GodFather Malware Targets 500 Banking & Crypto Apps Worldwide

Published: 2024-11-06 · Archived: 2026-04-05 16:09:29 UTC

GodFather Malware Expands Its Reach, Targeting 500 Banking And Crypto Applications Worldwide

GodFather Malware Expands Its Reach, Targeting 500 Banking And Crypto Applications Worldwide

Cyble analyzes the latest iteration of the GodFather Android banking trojan, which targets over 500 cryptocurrency and banking applications and has expanded its reach to Japan, Greece, Singapore, and Azerbaijan.

Key Takeaways

- Cyble Research and Intelligence Labs (CRIL) has identified a new variant of the GodFather malware, now targeting 500 banking and cryptocurrency apps.
- Initially focused on regions like the UK, US, Turkey, Spain, and Italy, GodFather has expanded its reach to include Japan, Singapore, Greece, and Azerbaijan.
- The GodFather malware has transitioned the Java code implementation to the Native code for its malicious activities.
- In its latest version, the GodFather malware uses limited permissions, relying heavily on Accessibility services to capture credentials from targeted applications.
- This updated variant also includes new commands that enable the malware to automate gestures on infected devices, mimicking user actions.
- The Threat Actor(TA) behind GodFather malware uses a phishing site to deliver the suspicious app and tracks visitor counts to plan further activity.

Overview

Cyble Research and Intelligence Labs (CRIL) recently identified a phishing site, “mygov-au[.]app,” masquerading as the official MyGov website of the Australian Government. Upon further analysis, this site was found to be distributing a suspicious APK file linked to the GodFather Malware, known for its ability to steal banking application credentials.

