Dissecting a Cloud-Connected E-Scooter

JD-HITBSecConf 2018
November 2, 2018
Beijing, China

Nikias Bassen
@pimskeks
nikias@corellium.com
Outline

• Introduction

• The Target

• Smartphone App

• GSM/GPRS Connectivity

• Small Demo

• Conclusions
About me (1)

• IT Expert from Germany, Diploma in Computer Science (University of Bremen, Germany)

• Involved in RE & Security Research for > 15 years 😱

• RE of iTunes database hashing algorithm

• RE of iTunes/iOS communication protocols

• Leading Developer of libimobiledevice project
About me (2)

• 2018-now  VP of Platform & Security, CORELLIUM

• 2017-2018  VP of Platform Research, ZIMPERIUM

• 2015-2017  Mobile Security Researcher, ZIMPERIUM

• 2010-2015  Self-Employed, custom IT solutions
  - RE & Research as a hobby
  - 2013 evad3rs
  - 2012 Jailbreak Dream Team
  - 2011 Chronic-Dev Team
Why this topic?

• Started to work ~6 months ago at Corellium, virtualizing iPhones (amazing stuff!)

• We have lots of work to focus on, no time to do any research (hopefully again in the near future)

• No completed research on iOS currently

• I just bought that E-Scooter, and said "why not?!"
The Target
The Target

- Niu N1S E-Scooter
- Jiangsu Niu Electric Technology Co., Ltd., China
- Cloud-Connected (GSM)
- Smartphone App
- USB Port
USB Port - Diagnostics?

- Vendor: charging port for phone
- Me: maybe used for diagnostics?
- Raspberry Pi: Nah…
- Nope, doesn’t detect anything
Real Diagnostics Port

- Battery charging port under the seat also used as diagnostics port
- Used by NIU dealer with dedicated diagnostics device
- Supposedly RS-485 serial communication
- Couldn’t check, lack of time and hardware
China Shopping List ++

Yes, the connector won’t fit, but you get the idea :)
Smartphone App
Smartphone App

- Battery level & estimated distance
- Lock status
- Current location
- Weather report
- Overview of past trips and statistics
- Smart Check (scooter self-diagnosis)
- Service information
- Push notifications about unexpected movement, battery removal, etc.
Registration + Binding

• Account registration required with Phone number or Email

• Scooter needs to be bound to account

• S/N required, printed on manual (QR code), not found on vehicle itself

• By default, adding someone’s S/N requires confirmation (see screenshot on the right)

• One vehicle can be bound to 5 accounts max.
Let’s dump the App

- Jailbroken iPhone + Clutch
- IDA Pro Disassembler
- ID: com.niu.xiaoniuAborad
- Lastest version: 3.4.8
  (version initially dumped 3.4.6)
- Binary: managerAborad.app/
  managerAborad
- Most likely a typo
  Aborad => Abroad
First, let's have a look
First, let's have a look
```
{
    "payload": {
        "USER_LOGIN": {
            "desc": "1.1. 用户名密码登陆接口",
            "url": "https://account.niu.com/appv2/login"
        },
        "USER_SENDCOCE": {
            "desc": "1.2. 获取验证码接口",
            "url": "https://account.niu.com/appv2/sendcoce"
        },
        "USER_RESETPASSWORD": {
            "desc": "1.3. 重置密码接口",
            "url": "https://account.niu.com/appv2/resetpassword"
        },
        "USER_SIGNUP": {
            "desc": "1.4. 用户注册接口",
            "url": "https://account.niu.com/appv2/signup"
        },
        "USER_LOGOUT": {
            "desc": "1.5. 退出登录",
            "url": "https://account.niu.com/appv2/logout"
        },
        "USER_BASICINFO_UPDATE": {
            "desc": "1.8更新个人信息",
            "url": "https://account.niu.com/appv2/basicinfo/update"
        },
        "USER_UPDATETIPUSHID": {
            "desc": "1.10. 更新极光推送id接口",
            "url": "https://account.niu.com/appv2/updatetipushid"
        }
    },
    "VEHICLE_SETSNNAME": {
        "desc": "3.3. 给车命名接口",
        "url": "https://app-api.niu.com/motoinfo/setsname"
    },
    "VEHICLE_LIST": {
        "desc": "3.4获取已绑定车辆列表接口",
        "url": "https://app-api.niu.com/motoinfo/list"
    },
    "VEHICLE_SETDEFAULT": {
        "desc": "3.5设置默认车辆",
        "url": "https://app-api.niu.com/userinfo/setdefault"
    },
    "VEHICLE_CURRENTPOS": {
        "desc": "3.6. 获取当前车辆坐标",
        "url": "https://app-api.niu.com/motoinfo/currentpos"
    },
    "URL_VEHICLE_BATTERYINFO": {
        "desc": "3.8. 电池信息接口",
        "url": "https://app-api.niu.com/v3/motor_data/battery_info"
    },
    "URL_VEHICLE_BATTERYINFO_V2": {
        "desc": "3.23. ",
        "url": "https://app-api.niu.com/motoinfo/batteryinfo/v2"
    },
    "VEHICLE_BINDLIST": {
        "desc": "3.12. 车主查看已绑定用户列表",
        "url": "https://app-api.niu.com/userinfo/bindlist"
    },
    "VEHICLE_RENAME_BIND_USER": {
```
}
Web API!

