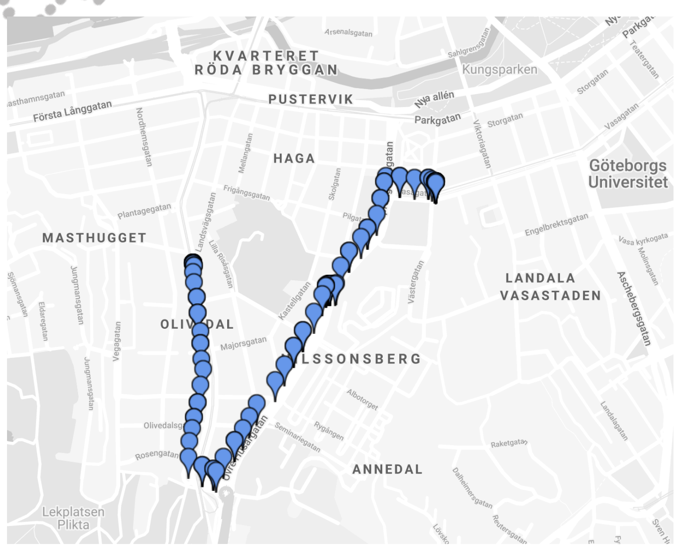


Figure 5: Tracking a TIER scooter in Gothenburg.

Finishing. Ending a ride does not require any physical interaction with the vehicle and can, thus, also be performed from any location.

4.4.1. Reliability

Since the six-digit vehicle code of a scooter can be queried directly and renting as well as finishing a TIER ride do not require any physical presence, it is fairly easy to create a fake ride within any operated city. Nevertheless, since no GPS information during the ride is stored, it is only possible to alter the start and end location of a trip by sending fake coordinates. However, this still leaves the correct vehicle start and end locations untouched, making it possible to find mismatches in the stored data and to identify a spoofed ride.

5. nextbike

nextbike is a German bike sharing provider operating in more than 200 cities in 25 countries with varying types of bikes (nextbike GmbH, 2019). As distinguished from the aforementioned scooter providers, which require the app in order to lock a scooter, nextbike also offers bikes with a frame lock requiring physical presence at the bike in order to end the trip. While many of the bikes are provided in cooperation with local traffic companies, all of them are consistently identified by a bike number currently going up to six digits. According to our evaluation, nextbike

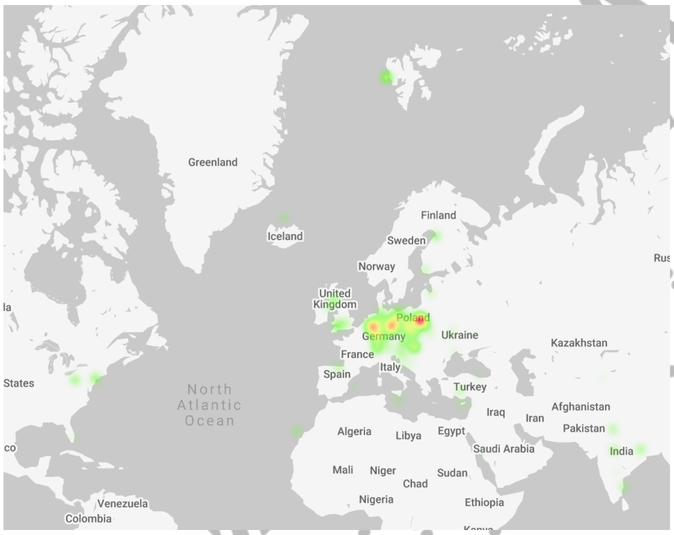


Figure 6: Distribution of nextbikes (February 2020).

currently operates around 76.000 vehicles, whose locations are shown in Figure 6.

Registration and Authentication. Registration for nextbike requires a valid phone number to which a six-digit PIN code is sent via SMS. The combination of phone number and PIN code is used for login. Afterwards, a 16 character alphanumerical login key is returned and used for authentication against the nextbike API. The same PIN and mobile combination can be used to login again, however resulting in a different login key. Thus, obtaining access to old messages of a suspect during an analysis, yields the required data to login into his or her nextbike account. The five most recent login keys have proven to be valid as long as no new PIN has been requested.

On iOS, the currently used login key can be found within the `net.nextbike.official2012.plist` preference file of the application.

On Android, the login key is stored in the file `USER_SHARED_PREFERENCES.xml` in the shared preferences directory `/data/data/de.nextbike/shared_prefs/`. Here, the JSON document stored in the entity with the name `userEntity` contains the key `loginkey`. Its corresponding value is the login key in plaintext. The API key can be found in the LevelDB stored under `/data/data/de.nextbike/app_webview/De` `Storage/leveldb/`. The key for the API key is `_https://webview.` Alternatively, both keys can be found in the cache directory of the application.

