

精译版

连网汽车被盗风险研究

非官方中文译文·安天技术公益翻译组 译注

文档信息			
原文名称	Mobile apps and stealing a connected car		
原文作者	Mikhail Kuzin , Victor Chebyshev	原文发布日期	2017 年 2 月 16 日
作者简介	Victor Chebyshev 是卡巴斯基实验室研究开发团队的主管。 https://www.linkedin.com/in/victor-chebyshev-1b48bb8b/		
原文发布单位	卡巴斯基实验室		
原文出处	https://securelist.com/analysis/publications/77576/mobile-apps-and-stealing-a-connected-car/		
译者	安天技术公益翻译组	校对者	安天技术公益翻译组
免责声明	<ul style="list-style-type: none"> 本译文译者为安天实验室工程师，本文系出自个人兴趣在业余时间所译，本文原文来自互联网的公共方式，译者力图忠于所获得之电子版本进行翻译，但受翻译水平和技术水平所限，不能完全保证译文完全与原文含义一致，同时对所获得原文是否存在臆造、或者是否与其原始版本一致未进行可靠性验证和评价。 本译文对应原文所有观点亦不受本译文中任何打字、排版、印刷或翻译错误的影响。译者与安天实验室不对译文及原文中包含或引用的信息的真实性、准确性、可靠性、或完整性提供任何明示或暗示的保证。译者与安天实验室亦对原文和译文的任何内容不承担任何责任。翻译本文的行为不代表译者和安天实验室对原文立场持有任何立场和态度。 译者与安天实验室均与原作者与原始发布者没有联系，亦未获得相关的版权授权，鉴于译者及安天实验室出于学习参考之目的翻译本文，而无出版、发售译文等任何商业利益意图，因此亦不对任何可能因此导致的版权问题承担责任。 本文为安天内部参考文献，主要用于安天实验室内部进行外语和技术学习使用，亦向中国大陆境内的网络安全领域的研究人士进行有限分享。望尊重译者的劳动和意愿，不得以任何方式修改本译文。译者和安天实验室并未授权任何人士和第三方二次分享本译文，因此第三方对本译文的全部或者部分所做的分享、传播、报道、张贴行为，及所带来的后果与译者和安天实验室无关。本译文亦不得用于任何商业目的，基于上述问题产生的法律责任，译者与安天实验室一律不予承担。 		

连网汽车被盗风险研究

Mikhail Kuzin , Victor Chebyshev

2017 年 2 月 16 日

最近几年，连网汽车逐渐流行起来。它不仅配有多媒体系统（豪华汽车可以听音乐，查看地图，观看电影），还有车钥匙系统（包括传统车钥匙和汽车控制程序）。通过专用的手机应用程序，可以获取汽车的 GPS 坐标，追踪其路线，打开车门，发动引擎和打开辅助设备。一方面，这些功能绝对是有用的，已被数百万用户使用。另一方面，如果偷车贼能够访问受害者的手机（安装了汽车控制程序），盗窃汽车岂不是轻而易举吗？

