Ginp - A malware patchwork borrowing from Anubis

threatfabric.com/blogs/ginp_a_malware_patchwork_borrowing_from_anubis.html

November 2019

Intro

ThreatFabric analysts have recently investigated an interesting new strain of banking malware. The malware was first spotted by Tatyana Shishkova from Kaspersky by end October 2019, but actually dates back to June 2019. It is still under active development, with at least 5 different versions of the Trojan released within the last 5 months (June - November 2019).

What makes Ginp stand out is that it was built from scratch being expanded through regular updates, the last of which including code copied from the infamous Anubis banking Trojan, indicating that its author is cherry-picking the most relevant functionality for its malware. In addition, its original target list is extremely narrow and seems to be focused on Spanish banks. Last but not least, all the overlay screens (injects) for the banks include two steps; first stealing the victim's login credentials, then their credit card details. Although multi-step overlays are not something new, their usage is generally limited to avoid raising suspicion.

Evolution

The initial version of the malware dates back to early June 2019, masquerading as a "Google Play Verificator" app. At that time, Ginp was a simple SMS stealer whose purpose was only to send a copy of incoming and outgoing SMS messages to the C2 server.

A couple of months later, in August 2019, a new version was released with additional banking-specific features. This and following versions were masquerading as fake "Adobe Flash Player" apps. The malware was able to perform overlay attacks and become the default SMS app through the abuse of the Accessibility Service. The overlay consisted of a generic credit card grabber targeting social and utility apps, such as Google Play, Facebook, WhatsApp, Chrome, Skype, Instagram and Twitter.

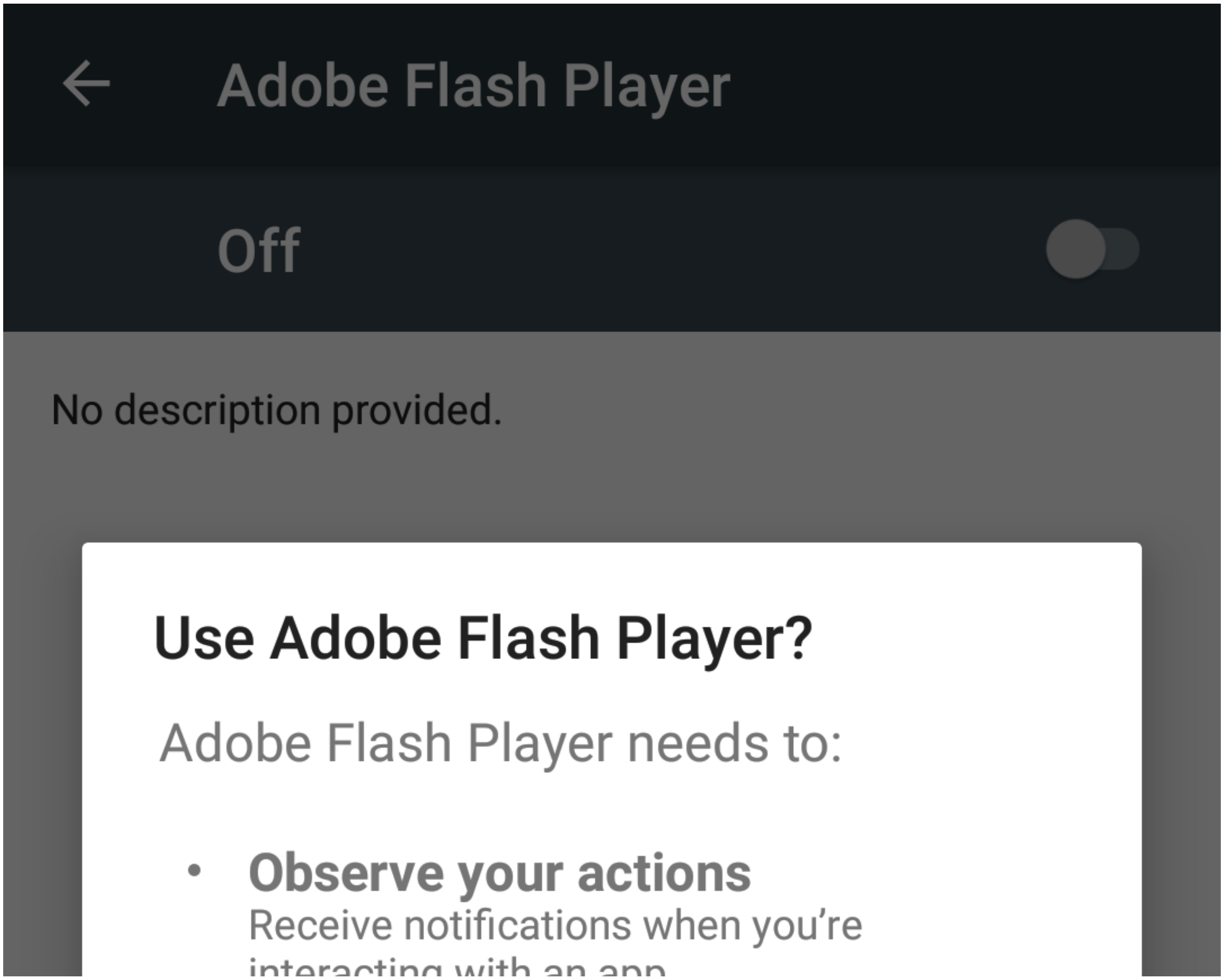
Although early versions had some basic code and string obfuscation, protection of the third version of the malware was enhanced with the use of payload obfuscation. The capabilities remained unchanged, but a new endpoint was added to the Trojan C2 allowing it to handle the generic card grabber overlay and specific target overlays (banking apps) separately. In addition, the credit card grabber target list was expanded with Snapchat and Viber.

In the third version spotted in the wild, the author introduced parts of the source code of the infamous Anubis Trojan (which was leaked earlier in 2019). This change came hand in hand with a new overlay target list, no longer targeting social apps, but focusing on banking instead. A remarkable fact is that all the targeted apps relate to Spanish banks, including targets never seen before in any other Android banking Trojan. The 24 target apps belong to 7 different Spanish banks: Caixa bank, Bankinter, Bankia, BBVA, EVO Banco, Kutxabank and Santander. The specific apps can be found in the target list in the appendix.

The most recent version of Ginp (at the time of writing) was detected at the end of November 2019. This version has some small modifications which seems to be unused, as the malware behaviour is the same as the previous version. The author has introduced the capability to grant the app the device admin permission. Additionally new endpoint was added that seems related to downloading a module for the malware, probably with new features or configuration.