• URLs for different actions:
  • User signup, login, account & permission settings
  • Vehicle position, battery and health status, smart check
  • Service status, Ownership transfer
  • Theft reports
  • Driving statistics
  • Some social media stuff
• Same API calls, just different base URL
  account-dev.niucache.com instead of account.niu.com
  app-api-dev.niucache.com instead of app-api.niu.com

• App offers test account
Let's check how this works
Let’s check how this works

```
$ curl -H "Content-Type: application/json" --request POST --data '{"account": "nXXX@YYYY.ZZ", "password": "yeah,Right"}' https://account.niu.com/appv2/login
{"data":{"token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJzaHJlY3QifQ.eyJ0eXBlIjoiQGhvdy5wdXRob3JpdHgwYXNzZXRzLm5ldCIsI2NlNjY3OTk1IiwicG9wdWIiOiJib3N0aWxlIiwid2lkdGgiLCJzdHlsZSI6eyJtb21lciI6W119fQ", "user": {"loginid": "", "v": 1, "iat": 1540677733, "exp": 1548453722}}}
```

Token!

JSON Web Token!
We can query data!

- **Vehicle(s) bound to account:**

  ```bash
  $ curl -H "Content-Type: application/json" --request POST --data '{"token" : "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9..", "sn" : "NAS************", "specialEdition" : "", "vehicleColorImg" : "https://app-api.niuocache.com/static/app-api/static/product/default/engineering@2x_95b256a.png", "vehicleLogoImg" : "", "vehicleTypeIdx" : "N-series52", "indexHeaderBg" : "https://app-api.niuocache.com/static/app-api/static/pc/v3/na/headerBg/pic_background_N1s_grey@3x_d58acca.png", "scooterImg" : "https://app-api.niuocache.com/static/app-api/static/pc/v3/na/productImg/pic_EU_moto_NA_grey_matte@3x_f2e4853.png", "batteryInfoBg" : "https://app-api.niuocache.com/static/app-api/static/pc/v3/na/batteryBg/pic_background_N1s_grey@3x_5841b73.png", "myPageHeaderBg" : "https://app-api.niuocache.com/static/app-api/static/pc/mytitlebackground/n1/bg_my_tittle_n1@2_d8a5504.png", "listScooterImg" : "https://app-api.niuocache.com/static/app-api/static/pc/v3/na/listScooterImg/pic_moto_NA_grey_matte@3x_917b290.png", "name" : "Blitzdings", "frameNo" : "R1B***********", "engineNo" : "RBNFF***********", "isSelected" : true, "isMaster" : true, "bindNum" : 1, "renovated" : false, "bindDate" : "1540649740000", "isShow" : true, "gpsTimestamp" : "1540688956124", "infoTimestamp" : "1540688956124", "productType" : "native", "process" : "", "brand" : "", "isDoubleBattery" : false, "features" : ["featureName" : "gpsSwitch", "isSupport" : false, "switch_status" : ""]}, "type" : "N-series Grey (Matte) "]
```