5.1. User Data

Apart from the the login key and the user identifier, which is comprised of the user's phone number and stored in the preferences file, no additional user data could be found within the local data on iOS. By using the login key and performing a request to nextbike's API, it is possible to obtain information about the associated user including his or her full name, mobile number, email address and used currency.

5.2. Payment Data

A rental with nextbike can be paid via PayPal, credit card or direct debit, but only one payment method can be stored at the same time. The only available information about it, is included in the user data returned via the API and indicates merely the payment type (e.g. "pp" for PayPal or "wc" for a Visa credit card) without any additional information.

5.3. Ride History

nextbike stores information about all performed rides of a user. The data is not stored locally on iOS, but can be obtained via an API endpoint. For each ride, it returns start and end time encoded as a Unix time stamp, representing the seconds since January 1st 1970, as well as the bike number used during the trip. Regarding geo information, nextbike stores the start and end place of a ride, which are identified by a unique identifier along with the corresponding coordinates. The advantage of using an identifier for places comes into place in cases when bikes are returned at dedicated areas within cities, easily identified by their place id. However, each time a bike is returned at position outside of these areas, a new place is created. Each place is given a name to identify it, e.g. for a special Campus building. When a bike is returned at an unknown place, the created place is given the name of the bike. Querying a place using its ID via the API also returns a list of all bikes available at this place at the moment as well as its GPS coordinates and associated city name.

5.4. Rental

Overview. Using nextbike's API, all available bikes within a city can be requested by specifying the corresponding city ID. For each available bike, the returned data includes its bike number, place information (including its name, ID and coordinates) and additional data such as the bike's type. Requesting detailed information of a bike yields the Bluetooth MAC address of the bike. Though Bluetooth is not required for the usage of nextbike, this information can be used in case any Bluetooth records or logs may be found on a suspect's device.

Renting. A nextbike trip can be started in various ways depending on the bike or lock type. Around 23.000 available nextbikes are using a frame lock. For this type, it is sufficient to know the bike number in order to unlock the lock automatically and start the trip. Other types include a fork as well as an analog code lock. In such cases, it may be necessary to enter a PIN code on the bike's lock in order to unlock it, thus requiring physical presence even when a bike's number is known.

Riding. During the ride, no data is sent from the mobile device. It is also not possible to track a bike while it is being used.

Finishing. As previously described, nextbike offers multiple types of bikes. As far as we know, all of them require physical presence in order to close the lock and end the trip. It is not necessary to use the application for this step.

5.4.1. Reliability

Seemingly, it is not possible to induce any faked data within nextbike's ride history. This is because it does not record any GPS data during the trip, but also due to the fact that start and end location of a ride are sent by the bike itself.

6. Voi

As stated on their website, Voi Technology, founded in Stockholm in 2018, was the first European scooter operator (Technology, 2020). They are operating only scooters, identified by a four character alphanumberic ID, across major cities in Europe. According to our evaluation, roughly 70.000 available scooters are distributed across Europe as shown in Figure 7.

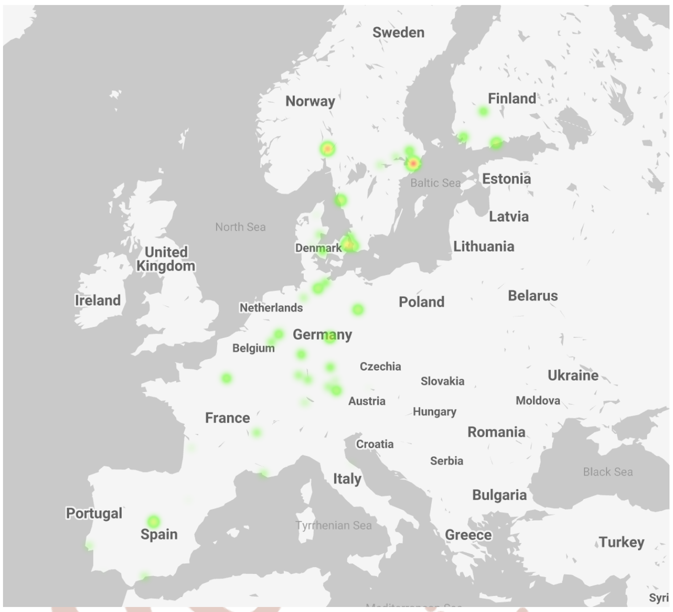


Figure 7: Distribution of Voi across Europe (February 2020).