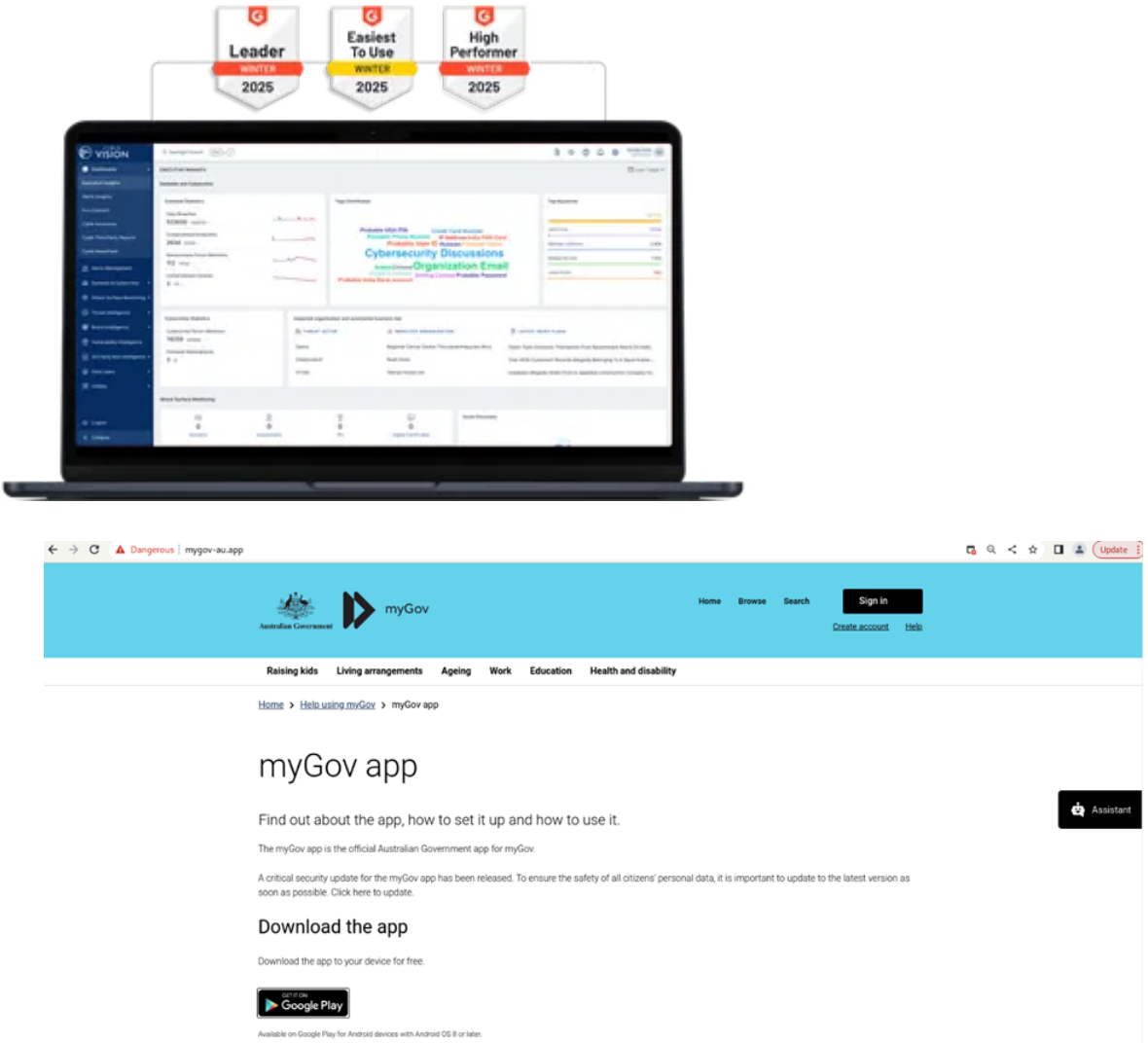

Figure 1 – Phishing site impersonating myGov website distributing APK file

The downloaded application, “MyGov.apk”, communicates with the URL “hxxps://az-inatv[.]com/.” This app is programmed to track the number of devices it is installed on, retrieve the device’s IP address, and store this information on the server in a text file. Figures 3 and 4 show the code of index.php and count.php responsible for getting the count and IP address.

```
    public static void error(Context context) {
        try {
            WebSettings webSettings = webView.getSettings();
            webSettings.setJavaScriptEnabled(true);
            WebViewClientImpl webViewClient = new WebViewClientImpl(context);
            webView.setWebViewClient(webViewClient);
            webView.loadUrl("https://az-inatv.com/Count1/index.php");
        } catch (Exception e) {
        }
    }

    public static void info(Context context) {
        try {
            WebSettings webSettings = webView.getSettings();
            webSettings.setJavaScriptEnabled(true);
            WebViewClientImpl webViewClient = new WebViewClientImpl(context);
            webView.setWebViewClient(webViewClient);
            webView.loadUrl("https://az-inatv.com/Count2/index.php");
        } catch (Exception e) {
        }
    }
}
```

Figure 2 – Malware loading URL, which maintains the counter

1 index.php

```php
<?php
require __DIR__. '/counter/counter.php';

// make one hit
makeHit();

// printing total hits
echo "Total Hit: " . getHit();

echo "<br>";

// add IP address if it doesn't exist in list
addUniqueIP();

// print unique visitors
echo "Unique Visitors: " .getUniqueVisitor();
```