- **Vehicle position (requires SN):**

  ```bash
  $ curl -H "Content-Type: application/json" --request POST --data '{"token" : "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9..", "sn" : "NAS************"}', https://app-api.niuocache.com/motoinfo/currentpos
  ```
  ```json
  {
    "data" : {
    }
  }
  ```
... and some more ...

- Battery information:

```
$ curl -H "Content-Type: application/json" --request POST --data '{"token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...","sn":"NAS**
************"}' https://app-api.niu.com/motoinfo/batteryinfo/v2
```

- Firmware information:

```
$ curl -H "Content-Type: application/json" --request POST --data '{"token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...","sn":"NAS**
************"}' https://app-api.niu.com/motorota/getfirmwareversion
```

A new function has been added to allow vehicle owners to change the status of the GPS sensor on the scooter.
Let's rename the scooter!

• Change vehicle name:

```bash
$ curl -H "Content-Type: application/json" --request POST --data '{"token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...","sn":"NAS**************","name":"Fancy E-Ride"}' https://app-api.niu.com/motoinfo/setsname
{"data":null,"desc":"车辆名称不能为空","trace":"SnName cannot be empty","status":1305}
```

• Whoops! Nice, web API speaks Chinese and English! "车辆名称不能为空" => "Vehicle name cannot be empty"

• Let's try again:

```
$ curl -H "Content-Type: application/json" --request POST --data '{"token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...","sn":"NAS**************","name":"Fancy E-Ride"}' https://app-api.niu.com/motoinfo/setsname
{"data":null,"desc":null,"trace":"Success","status":0}
```
Let’s rename the scooter!
So what else can we do?

- Web API requires Authentication ✓
- Uses HTTPS ✓
- No certificate pinning ❌
- Vehicle S/N bound to account, can’t be added by default, owner confirmation required ✓
- Some API calls even require confirmation by account owner by SMS or Email, e.g. ownership transfer ✓
- Attacker could MITM the connection, but bad stuff can’t easily be done, bind permissions just require a token though 😞
GSM/GPRS Connectivity
GSM/GPRS Connectivity

- Scooter comes with installed Prepaid SIM-Card (installed by Importer / KSR Group in Europe)
- Always connected (if there is network…)
- Scooter has a separate ECU battery, that lasts for about 3-4 days if main battery is unplugged
- Gives GPS and vehicle information without main battery
Let’s hack that GSM already!

- OK What do we need?
  - Something that can modulate GSM frequencies
  - Something that acts as a GSM base station
GSM Hacking Equipment

• While certainly not the best, this equipment works:
  • bladeRF x40 + GSM Antennas
  • Raspberry Pi 3
  • YateBTS base station software
  • Power!
So how to set this up?

• My former co-worker Simone Margaritelli (@evilsocket) tried this before:  
  https://www.evilsocket.net/2016/03/31/how-to-build-your-own-rogue-gsm-bts-for-fun-and-profit/

• However, he removed the version requirements which are really important for this to work.

• This blog article has all the information though:  
Let’s try it?! Not so fast.

- If you want a GSM device connect to your BTS, you need to simulate the right network

- Germany has 3 PLMN's:
  - Telekom (26201)
  - Vodafone (26202)
  - and Telefónica (26203)
Also, power...

- The integrated USB port rates 1 Amp only. This isn't enough to properly power the Raspberry Pi AND the bladeRF at the same time
- Strong battery pack or power supply via mains needed
Sounds easy, right?

• Doing a quick research, it showed that the importer said in a press release that they partnered with Vodafone

• So let’s set this up to simulate Vodafone.de!

