

利用Hackrf One进行GPS定位欺骗制作超级跑马机

GPS (/blog/tag/metorm/GPS)

HackRF (/blog/tag/metorm/HackRF)

© 2019-05-28 10:38:28

👁 0 🗨 0 💬 0

0×00 驾校的困惑

现行规定要求每个学员都必须在驾校练习够规定的学时，才能参加考试,在每台教练车上都安装有计时计程终端，学员刷卡刷指纹后开始累计里程。但是目前中国的很多驾校，存在车少人多的情况，假设有 300 个学员，但是只有 20 台车，每个学员要跑够 50km，而学员都很忙，只能业余时间练习，这种情况下是完全不可能满足要求的。

于是，跑马机应运而生，它通过伪造 GPS 轨迹，欺骗车载计程终端,达到刷学时数的目的。然而，受限于其硬件，使用效果非常不好，每个跑马机只能安装于一个终端，并且速度很不稳定，时快时慢，可能会长时间静止，也可能从飙车速度突然降到步行速度.使用跑马机的驾校经常会出现如下的场面：



(<http://image.3001.net/images/20170205/14862942459172.png>)

因为跑马机很不可靠，所以对于实在刷不够的学时，教练只能开着车去高速公路兜一圈，在油价飞涨的今天，每兜一圈驾校老板的心都要滴一次血。

既然跑马机的原理是伪造 GPS 信号，那么我想起了那个被我用来折腾过所有能接触到的遥控门，遥控车，遥控飞机之后，留在角落里吃土很久的 hackrf one。

Hackrf one 可以稳定的伪造 GPS 信号，并且不需要安装在接收设备上，只要是在发射范围内的接收机，都可能被伪造的信号欺骗，所以把它当做“超级跑马机”使用，理论上是完全可以的。想到了这一点，我一言不合就开始干。

需要的设备：

笔记本电脑一台,配置随意,最好安装 linux 系统;hackrf one 一块(带 TCXO 时钟模块和天线);连接数据线一条.

然后安装好 hackrf 所需的驱动和 hackrf-tools,具体安装方法这里不做介绍,网上有很多.

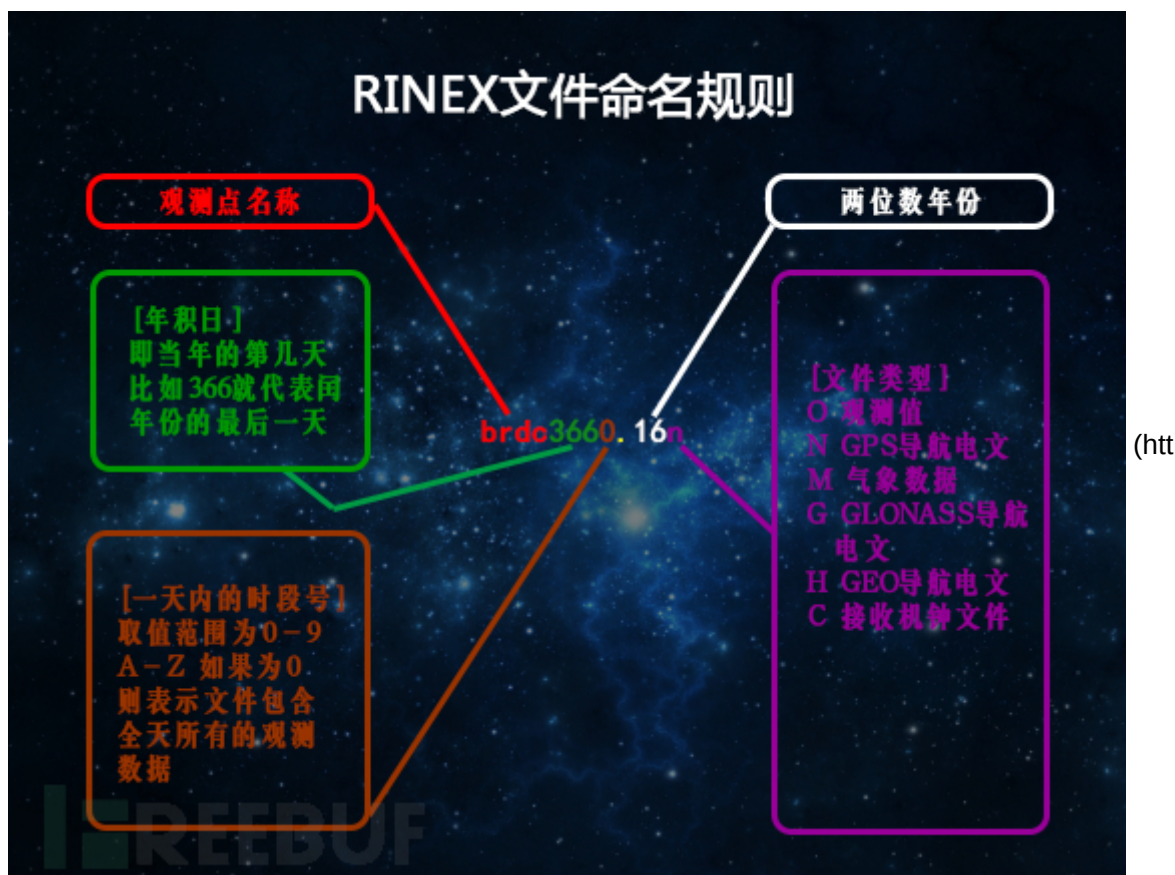
用到的软件：

gps-sdr-sim;Gnuradio;hackrf-tools;SatGen Trajectory Generation;Google Earth

具体流程: 获取GPS导航电文->制作运动轨迹文件->生成采样数据->发射模拟GPS 信号.