Figure 3 – Getting counts and IP addresses

1 counter.php

```php
<?php

function makeHit(){
        $hit = file_get_contents(__DIR__ . "/hits.txt");
        file_put_contents(__DIR__ . "/hits.txt", $hit + 1);
}


function getHit(){
        return file_get_contents(__DIR__ . "/hits.txt");
}

function addUniqueIP($ip=NULL){
        $ip = ($ip!=NULL) ? $ip : getIP();
        $iplist = file_get_contents(__DIR__ . '/iplist.txt');
        $iplist = explode(",", $iplist);
        if(!in_array(trim($ip), $iplist)){
                $file = fopen(__DIR__ . "/iplist.txt", 'a+');
                fwrite($file, "," . trim($ip));
                fclose($file);
        }
}

function getUniqueVisitor(){
        $file = file_get_contents(__DIR__ . "/iplist.txt");
        $file = explode(",", $file);
        return count($file);
}


// other

function getIP() {
    $ipaddress = NULL;
    if (isset($_SERVER['HTTP_CLIENT_IP']))
      { $ipaddress = $_SERVER['HTTP_CLIENT_IP'];}
        else if(isset($_SERVER['HTTP_X_FORWARDED_FOR']))
          { $ipaddress = $_SERVER['HTTP_X_FORWARDED_FOR'];}
        else if(isset($_SERVER['HTTP_X_FORWARDED']))
          { $ipaddress = $_SERVER['HTTP_X_FORWARDED'];}
        else if(isset($_SERVER['HTTP_FORWARDED_FOR']))
          { $ipaddress = $_SERVER['HTTP_FORWARDED_FOR'];}
        else if(isset($_SERVER['HTTP_FORWARDED']))
          {$ipaddress = $_SERVER['HTTP_FORWARDED'];}
        else if(isset($_SERVER['REMOTE_ADDR']))
          {$ipaddress = $_SERVER['REMOTE_ADDR'];}
        else
          {$ipaddress = NULL;}
```

Figure 4 – Getting the IP address of an infected device

The URL “hxxps://az-inatv[.]com/” hosted an open directory containing a file named **counters.zip**, which included the total count of infected devices and a list of IP addresses. Additionally, the directory featured a page labeled “down” that hosted another APK file called “lnat Tv Pro 2024.apk.” Upon analyzing this APK, it was identified as the GodFather Malware.

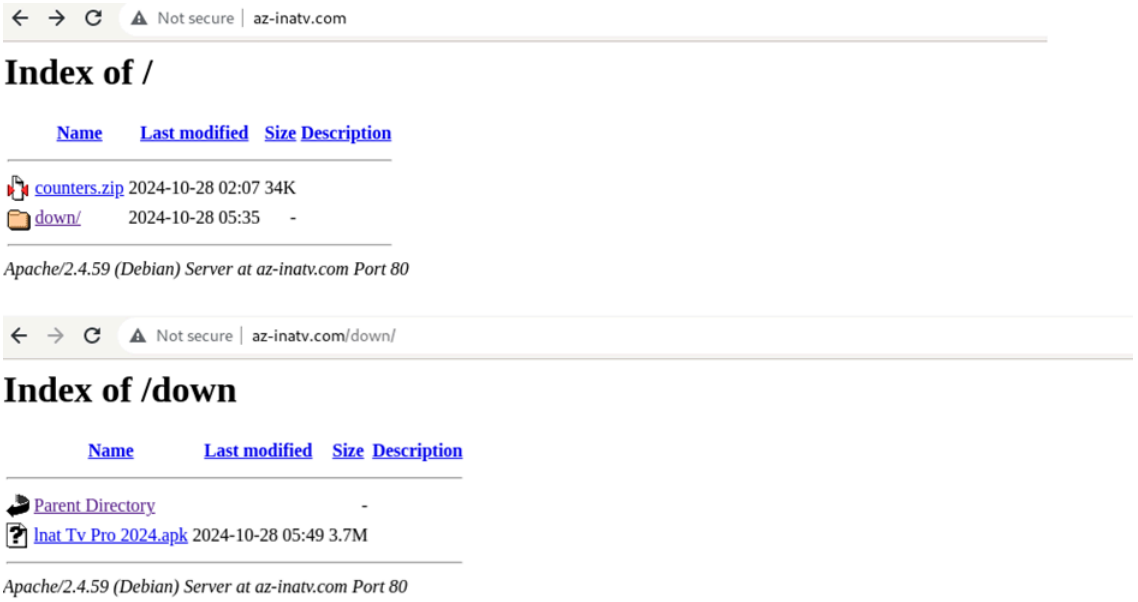

Figure 5 – Open directory hosting counters.zip and GodFather malware

Upon examining the **counters.zip** file, we found 151 counts in **hit.txt** and 59 unique IP addresses, reflecting the targeted device count. While the MyGov application collected this data, we suspect the TA may leverage this visitor information to identify potential victim counts and later use the same website to distribute the GodFather malware.