• Also, make sure to select a correct frequency in the right band (Vodafone uses GSM900 and GSM 1800)
YateBTS configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radio Band</td>
<td>EGSM000</td>
</tr>
<tr>
<td>RACIO C1</td>
<td>#1C: 937 MHz Downlink / E9</td>
</tr>
<tr>
<td>Identity MCC</td>
<td>269</td>
</tr>
<tr>
<td>Identity MNC</td>
<td>02</td>
</tr>
<tr>
<td>Identity LAC</td>
<td>1000</td>
</tr>
<tr>
<td>Identity CI</td>
<td>in</td>
</tr>
<tr>
<td>Identity.BSIC.BCC</td>
<td>2</td>
</tr>
<tr>
<td>Identity.BSIC.NCC</td>
<td>0</td>
</tr>
<tr>
<td>Identity.ShortName</td>
<td>dtop</td>
</tr>
<tr>
<td>Radio.PowerManager.MaxAttribDB</td>
<td>36</td>
</tr>
<tr>
<td>Radio.PowerManager.MinAttribDB</td>
<td>35</td>
</tr>
</tbody>
</table>

Section [gsm] controls basic GSM operation. You MUST set and review all parameters here before starting the BTS!

Note! To disable nb mode and enable roaming mode see Javascript Roaming
Now, wait…

• You can wait for a long time…

• Especially if you have a BTS near your home 😏

• If a nearby BTS has a strong signal the Scooter won’t connect

• But my phone always has bad network at home so this must work somehow…
Then suddenly…

• My phone - which also uses Vodafone - receives text message

Your allocated phone no. is 4955777777. Thank you for installing YateBTS. Call David at david(32843)

• Turns out the BTS actually works!
OK, let’s wait longer…

• I was already thinking about other solutions when I suddenly realized that the BTS showed a new subscriber!
Gotcha!

• IMSI shows prefix of 20404 - Vodafone Netherlands
• This SIM is actually Roaming! 😞
• Let’s see what else we can find out?

YateBTS verbose log output:
Let’s check out the IMEI
OK so what next?

- We want to MITM the connection between Scooter and remote server
- YateBTS supports GPRS routing
  => Remember to enable IP forwarding and IP masquerading on the Raspberry Pi!
- Let’s ask YateBTS’ SGSN (Serving GPRS Support Node)
- It doesn’t want to connect through GPRS 😭
Then, I lost the connection…

• The Scooter disconnected. I waited and waited, but it didn’t want to reconnect anymore…

• I had to come up with an idea to make it connect just to my BTS

• I tried setting the MCC and MNC to 20404, but it didn’t want to connect

• I tried restarting YateBTS, but nothing worked
Ideas, I need ideas...

• Maybe it connects via 3G or even LTE? I was skeptical but then also I didn’t know…

• Too bad I didn’t by that frequency jammer last time I was in Shenzhen, I knew I would need it!

• Let’s build a faraday cage?
Come on...

• Need to find a way it can’t find a real BTS to connect to

• At my son’s school there’s really bad reception, let’s go there…

• Still there seemed to be too much signal strength 😩

• Also, the battery pack I had, and also my MacBook couldn’t properly power the bladeRF…

IT CAN'T CONNECT TO A BTS

IF THERE IS NO BTS
Then, I had an idea

- I remembered there’s a parking garage nearby, *A DARK AND SHADY PLACE!*

- This must work! If there is no BTS it *JUST HAS* to connect to mine, right?

- Only problem was power… But I have a power converter in my car so that should do it
Into the Darkness…

- So I entered the garage and the scooter actually lost signal — PERFECT!

- I set up the BTS and everything, and waited…

- … and waited …

- I couldn’t believe it. It didn’t want to connect even though I am the only reachable BTS

- But somehow my phone also didn’t want to connect, not sure what was wrong… maybe interference? maybe the smell?
...there was another problem

• Even if it would connect to the BTS, it wouldn’t be able to connect to the internet (via YateBTS’ SGSN)

• Even my phone didn’t have a signal so I couldn’t use my hotspot

• I was disappointed and out of ideas, and went home

• I was about to give on on this, actually 😭
Let’s give it another try

• I set up my BTS at home again, because I said, hey it connected once maybe it connects again, what do I have to lose?