0x01 GPS 导航电文

要伪造 GPS 信号,首先需要获取 GPS 卫星的观测数据,在早期,各个接收机厂商的数据格式各不相同,现在已经普遍采用统一的 RINEX 格式, RINEX 本质上是文本文件, 其数据记录与接收机的制造厂商和具体型号无关,可以做到全通用,目前使用最广泛的是其第 2 版格式,RINEX 文件遵循标准的命名规则,具体如下图所示:



p://image.3001.net/images/20170205/14862942759661.png)

以上每种文件类型,其数据格式均不相同,这里我们只需要使用 GPS 导航电文,用文本编辑器打开后,其具体内容和格式说明如下:



(<http://image.3001.net/images/20170205/14862942906515.png>)

那么问题来了,我们从哪里获取 RINEX 格式的 GPS 导航电文呢?

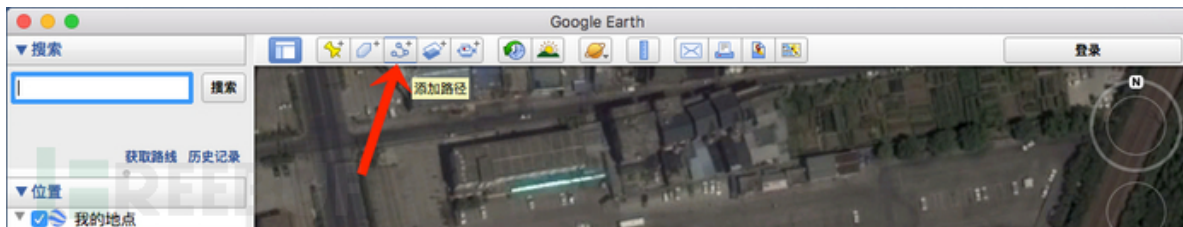
答案是找 NASA.

打开这个地址 <ftp://cddis.gsfc.nasa.gov/pub/gps/data/daily>, 里面是多个以年份命名的文件夹, 在里面的 brdc 子目录中, 按照上面的命名规则, 找到并下载相应年积日对应的 GPS 导航电文文件, 比如 2016 年 12 月 31 日的数据, 文件名是 brdc3660.16n.z, 下载然后解压缩, 得到的 brdc3660.16n 就是导航电文文件.

0x02 运动坐标轨迹文件

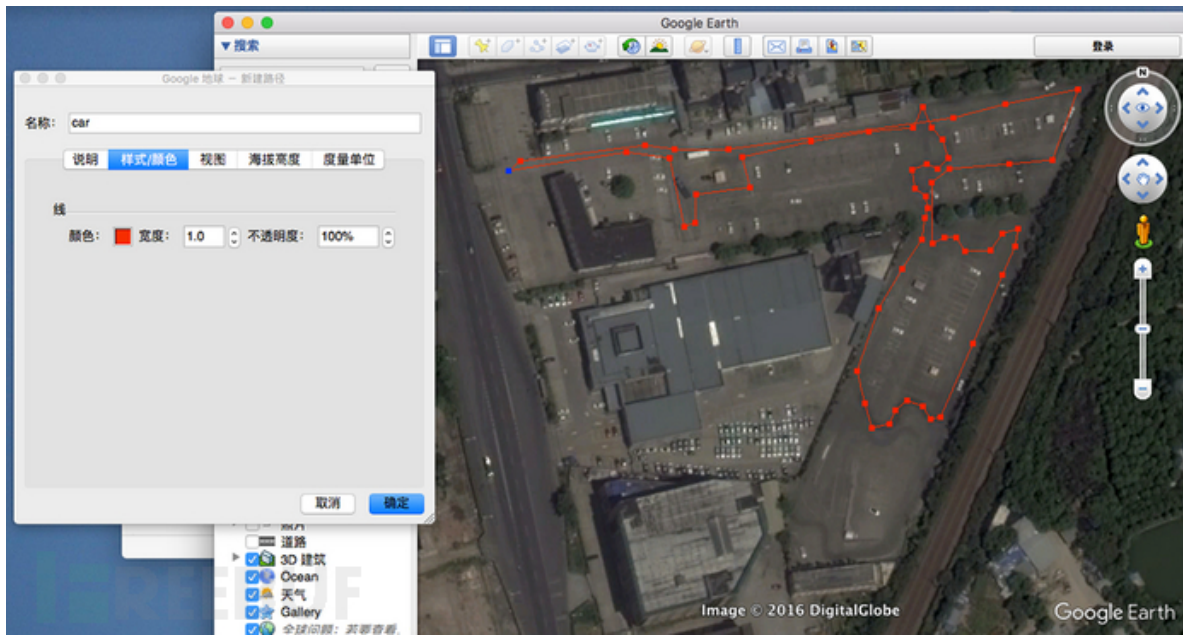
导航电文有了, 我们就有了 GPS 信号的数据资源, 接下来我们还需要具体的运动轨迹坐标, 事实上, 现在很多的驾校都是有电子围栏的, 也就是说, 教练车的运动轨迹随时都在监控之中, 其坐标必须在驾校的活动场地之内才能算学时, 所以我们要伪造的运动轨迹, 也必须使用真实的所在驾校的坐标, 这里我们需要利用 Google Earth.