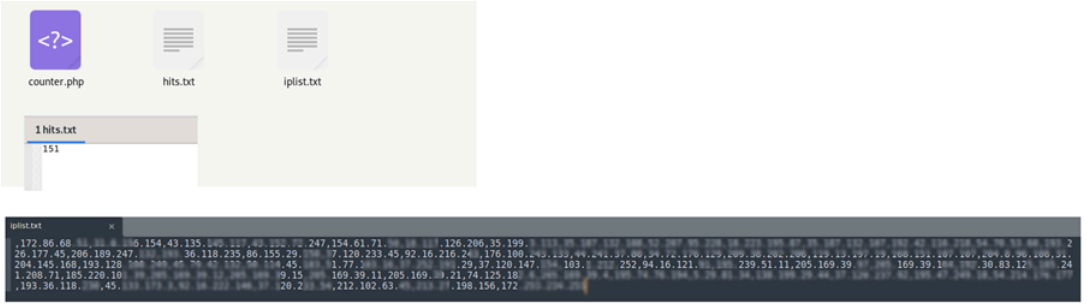

Figure 6 – Counters.zip content

Notably, we observed that the latest variant of the GodFather malware has moved from Java code to native code implementation. It is now targeting 500 banking and cryptocurrency applications and expanding its reach to Japan, Singapore, Azerbaijan, and Greece. Further details on this new variant of GodFather are provided in the following section.

Technical Details

In the latest version, the GodFather malware operates with minimal permissions, relying heavily on the Accessibility service to carry out its malicious activities.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1" android:compileSdkVersion="23" an
    <uses-sdk android:minSdkVersion="26" android:targetSdkVersion="33"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.QUERY_ALL_PACKAGES"/>
    <application android:theme="@style/Theme.AppCompat.NoActionBar" android:label="@string/app_names" android:icon="@mipmap/ic_launcher" android:name=
        <activity android:name="com.manubria.brahmanize.Xenophoridae" android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.INFO"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.VIEW"/>
                <category android:name="android.intent.category.BROWSABLE"/>
                <data android:scheme="https" android:host="google.com"/>
            </intent-filter>
        </activity>
        <activity android:name="com.manubria.brahmanize.versiloquy" android:exported="true"/>
        <activity android:name="com.manubria.brahmanize.misinferred" android:exported="true"/>
        <service android:label="@string/app_name" android:name="com.manubria.brahmanize.lowboiling" android:permission="android.permission.BIND_ACCESS
            <intent-filter>
                <action android:name="android.accessibilityservice.AccessibilityService"/>
            </intent-filter>
            <meta-data android:name="android.accessibilityservice" android:resource="@xml/preferences"/>
        </service>
        <activity android:name="com.manubria.brahmanize.sphaeropsidaceous" android:exported="true"/>
        <provider android:name="androidx.startup.InitializationProvider" android:exported="false" android:authorities="com.manubria.brahmanize.android
            <meta-data android:name="androidx.emoji2.text.EmojiCompatInitializer" android:value="androidx.startup"/>
            <meta-data android:name="androidx.lifecycle.ProcessLifecycleInitializer" android:value="androidx.startup"/>
        </provider>
    </application>
</manifest>
```

Figure 7 – Manifest with limited permissions

Native Code Implementation

Starting our analysis with the classes specified in the manifest file, we observed that the malware calls numerous native methods, which were previously implemented in Java code.