• But it didn’t want to connect. For an entire day, nothing happened. The real BTS was still too strong…

• I unscrewed the front panel of my scooter to check where the GSM module sits. It is in the upper front.

• But it has a sticker WARRANTY VOID IF BROKEN so I didn’t really want to mess around with that…
Making the signal weaker?

- Aluminum foil!
- Didn’t help, GPS signal lost a few bars though
- I re-parked my scooter so that my car would be between it and the BTS
- Still no real change…
Then...

- Suddenly, activity in the console where YateBTS was running 😳
- First I thought it’s probably my phone again but…
- IT ACTUALLY CONNECTED
Wait, let’s check the SGSN

• It was connected through the SGSN!

• Let’s dump some packets!

• Uh wait. How do we even do that? Did I enable GSM/GPRS tapping in YateBTS?

• I didn’t but…
Phew...

- Luckily, YateBTS creates a TUN device "sgsntun"
- So on the Raspberry Pi I can now do:
  `tcpdump -i sgsntun -n -v -w dump.pcap`
- Packet counter increased slowly, every few minutes
- With ignition on, it sends packets every few seconds
- I copied the dump.pcap to my computer and ran it through Wireshark
The vehicle gateway!

Let’s have a look at what we captured:

- Resolves ecu.niu.com via DNS
- Sends UDP packets to ecu.niu.com on port 8888
- (That ICMP is my attempt to ping the scooter)
The packets

- Binary packet format
  Seriously, I was expecting JSON!

- Let’s try to figure something out by looking at consecutive packets

- Shows some common patterns but also large parts that change

- Especially last few ~20 bytes

- Checksum? SHA1?
Packet Checksum

- Turns out to be MD5!
Packet format?

- First two (?) bytes seem to define the type of the packet
- Can’t really figure out a length field or anything obvious
- It needs to contain vehicle identification and GPS coordinates
Packet format?

• Seems somehow encoded. None of the Vehicle SN, or frame number or engine number seem to match in any way.

• Still it must have some kind of identification, otherwise it wouldn’t know which scooter sent the data.

• Even though we don’t understand the packet format completely we know that it has a checksum
What can we do with this?

• We can modify a packet, and apply the correct checksum and send it to ecu.niu.com 8888

• In the hopes of supplying different GPS coordinates I tried, but no reaction in the app…

• Research continues… (happy if someone has ideas!)
Can’t we do something?

• Maybe we can replay packets?

• Let’s use a simple python script that just reads a file and sends it to ecu.niu.com port 8888

• I could submit a slightly different position from a few minutes ago and it showed up in the app 🤘

• Let’s think about this. Meanwhile, let’s look at something else...
OTA Firmware update?

- Yes, the Niu can be updated over the air! Isn’t that awesome?

- Since we can now dump the traffic, let’s do this. What could possibly go wrong when it goes through our BTS?
Triggering the update

• To trigger an update, the Web API has this: POST to https://app-api.niu.com/motorota/updatemotor with SN (and token of course)

• To make the scooter start the update you have to turn the ignition off and on again, and then it shows progress:
Now be patient...

- The app says it will take about 10 minutes

- From the API we actually know the update size:

```
$ curl -H "Content-Type: application/json" --request POST --data '{"token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...","sn":"NAS***
************"}' https://app-api.niu.com/motorota/getfirmwareversion
{"data":{"needUpdate":true,"otaDescribe":"<p class="p1">A new function has been added to allow vehicle owners to change the status of the GPS sensor on the scooter.</p>","nowVersion":"TRA01C07","version":"TRA01C10","hardVersion":"V2.0","ss_protocol_ver":2,"isSupportUpdate":true,"byteSize":42384,"date":1526885222572},"desc":"成功","trace":"","status":0}
```