首先启动 Google Earth, 调整视野, 寻找到驾校所在地区, 然后寻找驾校的位置, 让驾校的练习场地完全出现在视野中, 然后点击工具栏上的添加路径:



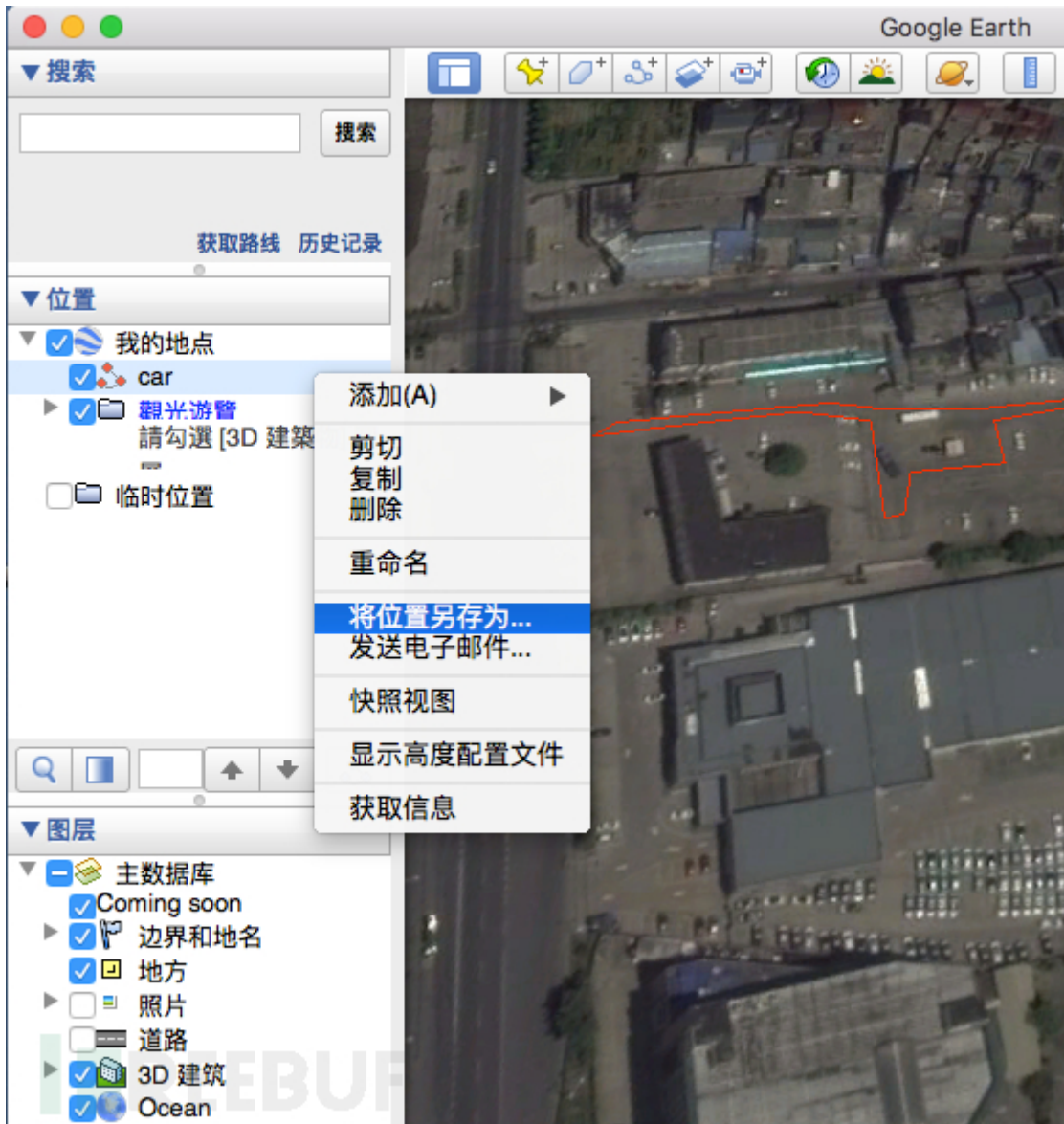
(<http://image.3001.net/images/20170205/14862943255632.png>)

在地图上, 沿着驾校的练习场地, 点击勾画出一个运动轨迹, 尽可能的符合正常行车的路线:



(<http://image.3001.net/images/20170205/14862943386322.png>)

为路径起名,点击确定后,它就会出现在左侧的位置列表之中,在上面点击右键,选择“将位置另存为”,将之保存为 kml 文件:



(htt

p://image.3001.net/images/20170205/14862943921060.png)

保存后的 kml 文件,是不能够直接使用的,它本质上只是一种包含路径中关键坐标点的格式,并不包含实际运动轨迹中加速,减速,静止这样的数据,我们要对其进行转换。

在本案例中,我们需要将 kml 文件转换成两种格式的轨迹数据文件(两种中任一种),下面分别做介绍:

1. NMEA 格式文件

NMEA 是 GPS 导航设备统一的 RTCM 标准协议,具体内容是一行行的描述坐标,海拔及时间轨迹的语句,主要有 GPGGA、GPGSA、GPGSV、GPRMC、GPVTG、GPGLL 等语句格式,这里我们只需要 GPGGA 格式,其内容如下:


```

$GPGGA,090000.00,3008.26136832,N,12016.99884601,E,1,05,2.87,160.00,M,-21.3213,M,,*72
$GPGGA,090000.10,3008.26204723,N,12017.01257578,E,1,05,2.87,160.00,M,-21.3213,M,,*77
$GPGGA,090000.20,3008.25583304,N,12017.02344025,E,1,05,2.87,160.00,M,-21.3213,M,,*73
$GPGGA,090000.30,3008.25063299,N,12017.02727863,E,1,05,2.87,160.00,M,-21.3213,M,,*77
$GPGGA,090000.40,3008.25577676,N,12017.03872475,E,1,05,2.87,160.00,M,-21.3213,M,,*70
$GPGGA,090000.50,3008.26321128,N,12017.04920731,E,1,05,2.87,160.00,M,-21.3213,M,,*79
$GPGGA,090000.60,3008.26623402,N,12017.06252747,E,1,05,2.87,160.00,M,-21.3213,M,,*7D
$GPGGA,090000.70,3008.26825271,N,12017.07615139,E,1,05,2.87,160.00,M,-21.3213,M,,*7F
* . . . . .

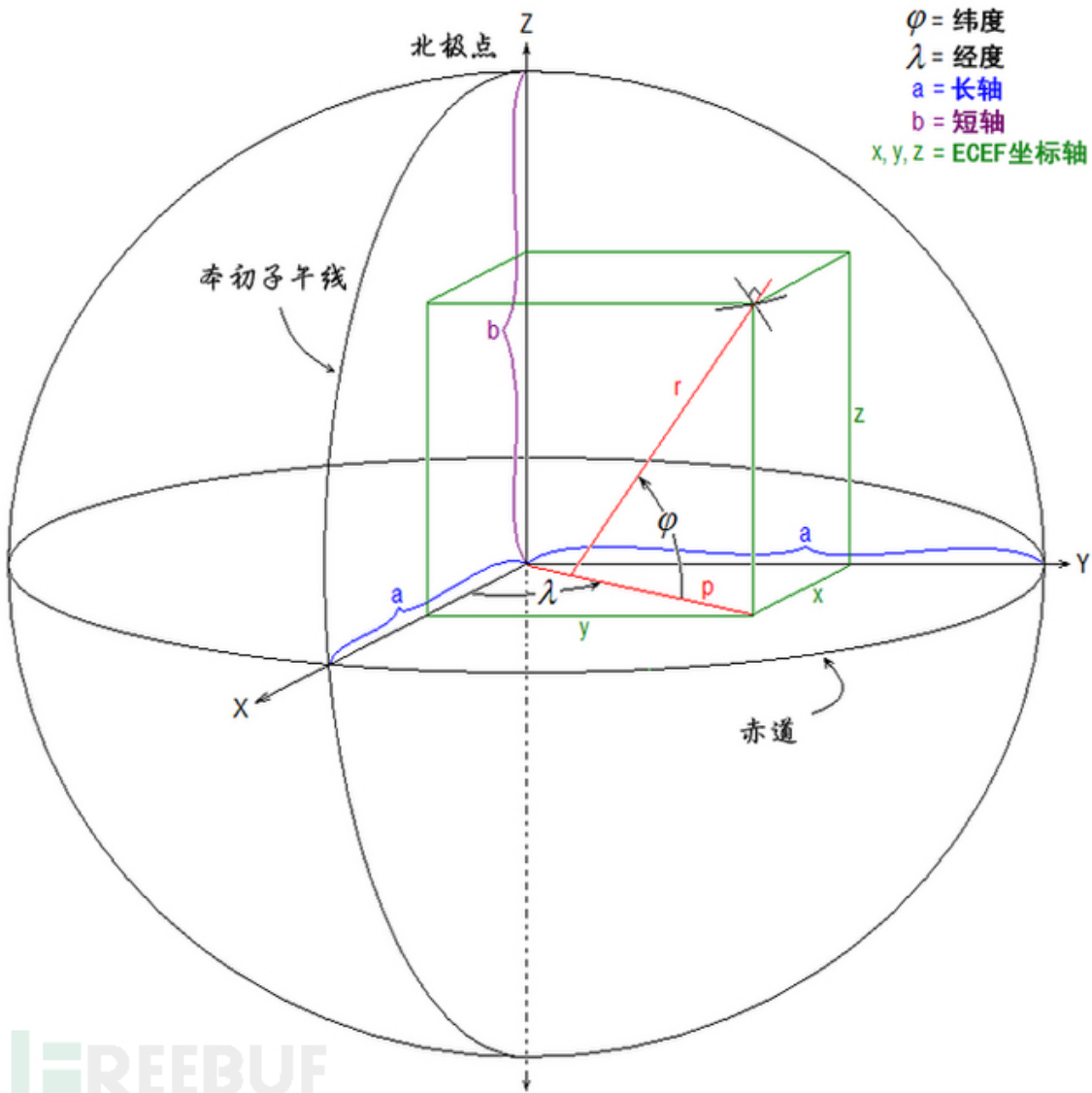
```

- UTC时间 格式为时分秒
- 纬度ddmm. mmmm度分格式 N为北纬 S为南纬
- 经度dddmm. mmmm度分格式 E为东经 W为西经
- GPS状态: 0=未定位 1=非差分定位 2=差分定位 6=正在估算
- 正在使用解算位置的卫星数量
- HDOP水平精度因子
- 海拔高度 单位M
- 地球椭球面相对大地水准面的高度 单位M
- 校验值

(<http://image.3001.net/images/20170205/14862944073455.png>)

2. ECEF 坐标轨迹文件

ECEF 坐标系又叫地心地固坐标系,它是一个笛卡尔右手坐标系,用 xyz 三个坐标轴来描述地表的某特定位置,具体如下图所示:



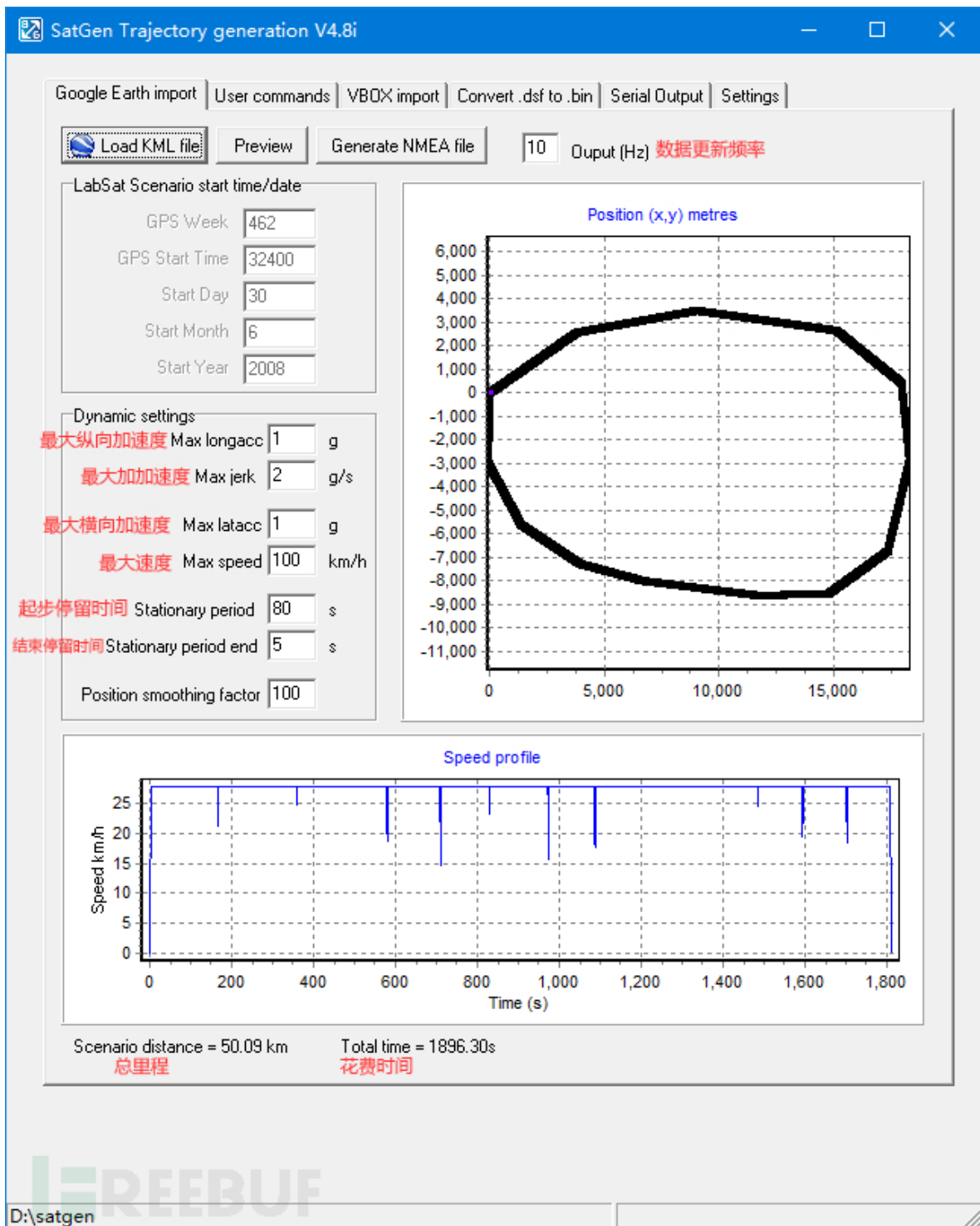
(<http://image.3001.net/images/20170205/14862944305815.png>)

ECEF 的原点与地球质心重合,x 轴从原点出发,延伸至本初子午线与赤道的交点,z轴延伸至北极点,y 轴遵循右手坐标系与 x 和 z 构成的平面垂直。

ECEF 的轨迹文件(csv 格式),格式很简单:时间(秒),x 坐标,y 坐标,z 坐标,示例如下:

```
0.0,-3813477.954, 3554276.552, 3662785.237
0.1,-3813477.599, 3554276.226, 3662785.918
0.2,-3813477.240, 3554275.906, 3662786.598
0.3,-3813476.876, 3554275.590, 3662787.278
0.4,-3813476.508, 3554275.280, 3662787.958
0.5,-3813476.135, 3554274.975, 3662788.638
0.6,-3813475.757, 3554274.675, 3662789.318
0.7,-3813475.375, 3554274.381, 3662789.997.....
```