```
public static native void Abourezk(Context context);
public static native /* synthetic */ PowerManager.WakeLock Aredale();
public static native void Denyse(Context context);
public static native /* synthetic */ GestureDescription.Builder JJ(GestureDescription.Builder builder);
public static native lowboiling Malacostraca();
public static native /* synthetic */ GestureDescription.StrokeDescription Manorville();
public static native /* synthetic */ Path Okolona();
public static native void Osnabruck(int i, int i2, AccessibilityNodeInfo accessibilityNodeInfo);
public static native void Periphas(Context context);
public static native /* synthetic */ GestureDescription.StrokeDescription Psedera(GestureDescription.StrokeDescription strokeDescription);
public static native String Samarkand(AccessibilityEvent accessibilityEvent);
public static native void Saundra(AccessibilityNodeInfo accessibilityNodeInfo);
public static native AccessibilityNodeInfoCompat Schopenhauerism(AccessibilityNodeInfoCompat accessibilityNodeInfoCompat);
public static native void Silda(Context context, String str);
public static native String Uriisa(AccessibilityEvent accessibilityEvent);
public static native String bedamned(Context context, AccessibilityNodeInfoCompat accessibilityNodeInfoCompat);
public static native /* synthetic */ GestureDescription.Builder carrageenan();
public static native void champaka(Context context);
public static native void cheerleader(Context context);
```

Figure 8 – Calls to native methods

These native functions implement various malicious capabilities, including loading an injection URL into the WebView, executing automated gestures, establishing connections with the Command and Control (C&C) server, and keylogging.

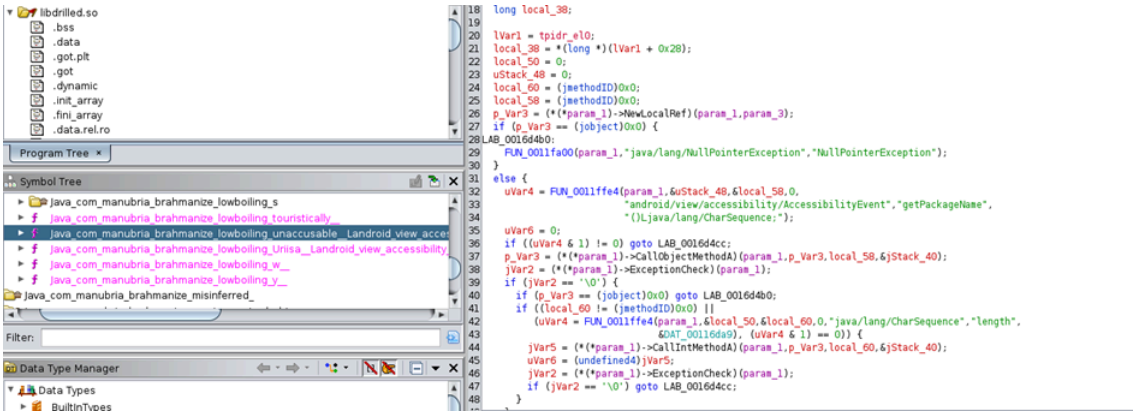

Figure 9 – Native code implementation

C&C Server

Similar to the previous variant, the latest samples also connect to the Telegram URL “hxxps://t.me/gafaramotamer,” where the TA has embedded a Base64-encoded C&C URL. The malware retrieves and decodes this URL to “hxxps://akozamora[.]top/z.php.”

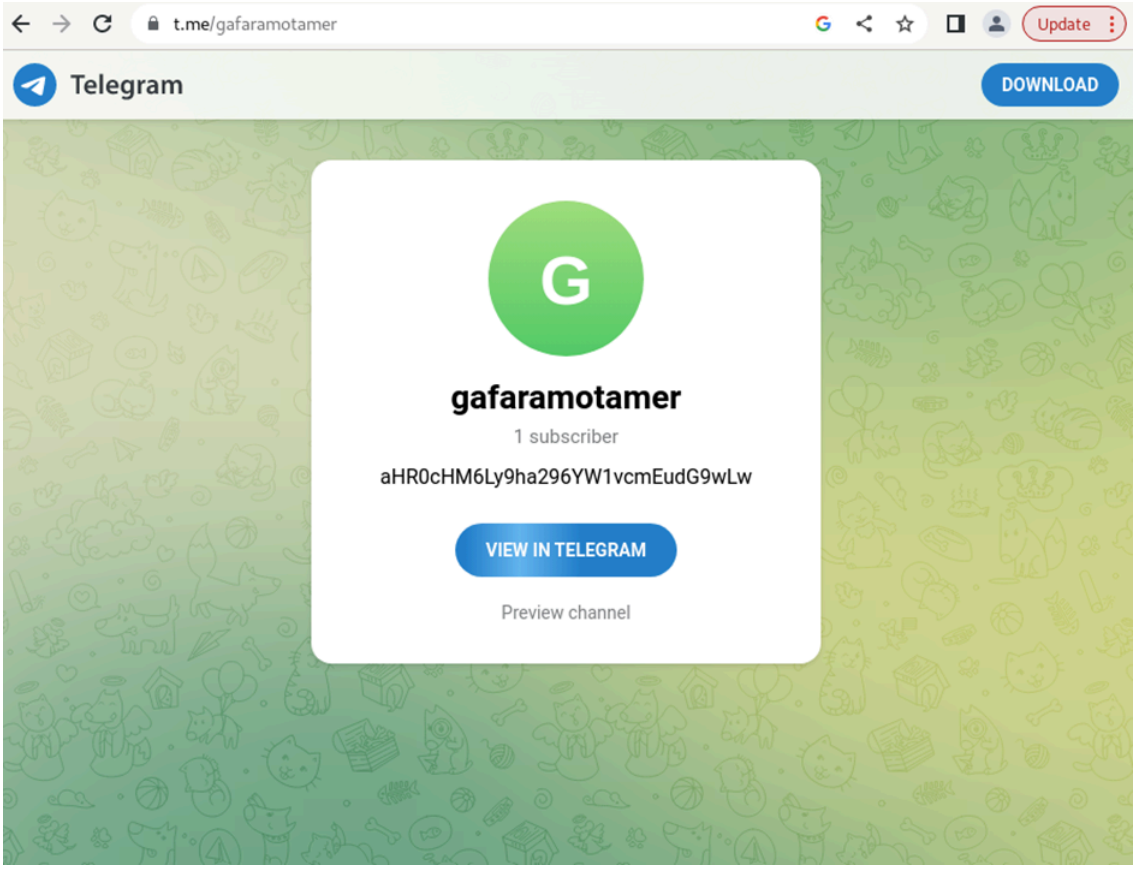

Figure 10 – Malware fetches C&C server URL from Telegram Profile

Targeting 500 Crypto and Banking Applications