要回答这一问题，我们需要先弄清窃贼能做什么，以及车主应如何避免这种问题。

潜在威胁

我们应注意，汽车控制应用程序非常流行——最受欢迎的应用程序的用户数量介于数万到数百万。例如，如下是一些应用程序的安装数量。



在试验中，我们选取了不同制造商中的几个汽车控制应用程序。我们不会公布应用程序的名称，但是我们已经将调查结果反馈给了制造商。

我们对每个应用程序的以下方面进行了审查：

- 潜在危险功能的可用性，这基本就意味着利用应用程序窃取汽车或者导致汽车系统无法正常工作可能性。
- 应用程序开发商是否采取了使逆向工程复杂化的措施（模糊处理或打包）。如果没有，窃贼很容易读取应用程序的代码，找到漏洞，利用漏洞控制汽车的基础设施。
- 应用程序是否会检查设备的 root 权限（一旦发现 root 权限被启用，就会取消安装）。毕竟，一旦恶意软件成功感染了启用 root 权限的设备，它几乎就能做任何事情了。这样的话，我们需要弄清开发商编程的用户凭证是否以明文形式保存在设备上，这很重要。
- 是否会验证显示给用户的是应用程序的 GUI（覆盖保护）。安卓允许监控显示给用户的程序，恶意软件会利用这一点，向用户显示带有完全相同的 GUI 的钓鱼窗口，从而窃取用户的凭证。
- 应用程序是否进行完整应用检查，即，是否对代码更改进行验证。这会影响罪犯向应用程序注入代码，然后在应用商店发布，同时保留原始应用程序的特征和功能。

不幸的是，审查发现所有的应用程序都或多或少地存在漏洞。

汽车应用程序测试

在此次研究中，我们选取了 7 款最受欢迎的汽车应用程序，然后测试它们是否存在可能会被罪犯利用的漏洞。

测试结果如下表所示。另外，我们对每个应用的安全功能进行了审查。

应用程序	功能	是否进行 代码模糊	是否采用未加密的用户名 和密码	是否采用 覆盖保护	是否启用 root 权 限	是否进行完整 性检查
1	开门	否	是 (用户名)	否	否	否
2	开门	否	是 (用户名和密码)	否	否	否
3	开门,启动引擎	否	-	否	否	否
4	开门	否	是 (用户名)	否	否	否
5	开门,启动引擎	否	是 (用户名)	否	否	否
6	开门,启动引擎	否	是 (用户名)	否	否	否
7	开门,启动引擎	否	是 (用户名和密码)	否	否	否

应用程序 1

整个汽车注册的基本过程是在应用程序中输入用户名和密码以及汽车 VIN (汽车识别码)。之后,按照传统的方法输入应用程序显示的 PIN (个人识别码),最后完成智能手机与汽车的连接。这意味着,仅知道 VIN 无法打开车门。

应用程序不检查设备是否启用 root 权限,然后把用户名与 VIN 以明文存储在账户.xml 文件中。一旦木马获得手机的超级用户权限,即可轻松窃取数据。

应用程序 1 很容易被反编译,之后犯罪分子就能读取和理解代码。除此之外,它不对自身重叠的 GUI 进行计数,这意味着一个只有 50 行代码的钓鱼应用程序能够获取用户名和密码。攻击者很容易检测哪个应用程序正在运行,如果该应用程序有目标软件包的名字,还可利用相似 GUI 发动恶意活动。

为了测试完整性检查,我们修改了凭证登录方法。

```
private void loginWithCredentials() {
    String v0 = this.mUsernameView.getText().toString();
    String v1 = this.mPasswordView.getText().toString();
    Toast.makeText(this.getApplicationContext(), v0 + v1, 0).show();
    this.performLogin(v0, v1);
}
```

在这种情况下,用户名和密码只在手机屏幕上显示,但无法阻止嵌入代码向罪犯服务器发送凭证。

完整性检查的缺失使得任何感兴趣的个人都可自行获取并修改应用,然后在潜在的受害

者中间开始传播。签名验证也非常缺失。毫无疑问，此类攻击需要耗费作恶者一番精力——欺骗用户下载修改后的应用。话虽如此，但该攻击悄无声息，因此汽车在用户毫无知觉的情况下就已经被盗了。

但应用程序 1 有一处做的还不错，就是利用 SSL 证书创建连接。总之，还算比较合理，因为 SSL 证书创建的连接能够预防中间人攻击。

应用程序 2

应用程序 2 提出保存用户凭证，同时推荐加密整个设备以防盗窃。听起来有道理，但我们不是去偷手机——只需“感染”即可。结论是，与之前应用程序 1 犯了同样的错误：用户名和密码以明文存储在 prefs file 中。{????????}.xml file（问号代表应用生成的随机字符）。