Registration and Authentication. Registration with Voi is only possible with a valid phone number to which a six-digit verification code is sent. It is also necessary to provide an email address during the registration process, though it is never validated in any way. After the verification code is received, it is used to obtain an authentication token with a validity of 36 hours. Besides the time it was issued as well as its expiration date, the token also contains the UUID of the user it belongs to. This special authentication token issued during registration can only be used once in order to obtain a new authentication token, with a longer validity of 3 months, as well as an access token with a limited validity of 15 minutes. Both of these tokens can be used to interact with Voi's API, but also to create a new pair of authentication and access tokens. During our tests, all created tokens stayed valid until their expiration time, even when new ones were generated in the meantime. However, as soon as the registration process was repeated, these tokens could not be used to generate new tokens anymore, while they still granted access to the API. For an analysis taking place 3 months after the last token was issued, so after its expiration, the aforementioned registration process has to be repeated in order to gain access to Voi's API.

Both, the authentication and access tokens are stored in the encrypted keychain database on iOS devices.

On Android, both tokens are stored in the file `io.voiapp.voi.U` `DATA.xml` located under `/data/data/io.voiapp.voi/shared_prefs/`. The access token is stored in the entity with the name `user_access_token`, while the authentication token is stored in the entity with the name `user_auth_token`.

6.1. User Data

Voi does not store any valuable information within their preference file on iOS nor does it make use of any dedicated databases. However, user data can be queried using their API by specifying the corresponding access token. It contains the user's provided email address and phone number, country, identifier and a timestamp of the time the account was created.

6.2. *Payment Data*

For payments, Voi accepts credit card as well as PayPal, however, only one payment method can be stored at the same time. For PayPal the information returned by the API contains the provider (Braintree) as well as the corresponding email address of the PayPal account. For credit cards, it contains the brand and the last four digits of the credit card as well as the provider for the payment process (Stripe).

6.3. *Ride History*

Information about a user's rides can easily be requested from Voi's API specifying the corresponding access token. For each ride, it contains the total duration in minutes, an exact start and end time stamp, as well as the start and end location. Additionally, Voi also translates the GPS information into start and end addresses which are stored along the coordinates. Regarding payment, the data includes the total cost of the trip, the currency used for the payment and also the payment method used (e.g. the last four digits and the brand for credit cards). Besides the UUID of the user performing the ride and the ID of the used vehicle, a device token can be found within the data. This token makes it possible to distinguish between different devices that have been used to start rides using the same account. The returned API data also contains a link to the ride's receipt, which also includes the most important information of a ride.

6.4. *Rental*

Overview. An overview of Voi scooters within a certain zone, identified by its zone ID, can easily be queried via their API. While the application does only partially show the four-character ID of available scooters, the API in fact returns the complete ID. Additionally, the information also includes the scooter's battery level and UUID.

Renting. Renting a scooter is possible by scanning its QR code or entering its four character identifier. Since the API returns the complete identifier as previously mentioned, it is possible to rent any scooter without any physical presence.

Riding. During the ride, the application constantly sends GPS positions to Voi's API. This information, however, is not stored and cannot be found within the ride history. It is only used to update the location of a scooter during its ride, which can be obtained by querying detailed information about a specific scooter using a different API endpoint. Thus, it is also possible to track Voi scooters while they are being used.

Finishing. Ending a trip is also done using the Voi application without requiring any physical interaction with the vehicle.

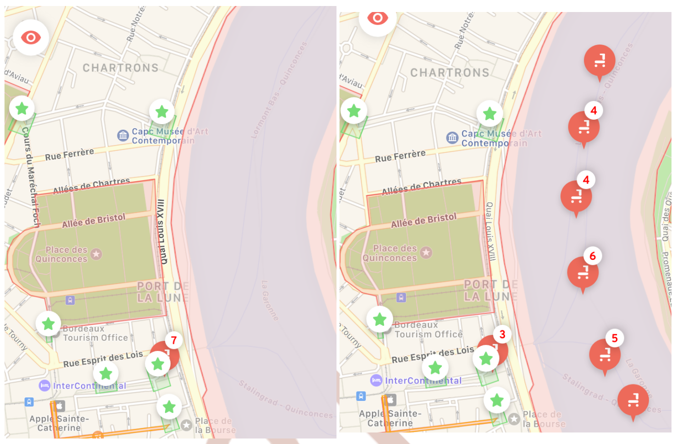


Figure 8: Voi's application map before and after throwing Voi scooters inside the Garonne river in Bordeaux—virtually though.