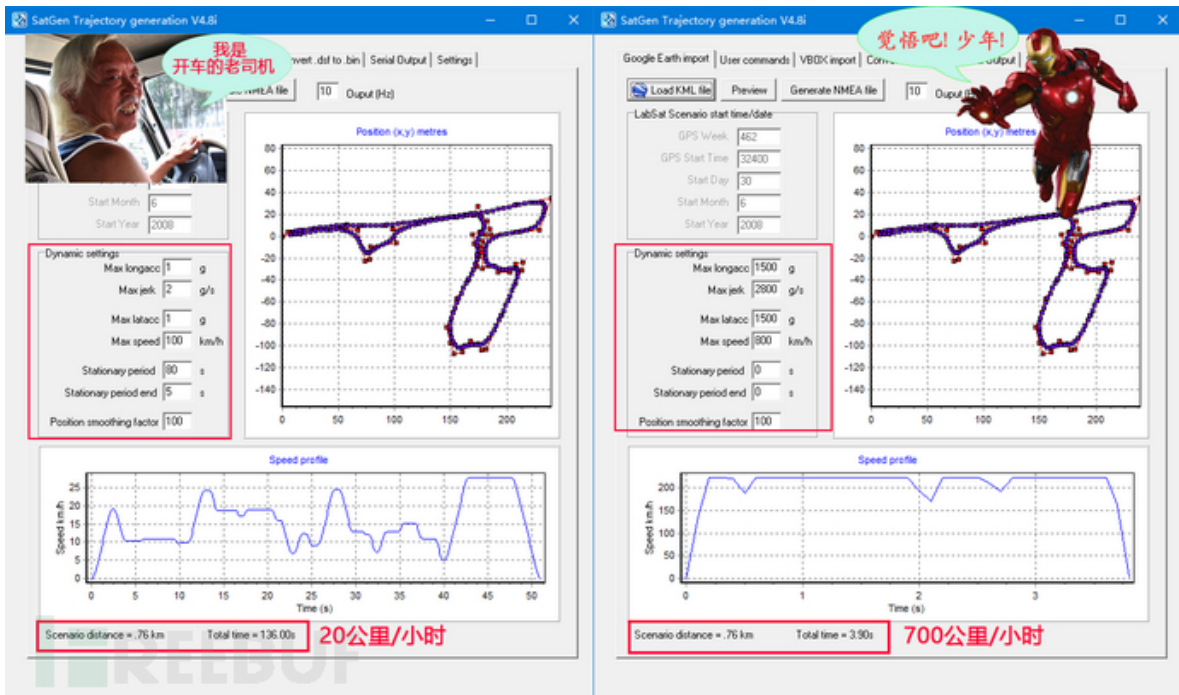
格式介绍完了,接下来,我们把 kml 文件转换成 NMEA 格式,这里我们需要用到一个小工具,它由 LabSat 提供,是 SatGen 软件的一个极简免费版(下载地址见文末):



(<http://image.3001.net/images/20170205/14862944627141.png>)

将频率设置为 10hz,然后点击“Load KML file”,就会自动加载 kml 路径,然后点击“Generate NMEA file”,即可生成标准的 NMEA 轨迹文件。

注意此软件会模拟真实的运动情况,比如拐弯会减速,起步和结束时会有停留,这些由横向加速度、纵向加速度、加加速度、最大速度、起步停留时间、结束停留时间这几个参数控制,默认设置是不适合于我们的。这里我们举个例子,一辆由老司机驾驶的汽车,和钢铁侠进行对比:



(<http://image.3001.net/images/20170205/14862944947502.png>)

很明显，在越复杂的运动轨迹下，参数设置的越接近实际物理情况，平均速度就越慢，要想在驾校稳定保持 100km/h 以上的飙车速度，我们必须增大三个加速度和最大速度的值，然后将起步和结束停留时间设置为 0 才可以。当然，如果驾校没有电子围栏，我们完全可以生成一个包含超长纯直线路径的 kml 文件，这样使用默认设置转换成 NMEA 后，仍然可以达到 100km/h 的稳定匀速。

0x03 生成采样数据文件

利用上面得到的 GPS 导航电文和 NMEA 文件，我们可以使用开源项目 `gps-sdr-sim`，生成一个采样数据文件，该文件会作为我们后面实际工作时的数据源。首先下载 `gps-sdr-sim` 的源码：

1. `$ git clone https://github.com/osqzss/gps-sdr-sim.git`
2. `$ cd gps-sdr-sim`

然后编译运行：

1. `$ gcc gpssim.c -lm -O3 -o gps-sdr-sim`

最后按以下参数执行：

[NMEA 轨迹]


1. `./gps-sdr-sim -e <导航电文文件> -g <轨迹文件> -b 8[ECEF 轨迹] ./gps-sdr-sim -e <导航电文文件> -u <轨迹文件> -b 8`

如果不事先指定名称，默认在程序所在目录生成名为 `gpssim.bin` 的数据文件。关于 ECEF 轨迹文件，`gps-sdr-sim` 在 `satgen` 目录中提供了一个小工具，可以把 NMEA 格式转换为 ECEF：

1. `$ cd satgen$ gcc nmea2um.c -o nmea2um`

然后执行 `nmea2um <源文件> <目标文件>` 就可以了。事实上，查阅源码会发现，在实际使用中，`gps-sdr-sim` 也是把 NMEA 转换成 ECEF 使用的。

为了更好的说明 `gps-sdr-sim` 各项参数的含义，我在编译前对 `Usage` 做了汉化：

 11.png (<http://image.3001.net/images/20170205/14862945313558.png>)

gps-sdr-sim 默认最大只能生成 300 秒的数据,需要在编译前,修改 gpssim.h 文件,找到这一行:

```
1. #define USER_MOTION_SIZE(3000)
```

注意这里的值是乘以十的,3000 代表 300 秒,将其修改为我们需要的时间再编译即可,修改的值越大,生成的文件体积越大,比如修改为 3000 秒后体积会达到将近15G 大小,不过相对于现在动辄 1t 的硬盘容量来说,这点体积并没有什么卵用。

需要特别注意的是,不管是哪种轨迹格式,实际使用中其持续时间受限于文件中的时间长度,即使按照前面的方法修改了 USER_MOTION_SIZE 的值,如果轨迹文件的时间长度小于该值,仍然会以轨迹文件为准。

为了让持续时间长一些,我们可以制作多个轨迹文件,然后进行合并,或者干脆图省事,用文本编辑器把轨迹文件的所有行全选,然后复制 N 遍 $\text{L}(\wedge\text{o}\wedge)\text{J}$