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <string name="USERNAME">[REDACTED]</string>
  <boolean name="VERIFIED" value="true" />
  <boolean name="CRMExportDone" value="true" />
  <string name="HOME_COUNTRY_ISO_CODE">GB</string>
  <string name="LastName">Rojers</string>
  <string name="PASSWORD">[REDACTED]</string>
  <string name="TOKENID">i6rq/YyWexko9K4j/MqDgbCW8k@EDzrqBJEHZGHU
sG+vTPWDQ4lxCDf1McDHRQMZG6CiQ/4UcUuePHdse7+RSMkKsNEgy1f8Mtkdce5s@ZD
uilSh0=</string>
  <boolean name="PrivacyWarningShown" value="true" />
  <string name="PASSWORDHASH_">WwJheOfI0r5rIrEiFQEFpSg1IUitILjYT9
<string name="MSGTOKENID">dS0cCR0e03Ee2nTmA12Ec0/ho7NjvRZij/Q8Y
<string name="FirstName">Ben</string>
  <boolean name="PASSWORD_POLICY_CONFORM" value="true" />
  <long name="LAST_SYNC" value="1467120065944" />
  <string name="USERID">W242860260</string>
  <int name="currentDashboardPage" value="1" />
</map>
```

随着我们调查的深入，发现问题的也更多。开发商甚至都不执行应用代码完整性检测，出于某些原因，他们还忘记了对用户名和密码进行模糊处理。我们甚至可以轻易的修改登录活动代码。

```
label_49:
    AppTracking.trackActionStarted("login");
    String v2 = this.userName;
    String v3 = this.password;
    Toast.makeText(this.getApplicationContext(), v2 + v3, 0).show();
    this.runInBackground(new LoginTask(((BaseActivity) this), v2, v3, this.persistentState));
}
```

因此，该应用程序保留了自身的功能。但，尝试登录之后，注册时输入的用户名和密码会立即显示在屏幕上。

应用程序 3

与应用匹配的汽车可随意提供控制组件启动引擎、打开车门。经销商安装的每一个组件都贴有贴纸标注了访问代码，一并交给了车主。这就是为什么即使知道 VIN，汽车也无法连接到其它凭证。

当然，还可能遭受到其它攻击：第一，应用程序极小，其 APK 大小只有 180KB；第二个整个应用的调试数据都记录在一个保存于 SD 卡的文件中。

```
private void b() {
    _monitor_enter(this);
    try {
        d.a("START", this.getApplicationContext());
        if (this.o.getString(this.getString(2131165201), "0").equals("1")) {
            if (this.u.getText().length() != 0 && this.v.getText().length() != 0) {
                goto label 28;
            }
        }
    }
}
```

登录活动初始记录

```
BufferedWriter v0 = null;
try {
    File v1_2 = new File(String.valueOf(Environment.getExternalStorageDirectory().getPath()) + "/marcsApp/");
    if (!v1_2.exists()) {
        v1_2.mkdir();
    }

    v2 = new Date();
    Stringbuilder v6 = new Stringbuilder(String.valueOf(Environment.getExternalStorageDirectory().getPath()) + "/marcsApp/log").append(v3.getYear() + 1900);
    if (v3.getMonth() + 1 < v0) {
        v2 = "0" + (v3.getMonth() + 1);
    }
    else {
        v2_1 = Integer.valueOf(v3.getMonth() + 1);
    }
    v6 = v6.append(v2_1);
    if (v3.getDate() < v0) {
        v2 = "0" + v3.getDate();
    }
    else {
        v2_1 = Integer.valueOf(v3.getDate());
    }
    v6 = v6.append(v2_1);

    v1_3 = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(v6.append(v2_1).append(".txt").toString(), true), "UTF-8"));
}
```

日志文件的转储位置

不幸的是，只有在应用程序中设置下列标识时才会登录：android:debuggable="true"。公开版的应用显然没有标识，但这无法阻止我们插入标识。我们只需使用 APK 工具即可实现。在运行编辑过的应用并尝试登录后，设备的 SK 卡会创建带有 TXT 后缀的 marcsApp 文件夹。在我们的案例中，账户的用户名和密码保存到了文件中。