6.4.1. *Reliability*

While the end location of a trip is sent by the user, the recent scooter location before a ride is taken as the start location. As previously described, however, Voi provides an API endpoint, which can be used to alter the location of a scooter. This makes it is possible to fake both, start and end coordinates stored within the ride history. It is also possible to constantly set a scooter's location to any arbitrary coordinates and thus creating a complete fake path, while it is being tracked for instance. Taking this a step further, it is even possible to alter the location of Voi scooters when no ride has been started, using the same tracking endpoint. That implies that Voi scooters can be placed at any arbitrary location causing disturbance in their availability across cities. An example for this behaviour within the Voi application can be seen in Figure 8. The faked location of a scooter remains active until it is being updated with its internal GPS data.

7. Conclusion

In this paper, we took a closer look at prominent examples of micromobility solutions from a forensic point of view including Lime, TIER, nextbike and Voi. Initially, we evaluated the different kind of data sources available during an analysis. While only Lime makes use of local storage on iOS in order to record crucial data, all applications store their data within the cloud and make it accessible via their APIs. In fact, we have shown the richness and importance of the data that can be obtained via provided APIs. For this, we described how the presented applications perform authentication and what is necessary in order to impersonate a user and gain access to his or her data via APIs. Except for nextbike, all applications store the authentication tokens within the encrypted keychain on iOS, which requires the passcode of the device to be accessed. In such cases, it may be more feasible to obtain access to the suspect's phone account in order to gain new authentication tokens.

Furthermore, we presented different kinds of categories of data and forensic considerations, which are of interest to an investigator including the user's personal data, payment information as well as the ride history. Although all of the described providers contain at least this kind of data, each of them stores the data to a different extent and implements their own API. In order to assist investigators during their analysis, we implemented a forensic toolkit in order to obtain data from the presented provider's APIs (Fraunhofer FKIE, 2020). The repository not only contains the toolkit, but also the complete results of our Android analysis of local application data.

At last, we evaluated the reliability of the data regarding a user's movements and described the process of creating spoofed data. For Lime, it is possible to create complete fake trips including an arbitrary path, as long as a valid license plate of a scooter is provided. If this license plate is chosen consistently (e.g. does not belong to a scooter of a different city), it is unlikely to identify a spoofed trip. The same holds for Voi, whose scooters' locations can be altered and changed to any location, even during a ride. This makes it possible to set start and end location of a trip without any possibility of identifying the spoofed ride. On the other hand, other providers complicate the possibility of spoofing rides. TIER for example stores the user-provided start and end location, however, these values can easily be compared to the vehicle start and end location in order to identify fake trips. nextbike does not evenrely on user-provided GPS data at all for their start and end locations making it impossible to fake any trip without any access to the bike's firmware itself. This evaluation demonstrates, that additional external information should always be used and compared to other obtained data in order to prove the plausibility of ride histories and their movement data.

Future Work. As already described, obtaining data from APIs is very application-specific. For this reason, it is necessary to take a closer look at the implementation of other micromobility providers in order make their data available for a forensic analysis.

Furthermore, it is an interesting topic to analyse the communication between vehicles and their provider's backend. This could make it possible to add an arbitrary, non-existing vehicle to the ecosystem and use it to store spoofed information into the databases.

References

Developers, G., 2019. Interactive Polyline Encoder Utility — Google Maps Platform. URL: `https://developers.google.com/maps/documentation/utilities/polylineutility`.

European Parliament, Council of the European Union, 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). URL: `https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A32016R0679`.

Fraunhofer FKIE, 2020. fkie-cad/micromobility-forensics: Forensic Analysis of Micro Mobility Solutions. URL: `https://github.com/fkie-cad/micromobility-forensics`.

Google Developers, 2019. Introduction — Identity Toolkit — Google Developers. URL: `https://developers.google.com/identity/toolkit`.

Lime, 2020. About Lime — Scooter and Bike Sharing Network for Cities and Universities. URL: `https://www.li.me/about-us`.

Mobility, T., 2020a. myTIER Online Shop. URL: `https://mytier.app`.

Mobility, T., 2020b. TIER Mobility launches operations with swappable batteries as the first e-scooter provider worldwide. URL: `https://www.tier.app`.

nextbike GmbH, 2019. nextbike — Germany's biggest Bike Rental. URL: `https://www.nextbike.de/en/faq/`.

Technology, V., 2020. Voi Technology - Shaping cities for people. URL: `https://www.voiscooters.com/voi-technology/`.

Vinayaga-Sureshkanth, N., Wijewickrama, R., Maiti, A., Jadliwala, M., 2020. Security and Privacy Challenges in Upcoming Intelligent Urban Micromobility Transportation Systems, in: Proceedings of the Second ACM Workshop on Automotive and Aerial Vehicle Security, Association for Computing Machinery, New York, NY, USA. p. 31–35. URL: `https://doi.org/10.1145/3375706.3380559`, doi:`10.1145/3375706.3380559`.