How it works

When the malware is first started on the device it will begin by removing its icon from the app drawer, hiding from the end user. In the second step it asks the victim for the Accessibility Service privilege as visible in following screenshot:



Adobe Flash Player

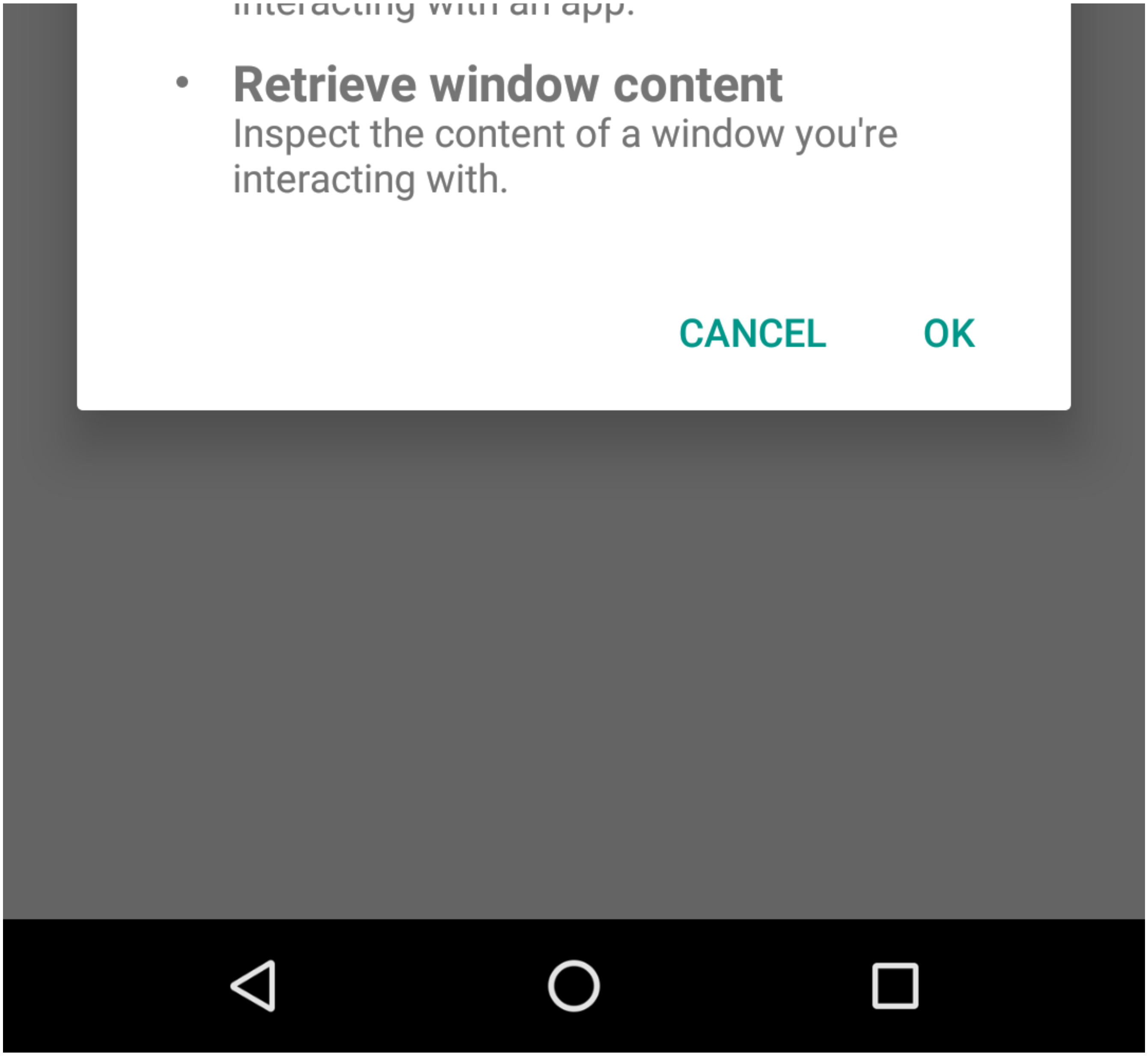
Off

No description provided.

Use Adobe Flash Player?

Adobe Flash Player needs to:

- Observe your actions
  Receive notifications when you're interacting with an app



interacting with an app.

- **Retrieve window content**
  Inspect the content of a window you're interacting with.

Once the user grants the requested Accessibility Service privilege, Ginp starts by granting itself additional permissions, such as (dynamic) permissions required in order to be able to send messages and make calls, without requiring any further action from the victim. When done, the bot is functional and ready to receive commands and perform overlay attacks.

The commands supported by the most recent version of the bot are listed below. As can be observed, the possibilities offered by the bot are pretty common.

Command	Description
SEND_SMS	Send an SMS from the bot to a specific number
NEW_URL	Update the C2 URL
KILL	Disable the bot
PING_DELAY	Update interval between each ping request
CLEAN_IGNORE_PKG	Empty list of overlayed apps

Command	Description
WRITE_INJECTS	Update target list
READ_INJECTS	Get current target list
START_ADMIN	Request Device Admin privileges
ALL_SMS	Get all SMS messages
DISABLE_ACCESSIBILITY	Stop preventing user from disabling the accessibility service
ENABLE_ACCESSIBILITY	Prevent user from disabling the accessibility service
ENABLE_HIDDEN_SMS	Set malware as default SMS app
DISABLE_HIDDEN_SMS	Remove malware as default SMS app
ENABLE_EXTENDED_INJECT	Enable overlay attacks
DISABLE_EXTENDED_INJECT	Disable overlay attacks
ENABLE_CC_GRABBER	Enable the Google Play overlay
DISABLE_CC_GRABBER	Disable the Google Play overlay
START_DEBUG	Enable debugging
GET_LOGCAT	Get logs from the device
STOP_DEBUG	Disable debugging
GET_APPS	Get installed applications
GET_CONTACTS	Get contacts
SEND_BULK_SMS	Send SMS to multiple numbers
UPDATE_APK	*Not implemented*
INJECT_PACKAGE	Add new overlay target
CALL_FORWARD	Enable/disable call forwarding
START_PERMISSIONS	Starts request for additional permissions(Accessibility privileges, battery optimizations bypass, dynamic permissions)

Features

The most recent version of Ginp has the same capabilities as most other Android banking Trojans, such as the use of overlay attacks, SMS control and contact list harvesting. Overall, it has a fairly common feature list, but it is expected to expand in future updates. Since Ginp is already using some code from the Anubis Trojan, it is quite likely that other, more advanced features from Anubis or other malware, such as a back-connect proxy, screen-streaming and RAT will also be added in the future.

Ginp embeds the following set of features, allowing it to remain under the radar and successfully perform attacks:

- Overlaying: Dynamic (local overlays obtained from the C2)
- SMS harvesting: SMS listing
- SMS harvesting: SMS forwarding
- Contact list collection
- Application listing

- Overlaying: Targets list update
- SMS: Sending
- Calls: Call forwarding
- C2 Resilience: Auxiliary C2 list
- Self-protection: Hiding the App icon
- Self-protection: Preventing removal
- Self-protection: Emulation-detection

Update 10/03/2020

At the end of February the actors behind Ginp added screen capture capabilities to their Trojan. Like previously added functionality, the code is borrowed from the leaked Anubis Trojan source code. It enables the bot to stream screenshots and send them to the C2 so that actors can see what is happening on the screen of the infected device.