After decoding the URL, the malware begins communication by sending data such as the list of installed application package names, the device's default language, model name, and SIM name. In return, it receives a list of 500 targeted application package names associated with banking and cryptocurrency apps. In addition to previous targets in the UK, US, Turkey, Spain, and Italy, GodFather has expanded its reach, now including Japan, Singapore, Greece, and Azerbaijan.

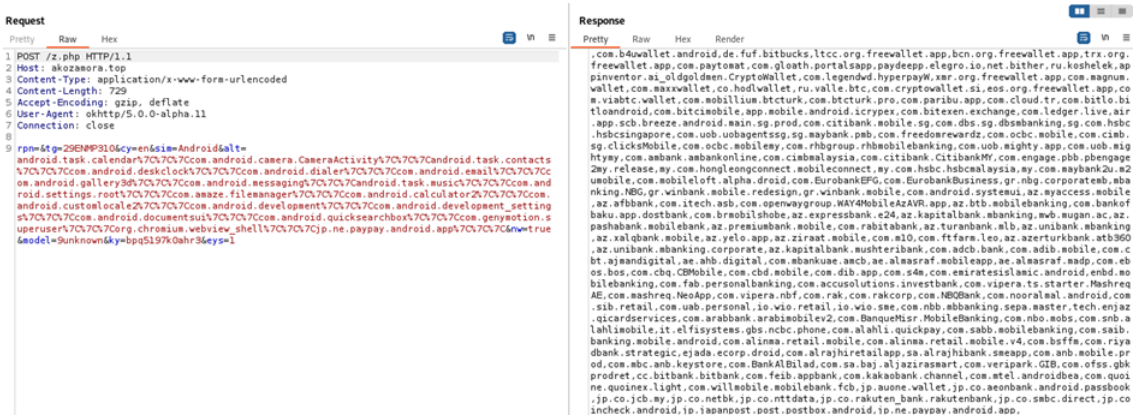

Figure 11 – Receives the list of target application package names

When the user tries to interact with the target application, the malware closes the genuine application. Instead, it loads a fake banking or crypto login URL into the WebView or displays a blank screen. It constructs the injection URL using the C&C server “hxxps://akozamora[.]top/” and appends the endpoint “rx/f.php?f=” along with the device name, package name, and default language, then loads the assembled URL in the WebView.

Figure 12 – Loading fake login pages

The GodFather malware has successfully replaced the traditional overlay attack with this technique. Rather than launching the legitimate application, the malware activates itself and loads a phishing page to steal banking credentials.

Commands Added In New Version

The previous version included commands for USSD and SMS operations, which have been removed in the latest version. Additionally, this malware version lacks permission to collect or send SMS messages from the infected device. Instead, the newly added commands focus primarily on automating actions on the infected device. Below is a list of commands observed in the latest version of the GodFather malware.

Command	Description
clickposition	Malware clicks on the position X and Y received from the server
backed	Take the user to the previous screen
home	Take the user to the home screen
recents	Take the user to the recent screen
scrollforward	Malware scrolls the page forward using the given parameter
scrollback	It scrolls the page backward till using the provided parameter
opencontrol	Perform gestures on the target app

setpattern	Receives some value from the server and saves it to a shared preference variable “pc”
screenlight	Manages the brightness of the screen
sl2	Setting up a wake lock to keep the device awake
sl3	Similar to sl2
autopattern	The value received using “setpattern” command is used to insert on the device screen using the accessibility service.