- So while we wait, let's take a look at the traffic...
## Start of OTA traffic

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.99.1</td>
<td>52.58.219.193</td>
<td>UDP</td>
<td>121</td>
<td>57991 → 8888 Len=93</td>
</tr>
<tr>
<td>52.58.219.193</td>
<td>192.168.99.1</td>
<td>UDP</td>
<td>134</td>
<td>8888 → 57991 Len=106</td>
</tr>
<tr>
<td>52.58.219.193</td>
<td>192.168.99.1</td>
<td>UDP</td>
<td>134</td>
<td>8888 → 57991 Len=106</td>
</tr>
<tr>
<td>52.58.219.193</td>
<td>192.168.99.1</td>
<td>UDP</td>
<td>134</td>
<td>8888 → 57991 Len=106</td>
</tr>
<tr>
<td>192.168.99.1</td>
<td>1.1.1.1</td>
<td>DNS</td>
<td>63</td>
<td>Standard query 0x0001 A erom.niucache.com</td>
</tr>
<tr>
<td>1.1.1.1</td>
<td>192.168.99.1</td>
<td>DNS</td>
<td>79</td>
<td>Standard query response 0x0001 A erom.niucache.com A 60...</td>
</tr>
<tr>
<td>192.168.99.1</td>
<td>60.205.12.173</td>
<td>TCP</td>
<td>64</td>
<td>58304 → 80 [SYN] Seq=0 Win=13600 Len=0 MSS=1360 WS=1 SAC...</td>
</tr>
<tr>
<td>60.205.12.173</td>
<td>192.168.99.1</td>
<td>TCP</td>
<td>52</td>
<td>80 → 58304 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=14...</td>
</tr>
</tbody>
</table>
| 60.205.12.173      | 192.168.99.1        | TCP      | 52     | [TCP Retransmission] 80 → 58304 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=14...
| 192.168.99.1       | 60.205.12.173       | TCP      | 64     | [TCP Spurious Retransmission] 58304 → 80 [SYN] Seq=0 Win=14600 Len=0 MSS=14...
| 60.205.12.173      | 192.168.99.1        | TCP      | 52     | [TCP Previous segment not captured] [TCP Port numbers reused] 58304 → 80 MSS=14...
| 60.205.12.173      | 192.168.99.1        | TCP      | 52     | [TCP Retransmission] [TCP Port numbers reused] 80 → 58304 MSS=14...
| 192.168.99.1       | 1.1.1.1             | DNS      | 63     | Standard query 0x0002 A erom.niucache.com                           |
| 1.1.1.1            | 192.168.99.1        | DNS      | 79     | Standard query response 0x0002 A erom.niucache.com A 60...           |
| 192.168.99.1       | 60.205.12.173       | TCP      | 40     | 58304 → 80 [RST] Seq=1 Win=0 Len=0                                  |
| 192.168.99.1       | 60.205.12.173       | TCP      | 40     | 58304 → 80 [RST] Seq=1 Win=0 Len=0                                  |
| 192.168.99.1       | 60.205.12.173       | TCP      | 40     | 58304 → 80 [RST] Seq=1 Win=0 Len=0                                  |
| 192.168.99.1       | 60.205.12.173       | TCP      | 40     | 58304 → 80 [RST] Seq=1 Win=0 Len=0                                  |
| 192.168.99.1       | 60.205.12.173       | TCP      | 64     | 64200 → 80 [SYN] Seq=0 Win=13600 Len=0 MSS=1360 WS=1 SAC...           |
| 60.205.12.173      | 192.168.99.1        | TCP      | 52     | 80 → 64200 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=14...           |
| 192.168.99.1       | 60.205.12.173       | TCP      | 40     | 64200 → 80 [ACK] Seq=1 Ack=1 Win=13600 Len=0                         |
| 192.168.99.1       | 60.205.12.173       | HTTP     | 154    | GET /rom/NISP/V1.0/TRA01C10ECP001.bin?sn=NAS                         |
| 60.205.12.173      | 192.168.99.1        | TCP      | 40     | 80 → 64200 [ACK] Seq=1 Ack=115 Win=14720 Len=0                       |
| 60.205.12.173      | 192.168.99.1        | HTTP     | 1216   | HTTP/1.1 200 OK (application/octet-stream)                          |
| 60.205.12.173      | 192.168.99.1        | TCP      | 40     | 80 → 64200 [FIN, ACK] Seq=1177 Ack=115 Win=14720 Len=0              |
OTA traffic continued