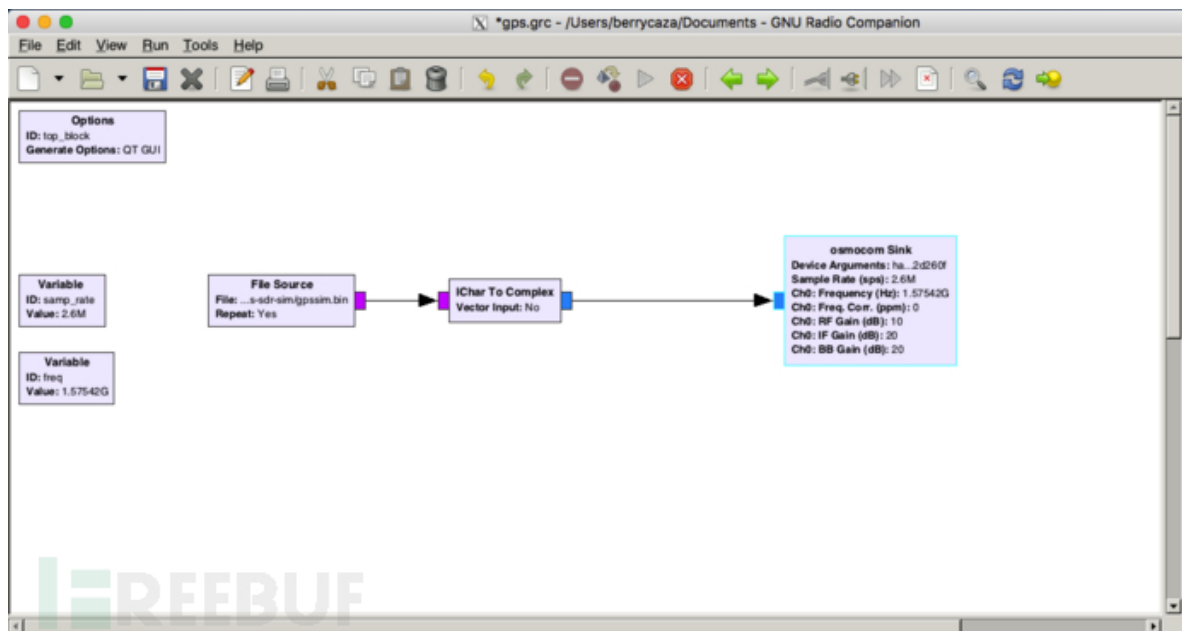
0x04 猥琐的行为终于开始了

现在,一切就绪,我们把生成的 gpssim.bin 文件,转换为信号发射出去,车载计程终端会被干扰,将其当做真实的 gps 信号,然后我们的目的就达到了,驾校的每一辆车都会以飙车速度”狂奔”.然后驾校老板笑逐颜开,然后我们会有银子赚,哇卡卡卡..... $\geq\leq$

这里特别说明,最初下载的 GPS 导航电文的年积日是哪一天,开始发射信号后,接收终端的时间就会变成相应的日期,如果需要准确日期,请下载当天的导航电文或者在生成采样数据时,使用-T 参数强制指定日期和时间.发射信号具体的实现,有以下三种方法:

1. 驾校老板只想傻瓜化:

安装 gnuradio ,按照下图搭个流程图,运行即可:



(<http://image.3001.net/images/20170205/148629458714.png>)

以上流程图中,采样速率使用 2600000,发射频率 1575420000hz 即是 GPS 民用L1 频段的频率.File Source 模块使用的是我们生成的 gpssim.bin 文件.

2. 驾校老板想用命令行装逼,这样看起来比较高大上:

使用 hackrf-tools 自带的 hackrf_transfer,执行如下命令:

```
1. $ hackrf_transfer -t gpssim.bin -f 1575420000 -s 2600000 -a 1 -x 0  
-R
```

3. 驾校老板非常厉害,会编程(好像实际中也不大可能-_-|||)

在 gps-sdr-sim 的 player 子目录中,有段源码 hackplayer.c,可以直接调用 libhackrf 发射信号,不过奇怪的是它居然是 Windows 代码,但移植到 Linux 也很简单,稍作改动即可,这里我直接提供改好的代码下载(地址见文末),然后编译,运行:

```
1. $ gcc hackplayer.c -lhackrf -O3 -o hackplayer$ ./hackplayer gpssim  
.bin
```

以上三种方法,效果完全一样,不同的只是逼格,下面是实际测试的效果,非常好,一台 hackrf 可以同时为很多个终端提供服务,并且速度基本稳定维持在预先定义的100km/h:

 13.png (<http://image.3001.net/images/20170205/14862946124114.png>)



(<http://image.3001.net/images/20170205/14862946257668.png>)

当然,如果老板不怕考试中心的交警叔叔来找麻烦,或者老板和交警叔叔是好基友,那么你预先设置成钢铁侠速度甚至火箭速度也是可以的... --

0x05 补充

我们还可以进一步节约成本,用安装好 Linux 的 arm 开发板来替代电脑,这样成本只需几百块钱,这里我使用 Cubietruck,系统为 Debian 7,Debian 源里已经有了 armhf 平台的 hackrf,直接 apt-get 安装,然后把生成好的 gpssim.bin 拷贝进去(需要注意,此类开发板功率比较低,同时接硬盘和 hackrf