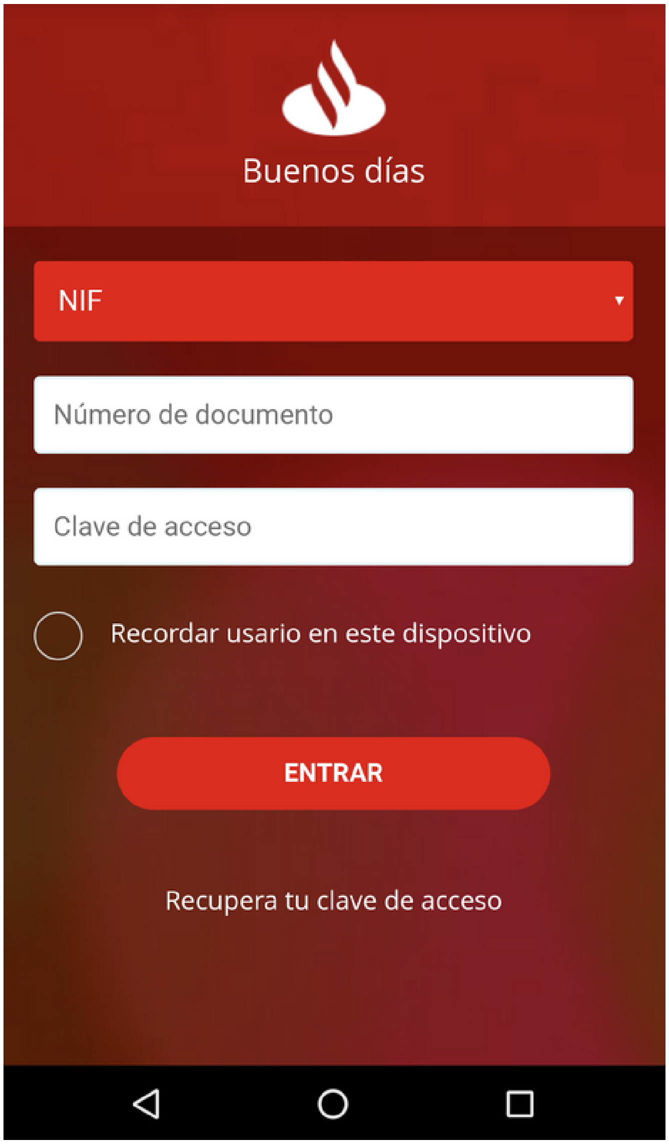
Overlay attack

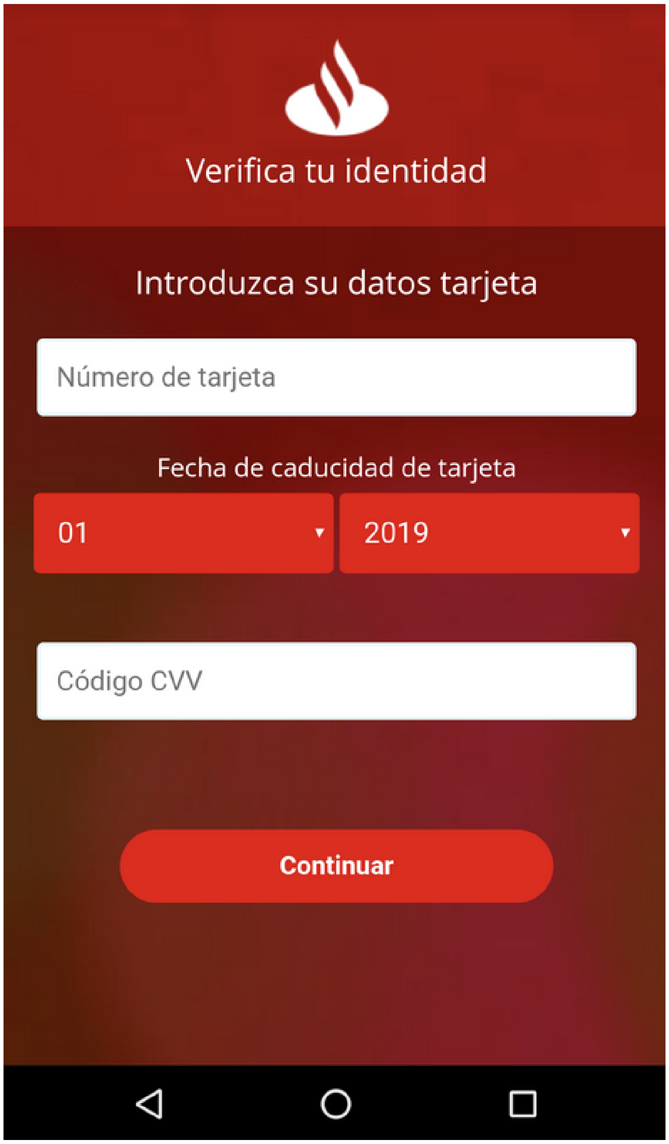
Ginp uses the Accessibility Service to check which application runs is the foreground. If the package name of the foreground app is included in the target list, an overlay is shown. The WebView-based overlay is loading an HTML page provided by the C2 in response to the package name provided by the bot.

The following code snippet shows how the WebView is created.

```
protected void onCreate(Bundle arg6) {
    super.onCreate(arg6);
    this.packageName = this.getIntent().getStringExtra("packageName");
    this.mContext = this;
    String v6 = this.store.get(this, "INJ_URL");
    Utils v0 = this.Tools;
    v0.Log(this, "\[Injector\] Starting injection on " + this.packageName, Boolean.valueOf(true));
    WebView v0_1 = new WebView(this);
    v0_1.getSettings().setJavaScriptEnabled(true);
    v0_1.setScrollBarStyle(0);
    v0_1.setWebViewClient(new MyWebViewClient(this, null));
    v0_1.setWebChromeClient(new MyWebChromeClient(this, null));
    String v1_1 = Resources.getSystem().getConfiguration().locale.getCountry();
    String v2 = Resources.getSystem().getConfiguration().locale.getLanguage();
    v0\_1.loadUrl(v6 + "?package=" + this.packageName + "&device\_id=" + this.store.get(this,
"ANDROID\_ID") + "&country=" + v1\_1.toLowerCase() + "&lang=" + v2.toLowerCase());
    this.setContentView(v0_1);
}
```

Something that makes Ginp special is that all of its overlay screens for banking apps are consist of multiple steps, first stealing the victim’s login credentials, then stealing the credit card details (to “validate” the user identity), as shown in the screenshots hereafter:


Buenos días
NIF
Número de documento
Clave de acceso
○ Recordar usario en este dispositivo
ENTRAR
Recupera tu clave de acceso