```
0116/07/00 17:57:04 752 GMT+03:00 a5m11:12 Network kind: Wi-Fi 1 Rssi: -19 1. Main Login 4
0116/07/00 17:57:04 750 GMT+03:00 b5m11:1 Entitled: -va@pocsource LoginIncert if ind
0116/07/00 17:57:04 755 GMT+03:00 b5m11:1 Entitled: -va@pocsource &phoneId=364a21c798f14536a133d2527cd181e
0116/07/00 17:57:06 744 GMT+03:00 a5m11:12 <?xml version="1.0" encoding="utf-8"?>
LoginIncert if ind
```

当然，劝说受害者移除原始应用程序，安装带有调试标识的同一个应用程序并不容易。不过，我们可以通过引诱受害者到编辑后的应用所在的网站，然后手动安装下载重要更新缓慢进行。经验丰富的病毒作者，一般都擅长于使用社会工程，例如该应用程序。现在，为应用程序添加功能（以短信的形式向指定服务器或号码发送日志文件）轻而易举。

应用程序 4

该应用允许现有的 VIN 与任何认证绑定，但该服务一定会向汽车内置电脑发送请求。因此，偷窃制造粗劣的 VIN 对入侵汽车毫无助益。

然而，测试的应用程序对窗口覆盖毫无防御。如果由于这个原因，导致罪犯获取了系统的用户名和密码，车门很可能会被打开。

遗憾的是，该应用程序存储的系统用户名和许多其它感兴趣的数据，例如汽车的型号，VIN 和车牌都在 MyCachingStrategy.xml 文件下以明文标注。

应用程序 5

安装了应用的手机必须知道汽车内置电脑显示的 PIN 才能连接上汽车。这意味着，与应用程序 4 一样只知道 VIN 也无法打开车门，必须从汽车内部才能访问。

应用程序 6

这是俄罗斯开发商开发的一款应用，与之前用车主手机验证的观念完全不同。它对所有的车主创建一定程度的风险管理：发动攻击，只需安卓 API 函数一执行即可获取系统用户名。

应用程序 7

我们上一个审查的应用程序，必须注意到它的用户名和密码都是以明文存储在.xml 凭证文件中。