csn	Set the timer to initiate the WebSocket connection
swpfull	Perform swipe operation
upswp	Perform swipe up
downswp	Perform swipe down
leftswp	Perform left swipe
rightswp	Perform right swipe
vncreset	Not Implemented
opnap	Open the application whose package name is received from the server
gif	Loads Gif from link “hxxps://s6.gifyu.com/images/S8uz3.gif”
opnsttings	Opens setting app
opnsound	Opens sound setting
opnmsc	Opens notification setting
opnpckg	Not Implemented
notifyopen	Opens notification using Accessibility service

Conclusion

The latest version of the GodFather malware shows how dangerous and adaptable mobile threats have become. By moving to native code and using fewer permissions, the attackers have made GodFather harder to analyze and better at stealing sensitive information from banking and cryptocurrency apps. With its new automated actions and broader targeting of apps in more countries, this malware poses a growing risk to users worldwide. Staying alert and using strong security practices on mobile devices is essential to avoid falling victim to threats like GodFather.

Our Recommendations

We have listed some essential cybersecurity best practices that create the first line of control against attackers. We recommend that our readers follow the best practices given below:

- Download and install software only from official app stores like Google Play Store or the iOS App Store.
- Use a reputed anti-virus and internet security software package on your connected devices, such as PCs, laptops, and mobile devices.
- Use strong passwords and enforce multi-factor authentication wherever possible.
- Enable biometric security features such as fingerprint or facial recognition for unlocking the mobile device where possible.
- Be wary of opening any links received via SMS or emails delivered to your phone.
- Ensure that Google Play Protect is enabled on Android devices.
- Be careful while enabling any permissions.
- Keep your devices, operating systems, and applications updated.

MITRE ATT&CK® Techniques

Tactic	Technique ID	Procedure
Initial Access (TA0027)	Phishing (T1660)	Malware distributing via phishing site
Execution (TA0041)	Native API (T1575)	Malware using native code to drop final payload
Persistence (TA0028)	Scheduled Task/Job (T1603)	Uses timer to initiate WebSocket connection