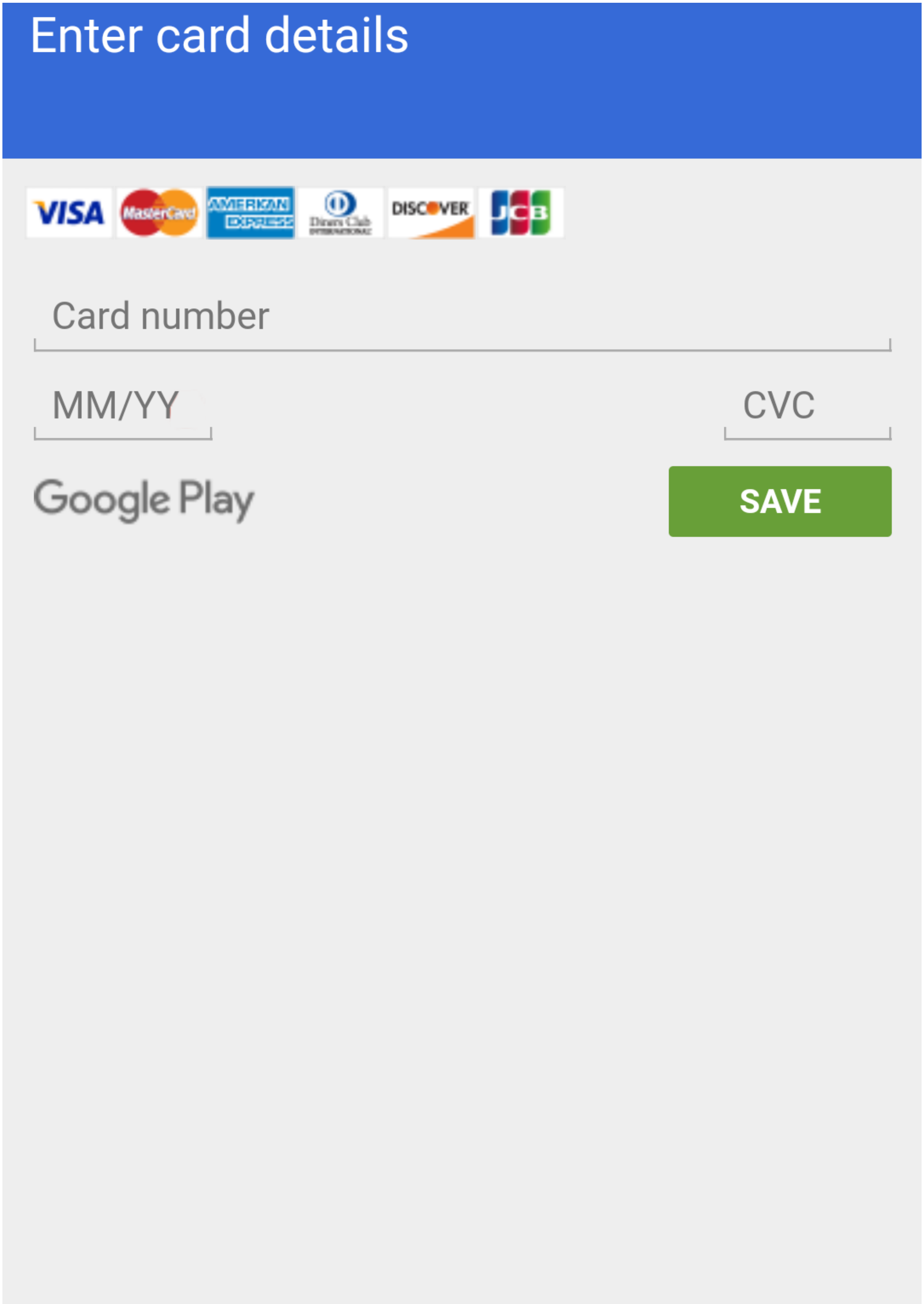
The following code snippet shows that after the second overlay is filled-in and validated, it disappears and the targeted application is added to the list of packages names to be ignored for future overlays attacks.

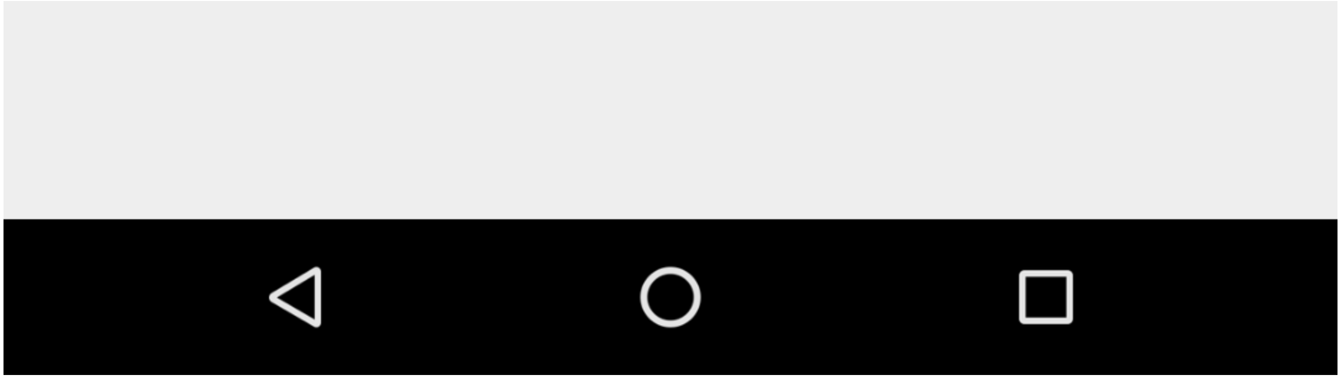
```
public void onPageFinished(WebView arg3, String arg4) {
    if(arg4.contains("|DONE|")) {
        Utils v3 = ActivityInjection.this.Tools;
        Context v4 = ActivityInjection.this.mContext;
        v3.Log(v4, "\[Injector\] Grabbing on " + ActivityInjection.this.packageName + " completed.",
Boolean.valueOf(true));
        ActivityInjection.this.Tools.addPackageToIgnore(ActivityInjection.this.mContext,
ActivityInjection.this.packageName);
        ActivityInjection.this.finish();
    }
}
```

Targets

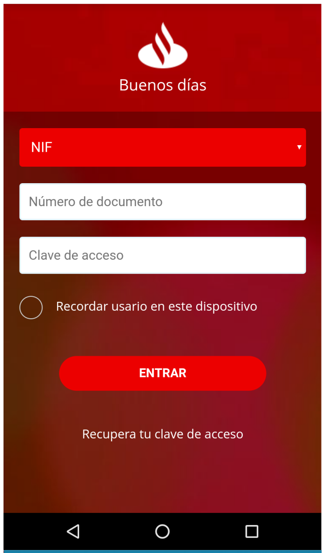
The initial version of Ginp had a generic credit card grabber overlay screen used for all targeted applications. Still included in the last versions, this screen is only used to overlay the official Google Play Store app. More apps could be added to the grabber target list in the future, such as the ones that were targeted in older versions:

The following screenshot shows the generic card grabber overlay screen:


Enter card details
VISA
MasterCard
AMERIKAN EXPRESS
Diners Club INTERNATIONAL
DISCOVER
JCB
Card number
MM/YY
CVC
Google Play
SAVE



The current active target list is available in the appendix, containing a total of 24 unique targets. The following screenshots show what type of information is collected in both steps of the overlay attack:



Buenos días

NIF ▾

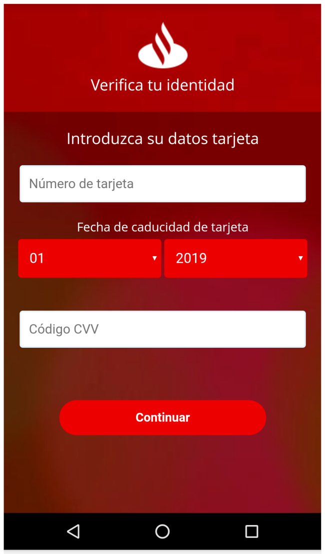
Número de documento

Clave de acceso

○ Recordar usario en este dispositivo

ENTRAR

Recupera tu clave de acceso



Verifica tu identidad

Introduzca su datos tarjeta

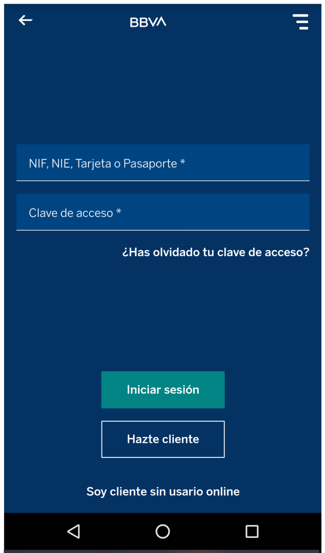
Número de tarjeta

Fecha de caducidad de tarjeta

01 ▾ 2019 ▾

Código CVV

Continuar



BBVA

NIF, NIE, Tarjeta o Pasaporte *

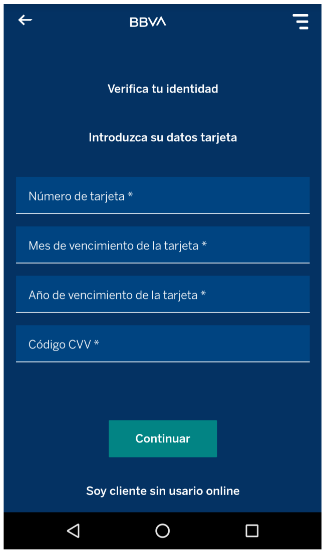
Clave de acceso *

¿Has olvidado tu clave de acceso?

Iniciar sesión

Hazte cliente

Soy cliente sin usario online



BBVA

Verifica tu identidad

Introduzca su datos tarjeta

Número de tarjeta *

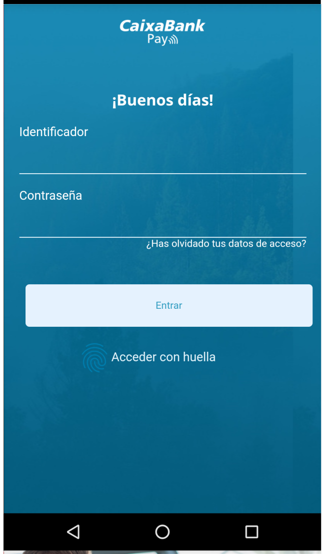
Mes de vencimiento de la tarjeta *

Año de vencimiento de la tarjeta *

Código CVV *

Continuar

Soy cliente sin usario online



CaixaBank Pay

¡Buenos días!

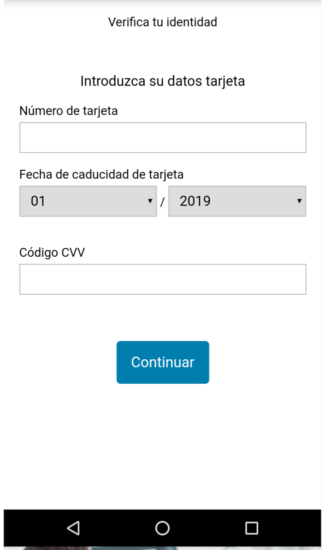
Identificador

Contraseña

¿Has olvidado tus datos de acceso?

Entrar

Acceder con huella



Verifica tu identidad

Introduzca su datos tarjeta

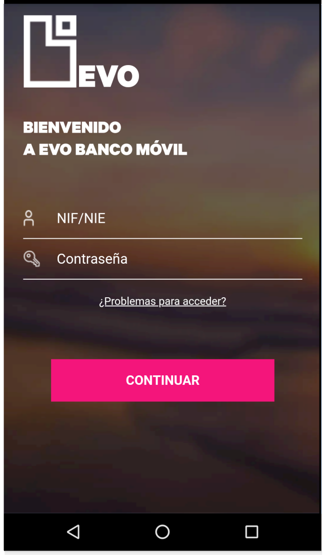
Número de tarjeta

Fecha de caducidad de tarjeta

01 ▾ / 2019 ▾

Código CVV

Continuar



EVO

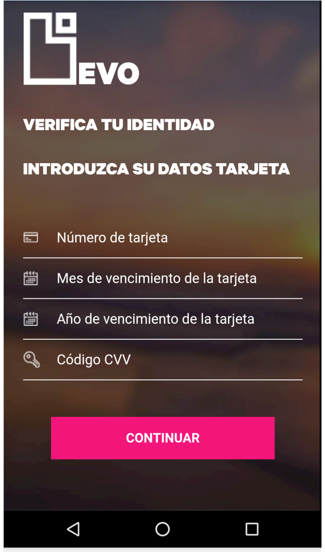
BIENVENIDO
A EVO BANCO MÓVIL

NIF/NIE

Contraseña

¿Problemas para acceder?

CONTINUAR



EVO

VERIFICA TU IDENTIDAD

INTRODUZCA SU DATOS TARJETA

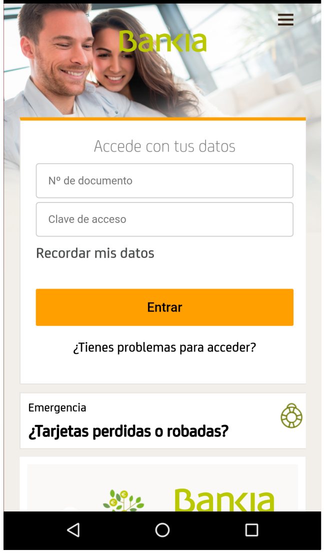
Número de tarjeta

Mes de vencimiento de la tarjeta

Año de vencimiento de la tarjeta

Código CVV

CONTINUAR



Bankia

Accede con tus datos

Nº de documento

Clave de acceso

Recordar mis datos

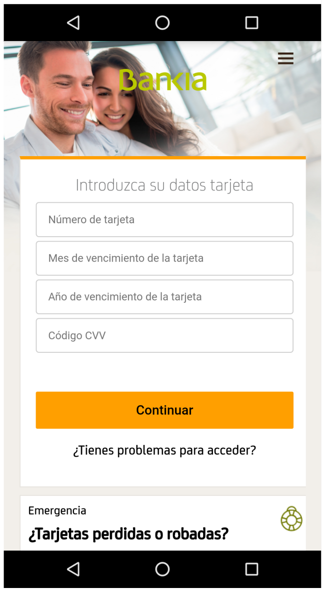
Entrar

¿Tienes problemas para acceder?

Emergencia

¿Tarjetas perdidas o robadas?