带不起来,而 gpssim.bin 体积比较大,所以只能使用大容量 tf 卡来存放),写个两行脚本(gpssim.bin 15的路径写成实际的绝对路径):

```
1. #!/bin/bashhackrf_transfer -t gpssim.bin -f 1575420000 -s 260000  
0 -a 1 -x 0
```

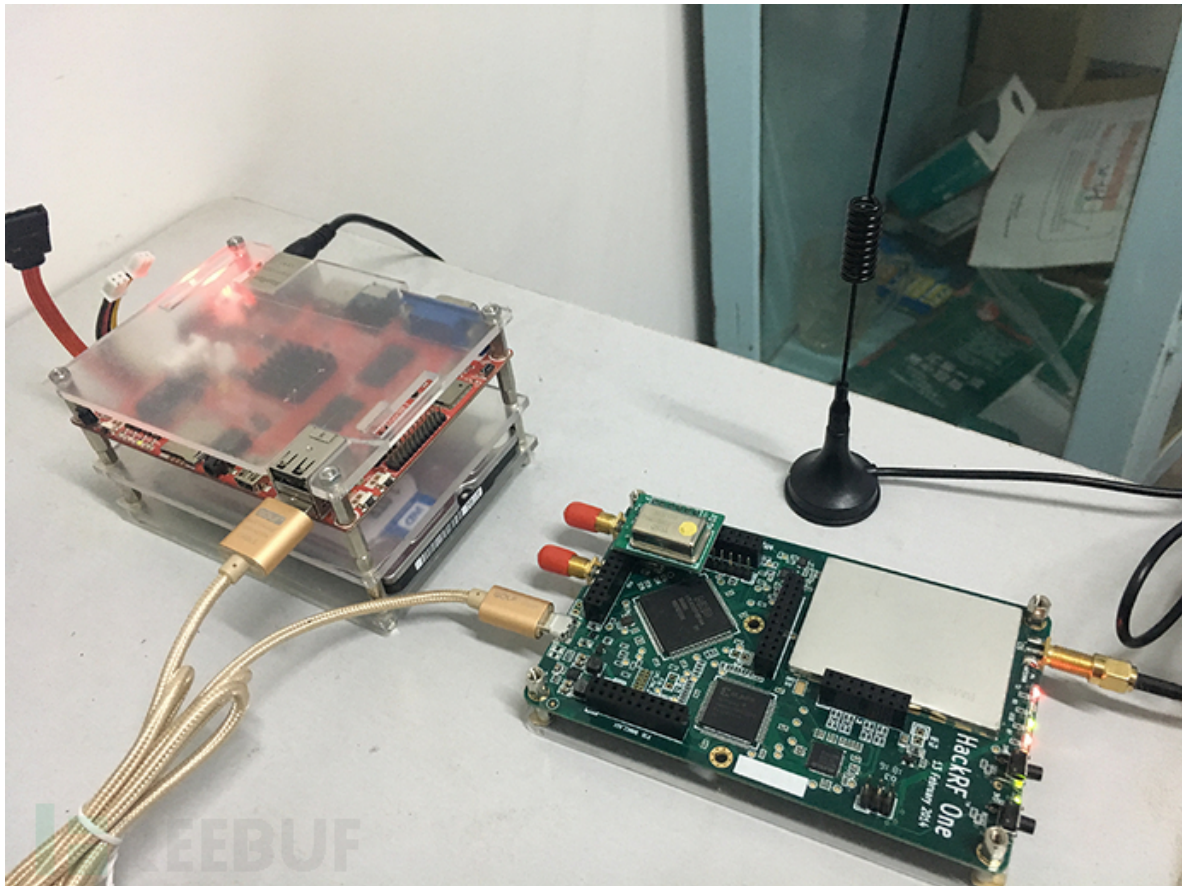
保存成 start.sh,添加权限:

```
1. $ sudo chmod 777 start.sh
```

然后在/etc/rc.local 加入一行(start.sh 的路径写成实际的绝对路径):

```
1. sh start.sh
```

用 usb 线接好开发板和 hackrf one,这样开发板在启动后就会自动开启 hackrfone 的信号发射:



(<http://image.3001.net/images/20170205/14862946389336.png>)

最后,我必须严肃的,负责的,不误人子弟的讲清楚,即便因为人多车少或者自己没有时间练习亦或是其他原因,用了这样的方法在驾校刷够了学时,拿到了驾照,但是仍然应该自己想办法,以其他方式进行多次练习直至熟练才能真正上路,毕竟,有无数血的教训证明,靠作弊得来的驾照,上路真的危险重重.....

转载自: <http://www.freebuf.com/geek/126236.html> (<http://www.freebuf.com/geek/126236.html>)

Pre: No Post

Next: CMake生成Visual Studio Nmake文件时报错问题分析 (/blog/post/metorm/CMake%E7%94%9F%E6%88%90Visual-Studio-Nmake%E6%96%87%E4%BB%B6%E6%97%B6%E6%8A%A5%E9%94%99%E9%97%AE%E9%A2%98%E5%88%86%E6%9E%90)

👍 0 likes

👁 1

🐦 Weibo

👤 Wechat

...



(/blog/metorm)

船长的日志 (/blog/metorm)

非典型导弹设计师

分类

导航

[主页 \(/blog/metorm\)](/blog/metorm)

[关于 \(/blog/single/metorm/About-Me\)](/blog/single/metorm/About-Me)

[归档 \(/blog/archives/metorm\)](/blog/archives/metorm)

[标签 \(/blog/tags/metorm\)](/blog/tags/metorm)

友情链接



search

Theme by roomcar (<http://weibo.com/1399064863>)

Proudly powered by Leanote (<https://leanote.com>)