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.99.1</td>
<td>1.1.1.1</td>
<td>DNS</td>
<td>63</td>
<td>Standard query 0x0003 A erom.niuarchive.com</td>
</tr>
<tr>
<td>1.1.1.1</td>
<td>192.168.99.1</td>
<td>DNS</td>
<td>79</td>
<td>Standard query response 0x0003 A erom.niuarchive.com A 60...</td>
</tr>
<tr>
<td>192.168.99.1</td>
<td>1.1.1.1</td>
<td>DNS</td>
<td>63</td>
<td>Standard query 0x0004 A erom.niuarchive.com</td>
</tr>
<tr>
<td>1.1.1.1</td>
<td>192.168.99.1</td>
<td>DNS</td>
<td>79</td>
<td>Standard query response 0x0004 A erom.niuarchive.com A 60...</td>
</tr>
<tr>
<td>192.168.99.1</td>
<td>1.1.1.1</td>
<td>DNS</td>
<td>63</td>
<td>Standard query 0x0004 A erom.niuarchive.com</td>
</tr>
<tr>
<td>1.1.1.1</td>
<td>192.168.99.1</td>
<td>DNS</td>
<td>79</td>
<td>Standard query response 0x0004 A erom.niuarchive.com A 60...</td>
</tr>
<tr>
<td>192.168.99.1</td>
<td>1.1.1.1</td>
<td>ICMP</td>
<td>64</td>
<td>Destination unreachable (Port unreachable)</td>
</tr>
<tr>
<td>192.168.99.1</td>
<td>1.1.1.1</td>
<td>ICMP</td>
<td>64</td>
<td>Destination unreachable (Port unreachable)</td>
</tr>
<tr>
<td>192.168.99.1</td>
<td>1.1.1.1</td>
<td>ICMP</td>
<td>64</td>
<td>Destination unreachable (Port unreachable)</td>
</tr>
<tr>
<td>192.168.99.1</td>
<td>1.1.1.1</td>
<td>ICMP</td>
<td>64</td>
<td>Destination unreachable (Port unreachable)</td>
</tr>
<tr>
<td>192.168.99.1</td>
<td>1.1.1.1</td>
<td>ICMP</td>
<td>64</td>
<td>Destination unreachable (Port unreachable)</td>
</tr>
<tr>
<td>192.168.99.1</td>
<td>1.1.1.1</td>
<td>ICMP</td>
<td>64</td>
<td>Destination unreachable (Port unreachable)</td>
</tr>
<tr>
<td>192.168.99.1</td>
<td>60.205.12.173</td>
<td>TCP</td>
<td>64</td>
<td>57548 → 80 [SYN] Seq=0 Win=13600 Len=0 MSS=13600 WS=1 SAC...</td>
</tr>
<tr>
<td>192.168.99.1</td>
<td>1.1.1.1</td>
<td>ICMP</td>
<td>64</td>
<td>Destination unreachable (Port unreachable)</td>
</tr>
<tr>
<td>60.205.12.173</td>
<td>192.168.99.1</td>
<td>TCP</td>
<td>52</td>
<td>80 → 57548 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=14...</td>
</tr>
<tr>
<td>192.168.99.1</td>
<td>60.205.12.173</td>
<td>TCP</td>
<td>40</td>
<td>57548 → 80 [ACK] Seq=1 Ack=1 Win=13600 Len=0</td>
</tr>
<tr>
<td>192.168.99.1</td>
<td>60.205.12.173</td>
<td>HTTP</td>
<td>154</td>
<td>GET /rom/N1SP/V1.0/TRA011C10E2P002.bin?sn=NAS...</td>
</tr>
<tr>
<td>60.205.12.173</td>
<td>192.168.99.1</td>
<td>TCP</td>
<td>40</td>
<td>80 → 57548 [ACK] Seq=1 Ack=115 Win=14720 Len=0</td>
</tr>
<tr>
<td>60.205.12.173</td>
<td>192.168.99.1</td>
<td>HTTP</td>
<td>1216</td>
<td>HTTP/1.1 200 OK (application/octet-stream)</td>
</tr>
<tr>
<td>60.205.12.173</td>
<td>192.168.99.1</td>
<td>TCP</td>
<td>40</td>
<td>80 → 57548 [FIN, ACK] Seq=1177 Ack=115 Win=14720 Len=0</td>
</tr>
<tr>
<td>192.168.99.1</td>
<td>60.205.12.173</td>
<td>TCP</td>
<td>40</td>
<td>57548 → 80 [ACK] Seq=115 Ack=1178 Win=13600 Len=0</td>
</tr>
<tr>
<td>192.168.99.1</td>
<td>60.205.12.173</td>
<td>TCP</td>
<td>40</td>
<td>57548 → 80 [FIN, ACK] Seq=115 Ack=1178 Win=13600 Len=0</td>
</tr>
<tr>
<td>60.205.12.173</td>
<td>192.168.99.1</td>
<td>TCP</td>
<td>40</td>
<td>80 → 57548 [ACK] Seq=1178 Ack=116 Win=14720 Len=0</td>
</tr>
</tbody>
</table>
OTA Download