Bankia



Bankia

Introduzca su datos tarjeta

Número de tarjeta

Mes de vencimiento de la tarjeta

Año de vencimiento de la tarjeta

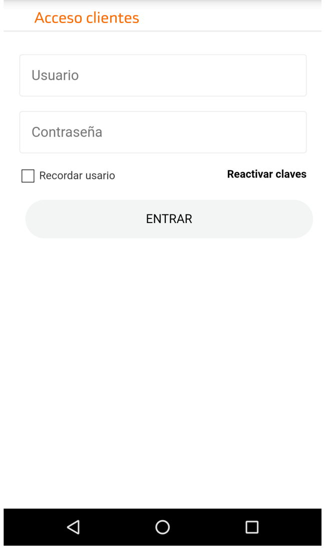
Código CVV

Continuar

¿Tienes problemas para acceder?

Emergencia

¿Tarjetas perdidas o robadas?



Acceso clientes

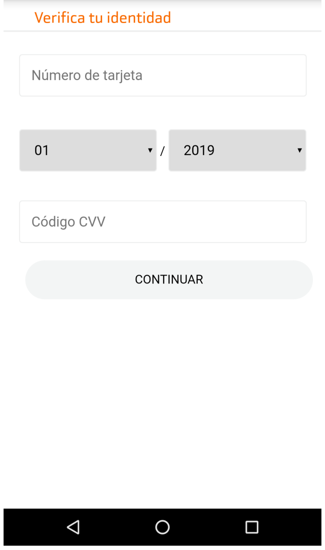
Usuario

Contraseña

□ Recordar usario

Reactivar claves

ENTRAR



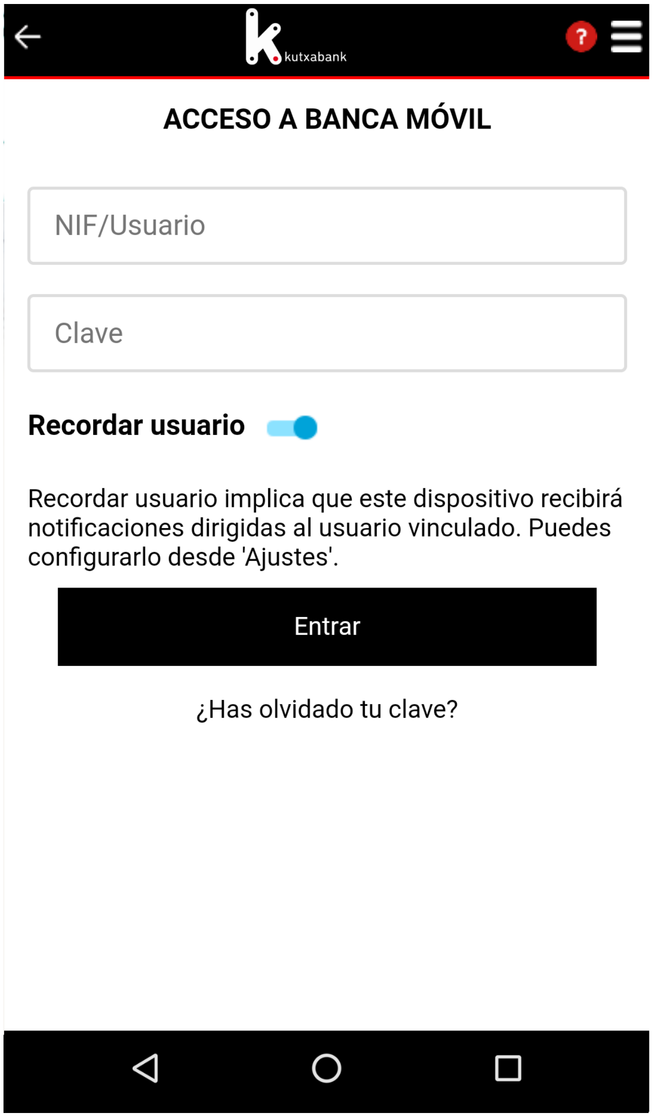
Verifica tu identidad

Número de tarjeta

01 ▾ / 2019 ▾

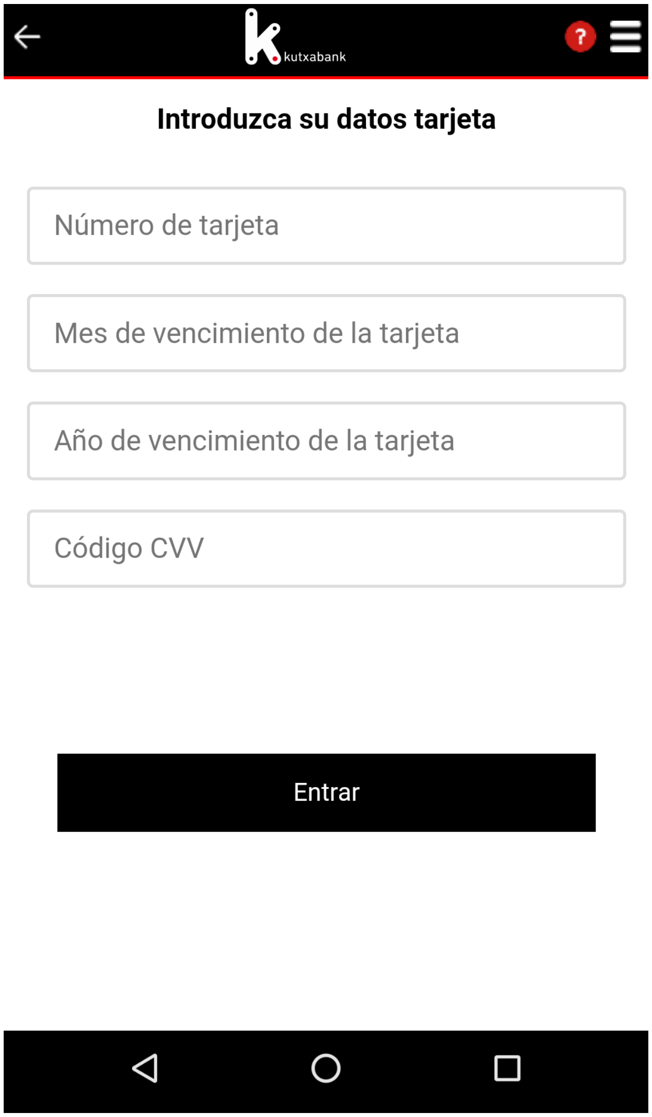
Código CVV

CONTINUAR



ACCESO A BANCA MÓVIL

Recordar usuario implica que este dispositivo recibirá notificaciones dirigidas al usuario vinculado. Puedes configurarlo desde 'Ajustes'.