Defense Evasion (TA0030)	Masquerading: Match Legitimate Name or Location (T1655.001)	Malware pretending to be a genuine Music application
Defense Evasion (TA0030)	Application Discovery (T1418)	Collects installed application package name list to identify target
Defense Evasion (TA0030)	Input Injection (T1516)	Malware can mimic user interaction, perform clicks and various gestures, and input data
Collection (TA0035)	Input Capture: Keylogging (T1417.001)	Malware can capture keystrokes
Discovery (TA0032)	System Information Discovery (T1426)	The malware collects basic device information.
Command and Control (TA0037)	Web Service: Dead Drop Resolver (T1481.001)	Malware communicates with Telegram to fetch C&C server

Exfiltration (TA0036)	Exfiltration Over C2 Channel (T1646)	Sending exfiltrated data over C&C server

Indicators of Compromise (IOCs)

Indicators	Indicator Type	Description
d8165712329fa120b5cc696514b5dd0d7043fbf7d6b6ef5f767348e0ba31aa6e e789b03b60ad99727ea65b52ce931482fb70814e 87ccf62e07cf69c25a204bffdbc89630	SHA256 SHA1 MD5	Analyzed GodFather malware
hxxps://akozamora[.]top/	URL	C&C server
hxxps://t.me/gafaramotamer	URL	Malware fetching C&C from Telegram URL
hxxps://az-inatv[.]com	URL	URL hosting new GodFather variant
mygov-au[.]app	Domain	Phishing domain distributing counter app
8ae2fcc8bef4d9a0ae3d1ac5356dbd85a4f332ad497375cd217bd1e945e64692 d57ef894b53f804c97d40c3e365faf729ce2ea7386b280f9909ebc8432008eee d508078368d8775fcfff5a7886392da57fcf757c89687f22c0504c3df9075b00 b3d3019ed0a4602fb7e502e54ac12a59da1a0ed7b6736feb98ce7c417091b2e6 3aa7e2353c2de16734f612eba7b43a2538d96f73702a6c25283d6ef0c9300a4c 1ce2a392dd2c1df22dfeb080c7ad290d63e3afe983729927b2f15c6705861070 d8165712329fa120b5cc696514b5dd0d7043fbf7d6b6ef5f767348e0ba31aa6e d8165712329fa120b5cc696514b5dd0d7043fbf7d6b6ef5f767348e0ba31aa6e 0c9e2ae9c699374f06a6d38cf2ea41232fc8a712e110be8069b08659fdf50514 19ed4f67710d455da42017de28688f5e55ed36809cc70252d825ac81713e95d1 7b4543cc4df1fc57af2cd9a892b2fab3647bdceb027d576217724a8c012a2065 2b1b527b87929a13f0c33391c641b3013da099fd7de10695d762da097bc13ffc 2b1b527b87929a13f0c33391c641b3013da099fd7de10695d762da097bc13ffc 72d40ff8ad114724b8d4e0350f81f797866c0f271844aeddc3b92f33faa6fbc0	SHA256	New GodFather variant hashes

Source: https://cyble.com/blog/godfather-malware-targets-500-banking-and-crypto-apps-worldwide/