• Download via plain HTTP in 1kB chunks from:
  http://erom.niucache.com/rom/N1SP/V1.0/
  TRA01C10ECP001.bin?sn=NASxxxxxxxxxxxxxxxx
  http://erom.niucache.com/rom/N1SP/V1.0/
  TRA01C10ECP002.bin?sn=NASxxxxxxxxxxxxxxxx
  ...

• Vehicle SN as query parameter, however turns out you can pass whatever you want 😜

• New connection for every chunk

• In my dump I could see chunks being re-transferred, guess my BTS hardware isn’t the most reliable 😳
OTA Download

- To download the firmware you basically need to know the size and then you can do something like (bash):

```bash
$ for I in {1..42}; do curl http://erom.niuucache.com/rom/N1SP/V1.0/
  TRA01C10ECP`printf %03d $I`.bin?sn=blah >
  TRA01C10ECP`printf %03d $I`; done

$ cat TRA01C10ECP0* > firmwareTRA01C10ECP.bin
```
OTA Firmware

- Seems encrypted. No obvious header, high entropy, no strings…¯\_(ツ)_/¯
Meanwhile: Update finished?

• Almost there…

• App reported an error, saying to try again

• But the scooter seems fine. After closing the app it was actually shown as being up-to date.
OTA Risks?

• The vehicle gateway sends update trigger packet(s) to the ECU

• In theory, the vendor could trigger an update at any time

• However if the ignition is on it won’t start until you turn off the ignition and turn it back on
Firmware hackable?

• Possibly, but need to understand the firmware first

• Also the update trigger packet will probably contain information about the update package and size so the ECU knows what to download

• But… I want to make my Scooter faster!
Behold! There’s a solution

- **Source:** [http://www.myniu.org/making-the-nls-faster/](http://www.myniu.org/making-the-nls-faster/)
  
- By adding a 2nd controller that drives the motor while the original controller talks to the system 😎

This is probably illegal in most countries. Don’t do it.
China shopping list ++

[Image of electronic components]
So. Back to replaying…

• What could we actually replay to see if it works properly?

• Remember, the App has push notifications :)

• For some reason, the 'unusual movement' detection hasn’t been working for a while

• But every time you unplug the battery, the app shows a notification

• Let’s unplug the battery, dump the packet, and replay
Conclusions

• Overall, the vendor did a really good job!

• Pretty solid implementation, safety checks etc.

• It has some small issues, like missing certificate pinning, but that’s minor

• (Most likely) Encrypted firmware

• Encrypted(?) packet format for GPRS connection though vulnerable to replaying
Thanks!
谢谢!