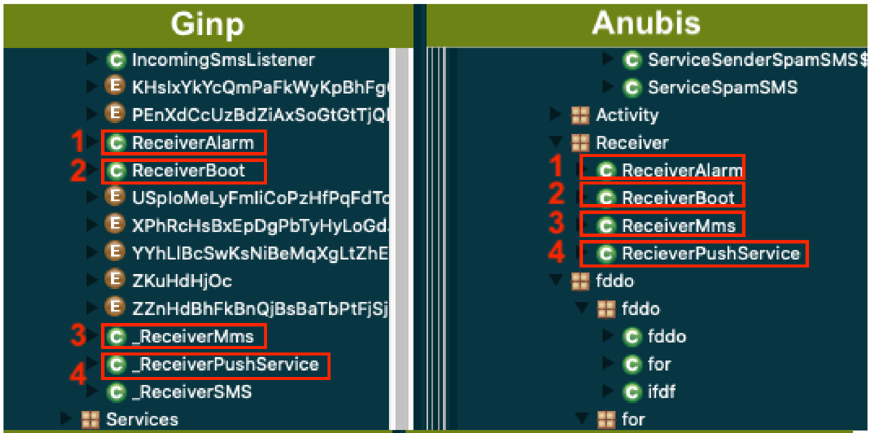

kutxabank
Número de tarjeta
Mes de vencimiento de la tarjeta
Año de vencimiento de la tarjeta
Código CVV
Entrar

Introduzca su datos tarjeta

Based on Anubis

Once the Anubis bot code got leaked, it was just a matter of time before new banking Trojans based on Anubis would surface. When analyzing the Ginp's recent samples, ThreatFabric analysts found some similarities with the famous Android banking Trojan. Based on the evolution of Ginp it is clear that it isn't based on Anubis, but rather reuses some of its code. Below are some of the elements showing the relation.

The names used for Android components are similar:

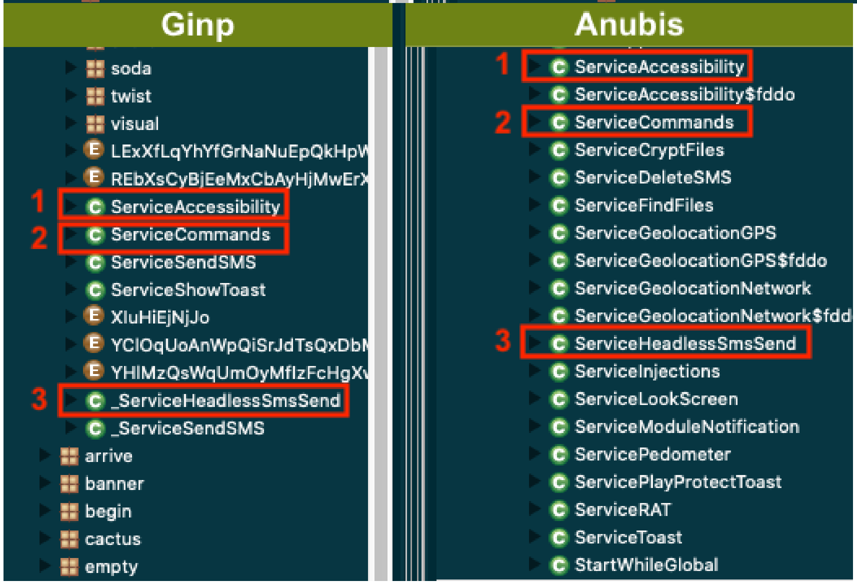


Ginp

```
IncomingSmsListener
KHslxYkYcQmPaFkWyKpBhFg
PEnXdCcUzBdZiAxSoGtGtTjQ
1 ReceiverAlarm
2 ReceiverBoot
USploMeLyFmliCoPzHfPqFdTc
XPhRcHsBxEpDgPbTyHyLoGd
YYhLlBcSwKsNiBeMqXgLtZhE
ZKuHdHjOc
ZZnHdBhFkBnQjBsBaTbPtFjSj
3 _ReceiverMms
4 _ReceiverPushService
_ReceiverSMS
Services
```

Anubis

```
ServiceSenderSpamSMS$
ServiceSpamSMS
Activity
Receiver
1 ReceiverAlarm
2 ReceiverBoot
3 ReceiverMms
4 RecieverPushService
fddo
fddo
fddo
for
ifdf
for
```



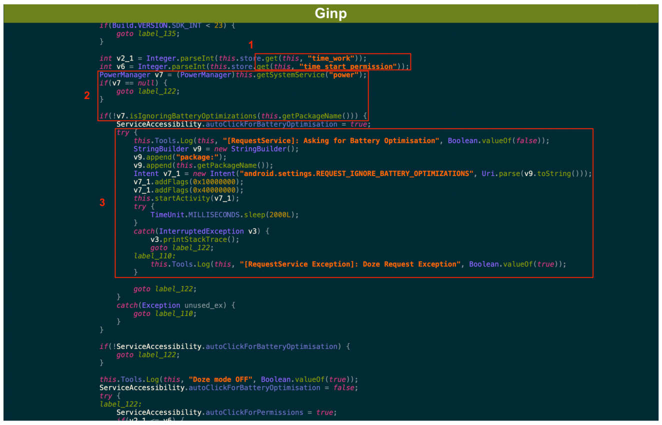
Ginp

```
soda
twist
visual
LExXfLqYhYfGrNaNuEpQkHpW
REbXsCyBjEeMxCbAyHjMwErX
1 ServiceAccessibility
2 ServiceCommands
ServiceSendSMS
ServiceShowToast
XluHiEjNjJo
YClOqUoAnWpQiSrJdTsQxDbM
YHlMzQsWqUmOyMflzFcHgXv
3 _ServiceHeadlessSmsSend
_ServiceSendSMS
arrive
banner
begin
cactus
empty
```

Anubis

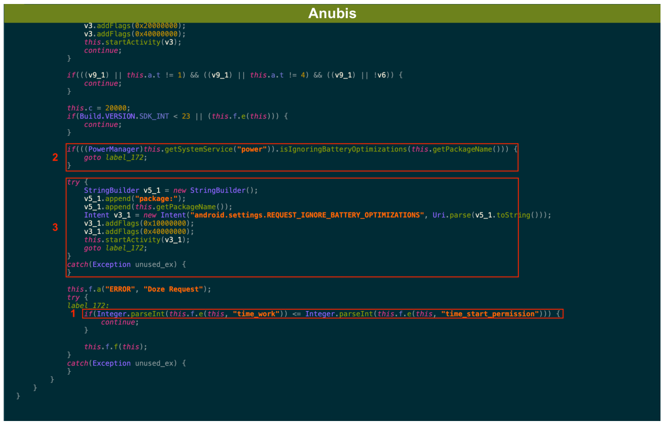
```
1 ServiceAccessibility
ServiceAccessibility$fddo
2 ServiceCommands
ServiceCryptFiles
ServiceDeleteSMS
ServiceFindFiles
ServiceGeolocationGPS
ServiceGeolocationGPS$fddo
ServiceGeolocationNetwork
ServiceGeolocationNetwork$fdd
3 ServiceHeadlessSmsSend
ServiceInjections
ServiceLookScreen
ServiceModuleNotification
ServicePedometer
ServicePlayProtectToast
ServiceRAT
ServiceToast
StartWhileGlobal
```

When analyzing these components, similarities were found in the code of both malware families:



Ginp

```
if(Build.VERSION.SDK_INT < 23) {
    goto label_135;
}

int v2_1 = Integer.parseInt(this.store.get(this, "time_work"));
int v6 = Integer.parseInt(this.store.get(this, "time_start_permission"));
PowerManager v7 = (PowerManager)this.getSystemService("power");
if(v7 == null) {
    goto label_122;
}

if(!v7.isIgnoringBatteryOptimizations(this.getPackageName())) {
    ServiceAccessibility.autoClickForBatteryOptimisation = true;
    try {
        this.Tools.Log(this, "[RequestService]: Asking for Battery Optimisation", Boolean.valueOf(false));
        StringBuilder v9 = new StringBuilder();
        v9.append("package:");
        v9.append(this.getPackageName());
        Intent v7_1 = new Intent("android.settings.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS", Uri.parse(v9.toString()));
        v7_1.addFlags(0x10000000);
        v7_1.addFlags(0x40000000);
        this.startActivity(v7_1);
        try {
            TimeUnit.MILLISECONDS.sleep(2000L);
        }
        catch(InterruptedException v3) {
            v3.printStackTrace();
            goto label_122;
        label_110:
            this.Tools.Log(this, "[RequestService Exception]: Doze Request Exception", Boolean.valueOf(true));
        }

        goto label_122;
    }
    catch(Exception unused_ex) {
        goto label_110;
    }
}

if(!ServiceAccessibility.autoClickForBatteryOptimisation) {
    goto label_122;
}

this.Tools.Log(this, "Doze mode OFF", Boolean.valueOf(true));
ServiceAccessibility.autoClickForBatteryOptimisation = false;
try {
label_122:
    ServiceAccessibility.autoClickForPermissions = true;
    if(v2_1 <= v6) {
```



Anubis

```
            v3.addFlags(0x20000000);
            v3.addFlags(0x40000000);
            this.startActivity(v3);
            continue;
        }

        if(((v9_1) || this.a.t != 1) && ((v9_1) || this.a.t != 4) && ((v9_1) || !v6)) {
            continue;
        }

        this.c = 20000;
        if(Build.VERSION.SDK_INT < 23 || (this.f.e(this))) {
            continue;
        }

        if(((PowerManager)this.getSystemService("power")).isIgnoringBatteryOptimizations(this.getPackageName())) {
            goto label_172;
        }

        try {
            StringBuilder v5_1 = new StringBuilder();
            v5_1.append("package:");
            v5_1.append(this.getPackageName());
            Intent v3_1 = new Intent("android.settings.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS", Uri.parse(v5_1.toString()));
            v3_1.addFlags(0x10000000);
            v3_1.addFlags(0x40000000);
            this.startActivity(v3_1);
            goto label_172;
        }
        catch(Exception unused_ex) {
        }

        this.f.a("ERROR", "Doze Request");
        try {
        label_172:
            if(Integer.parseInt(this.f.e(this, "time_work")) <= Integer.parseInt(this.f.e(this, "time_start_permission"))) {
                continue;
            }

            this.f.f(this);
        }
        catch(Exception unused_ex) {
        }
      }
    }
  }
```

Another major change that indicated that the actor copied code from the Anubis Trojan is the way of handling configuration values. Previous versions were storing config values within the variables of a class, while the latest version is using SharedPreferences with some of the keys being identical to those used by Anubis:

- isAccessibility
- time_work
- time_start_permission
- url_inj

Conclusion

Ginp is a simple but rather efficient banking Trojan providing the basic functionality to be able to trick victims into delivering personal information. In a 5-month timespan, actor managed to create a Trojan from scratch which will presumably continue evolving offering new features such as keylogging, back-connect proxy or RAT capabilities.

Ginp’s unusual target selection is not just about its focus on Spanish banks but also the wide selection of targeted apps per bank. The fact that the overlay screens are almost identical to the legitimate banking apps suggests that the actors might be very familiar with the Spanish banking applications and might even be accustomed to the language.

Although the current target list is limited to Spanish apps, it seems that the actor is taking into account that the bot should also be able to target other countries, seeing that the path used in the inject requests contains the country code of the targeted institution. This could indicate that actor already has plans in expanding the targets to applications from different countries and regions.

Mobile Threat Intelligence

Our threat intelligence solution – MTI, provides the context and in-depth knowledge of the past and present malware-powered threats in order to understand the future of the threat landscape. Such intelligence, includes both the strategic overview on trends and the operational indicators to discern early signals of upcoming threats and build a future-proof security strategy.

Client Side Detection

Our online fraud detection solution – CSD, presents financial institutions with the real-time overview on the risk status of their online channels and related devices. This overview provides all the relevant information and context to act upon threats before they turn into fraud. The connectivity with existing risk or fraud engines allows for automated and orchestrated, round the clock fraud mitigation.

Appendix

Samples

Some of the latest Ginp samples found in the wild:

App name	Package name	SHA-256 hash
Google Play Verificator	sing.guide.false	0ee075219a2dfde018f17561467272633821d19420c08cba14322cc3b93bb5d5
Google Play Verificator	park.rather.dance	087a3beea46f3d45649b7506073ef51c784036629ca78601a4593759b253d1b7
Adobe Flash Player	ethics.unknown.during	5ac6901b232c629bc246227b783867a0122f62f9e087ceb86d83d991e92dba2f
Adobe Flash Player	solution.rail.forward	7eb239cc86e80e6e1866e2b3a132b5af94a13d0d24f92068a6d2e66cfe5c2cea
Adobe Flash Player	com.pubhny.hekzhgjty	14a1b1dce69b742f7e258805594f07e0c5148b6963c12a8429d6e15ace3a503c

App name	Package name	SHA-256 hash
Adobe Flash Player	sentence.fancy.humble	78557094dbabecdc17fb0edb4e3a94bae184e97b1b92801e4f8eb0f0626d6212

Target list

The current list of apps observed to be targeted by Ginp contains a total of 24 unique applications as seen below. This list is expected to grow in the future.

Package name	Application name
com.android.vending	Play Store
es.lacaixa.hceicon2	CaixaBank Pay: Mobile Payments
es.lacaixa.mobile.android.newwapicon	CaixaBank
es.caixabank.caixabanksign	CaixaBank Sign - Digital Coordinate Card
es.caixabank.mobile.android.tablet	CaixaBank Tablet
com.imaginbank.app	imaginBank - Your mobile bank
es.lacaixa.app.multiestrella	Family
com.bankinter.launcher	Bankinter Móvil
com.bankinter.bkwallet	Bankinter Wallet
com.bankinter.coincwallet	COINC Wallet
com.bankinter.bankintercard	bankintercard
es.cm.android	Bankia
com.bankia.wallet	Bankia Wallet
es.cm.android.tablet	Bankia Tablet
com.bbva.bbvacontigo	BBVA Spain
com.bbva.netcash	BBVA Net Cash | ES & PT
es.evobanco.bancamovil	EVO Banco móvil
com.redsys.bizum	EVO Bizum
com.kutxabank.android	Kutxabank
es.redsys.walletmb.app.kutxa.pro	KutxabankPay
es.bancosantander.apps	Santander
es.banconsantander.app.tablet	Santander Tablet
es.bancosantander.android.confirming	Confirming Santander
com.tm.sanstp	Santander Cash Nexus