```

credentials.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <int name="pref.userstore.msgflag" value="0" />
  <int name="pref.userstore.group" value="7" />
  <string name="pref.userstore.pincod" value="703356"/>
  <string name="dcsauthkey">[REDACTED]</string>
  <string name="pref.userstore.measurement">Imperial</string>
  <string name="pwd">[REDACTED]</string>
  <int name="pref.userstore.usercode" value="845866" />
  <string name="user_first_name">Ben</string>
  <string name="dcssessid">2[REDACTED]</string>
  <string name="pref.userstore.language">English</string>
  <string name="pref.userstore.lastname">[REDACTED]</string>
  <string name="sessionid">[REDACTED]</string>
  <string name="pref.userstore.accountid">315324</string>
  <string name="username">[REDACTED].ru</string>
  <string name="pref.userstore.accountname">[REDACTED]@mail.ru</string>
  <string name="pref.userstore.email">[REDACTED]@mail.ru</string>
  <string name="user_last_name">[REDACTED]</string>
  <long name="pref.userstore.id" value="579509" />
  <string name="pref.userstore.username">[REDACTED].ru</string>
  <string name="pref.userstore.firstname">[REDACTED]</string>
  <string name="pref.userstore.timezone"></string>
  <string name="pref.userstore.phone"></string>
</map>

```

如果拥有超级用户权限的木马成功感染手机，窃取文件将不费吹灰之力。

汽车失窃的几率

理论上说，窃取凭证后，罪犯可以控制汽车，但并不是说罪犯能直接将汽车开走。实际上，还是需要钥匙启动汽车。因此，访问汽车内部系统后，偷车贼利用程序设备往汽车内部系统写入新的钥匙控制程序。现在，回忆一下，几乎我们讨论的所有应用程序都允许打开车门，那就是说，它能解除汽车警报系统。因此，罪犯可以在不毁坏任何东西的情况下，迅速地，偷偷地实施所有的行动盗窃汽车。

另外，汽车失窃的风险远不止于此。访问汽车，故意篡改可能导致道路交通事故，造成伤亡。

虽然所有审查的应用程序都没有防御机制。但我们仍应赞扬应用开发商：我们应庆幸上述的应用程序没有一个是用语音或短信控制汽车的。但是，有些售后市场警报系统制造商使用了这些方法，包括俄罗斯制造商。一方面，我们对此并不惊奇，因为移动网络通讯质量无法随时保障汽车连接到网络，但语音电话和短信却能获取这一基本功能。否则的话，我们审查的应用会产生大量的安全威胁。

语音控制是由名为 DTMF 的指令控制的。表面上，车主必须呼叫汽车，然后警报系统响应以女声报告车况，然后系统转待机模式等待车主发送指令。之后，手机拨打预设号码命令汽车打开车门、发动引擎。然后，警报系统识别代码，执行恰当的指令。

系统开发商已考虑到了通过提供手机白名单，来保障汽车控制权的安全性。但人们没想到车主手机可能会被入侵这种情况。这意味着，罪犯足以感染受害人手机中存在漏洞的应用，以受害人的身份控制警报系统。如果系统喇叭和屏幕同时关闭，罪犯很有可能瞒住受害人完全控制汽车。

当然，不是所有的事情都像它看起来的那么简单。例如，很多汽车爱好者在警报系统中用假名，那就是说罪犯要成功发动攻击，就必须通过电话频繁与受害者联系。只有这样，罪犯才能盗取呼叫记录，在受害人联系人中找到汽车号码。

其它汽车警报系统开发商当然没有读过我们关于安卓设备安全文章，因为汽车是通过 SMS 指令操作的。实际情况是，卡巴斯基实验室遇到的第一个最常用的手机木马是 SMS 木马，或者内含代码秘密发送 SMS 的恶意软件，它们既可通过普通的木马操作，又可以通过罪犯发送远程指令完成。结果是，一旦恶意软件开发商执行以下三步，受害人的车门就会被打开。

1. 浏览手机上所有的短信，找出汽车指令
2. 一旦需要的短信被锁定，然后就可从中提取手机号码和密码获取访问权
3. 向发现的号码发送短信，打开车门以上三步在受害人察觉之前就可完成。罪犯需要做的也能办到的唯一一件事就是感染手机。

结论

确保汽车的安全并不比保护银行账户轻松，而且花费极高。汽车制造商和开发商的态度很明确：利用应用的新功能提升车主的生活质量借以迅速抢占市场。但，考虑联网汽车安全性时，我们不能只考虑基础设施（控制服务器）、交互和基础设施渠道的安全性。我们还应把注意力放到客户身上，尤其是用户设备安装的应用。现今，利用应用针对车主实在是太容易了，因为他们是整个环节最薄弱的点，极可能被罪犯攻击。

迄今为止,我们还未发现一起应用控制汽车的攻击,在数千个检测的恶意软件中也不包含下载此类应用的配置文件代码。但是,当今木马非常的狡猾:如果木马持续显示广告(用户自己无法关闭),在罪犯请求时,它会将汽车应用程序的配置文件上传到 C2 服务器。木马还可以删除配置文件并用修改之后的应用控制汽车。一旦罪犯能够承担行动的所有费用,我们可以预见即使是最普通的手机木马也会出现新的功能。