Oscorp evolves into UBEL: an advanced Android malware spreading across the globe

By Federico Valentini, Francesco Iubatti

Archived: 2026-04-05 16:36:14 UTC

Key Points

- Back in February 2021 a smishing campaign was detected distributing **Oscorp**, a new Android malware at that time. The main goal of that campaign was stealing funds from the victims' home banking service, by combining the usage of phishing kits and vishing calls
- **Oscorp** has been developed to attack multiple financial targets (both banks and crypto currency apps) and its main features are the following:
  - Ability to send/intercept/delete SMS and make phone calls
  - Ability to perform Overlay Attacks for more than 150 mobile applications
  - VNC feature through WebRTC protocol and Android Accessibility Services
  - Enabling key logging functionalities
- Once **Oscorp** is successfully installed in the victim's device, it enables Threat Actors (TAs) to remotely connect to it via WebRTC protocol. In some cases, we found a specific Threat Actor (TA) leveraging on fake bank operators to persuade victims over the phone while performing unauthorized bank transfers in the background.
- After an apparent stop of the initial activities, during May 2021, new **Oscorp** samples have been found in the wild, with some minor changes; at the same time, on multiple hacking forums, a new Android botnet known as UBEL started being promoted.
- We found multiple indicators linking **Oscorp** and **UBEL** to the same malicious codebase, suggesting a fork of the same original project or just a rebrand by other affiliates, as its source-code appears to be shared between multiple TAs.

Overview

At the end of January 2021, a new Android malware started appearing and it was dubbed as **Oscorp** [1]. During February 2021, a new version of Oscorp was detected by Cleafy systems and after a couple of hours a first incident related to this threat was reported to us.

Thanks to the data retrieved plus an in-depth technical analysis of the distributed Oscorp samples we were able to reconstruct the detailed chain of events and share all the methodologies used by a specific TA for conducting bank frauds via ATO (Account Takeover fraud).

The following list include some of the high-level indicators we extracted in our recent analysis:

- EU retail banks appear to be among the targets of this specific TA, and multiple incidents have already been confirmed. Since the list of targets also includes banks and financial institutions from US, JP, AU (see

the affected countries in the Appendix 4) we don't exclude that other local TAs might be using the same attack vector (Oscorp) to carry over other malicious activities.

- Phishing campaigns were distributed via SMS messages (**smishing**), a common tactic nowadays for retrieving valid credentials and phone numbers
- A fake bank operator conducts attacks in real-time by persuading victims over the phone (**vishing**), a common tactic typically used for bypassing multi-factor authentication (e.g. OTP codes).
- Oscorp appears to be distributed by this TA for gaining full remote access to the infected mobile device and performing unauthorized bank transfers from the infected device itself, **drastically reducing their footprint since a new device enrollment is not required in this scenario**.
- Instant Payments appears to be the most popular cash-out mechanism mainly routed through a network of money mules. We don't exclude other cash-out mechanisms (e.g. virtual cards generation, prepaid cards recharge, card-less ATM, etc..) since those services are quite common on modern retail banks services

The following image shows the timeline of captured events describing how this TA managed to retrieve valid banking credentials via smishing and successfully deliver Oscorp to the victim device for performing an ATO fraud scenario directly from its infected device:

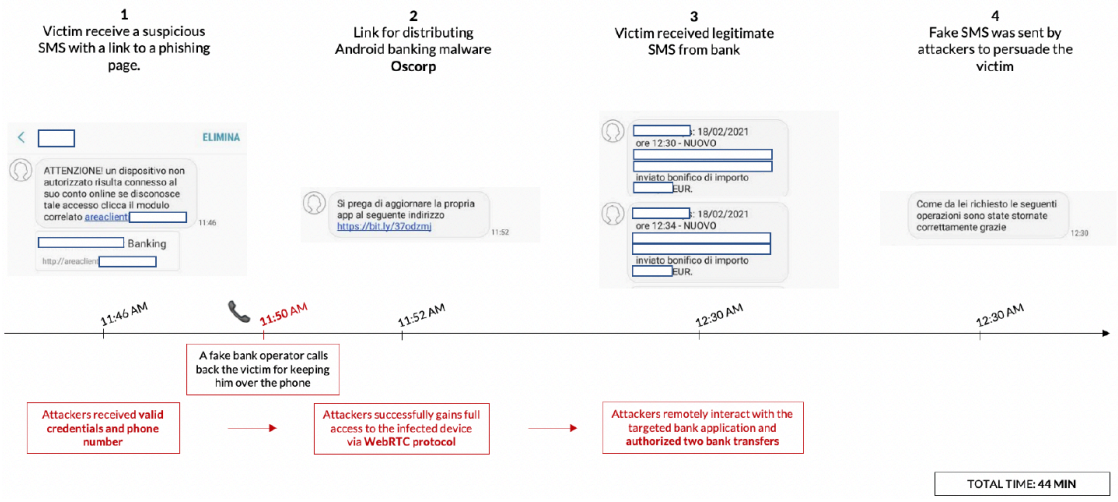

Figure 1 – Timeline of events retrieved from this new Oscorp campaign

Moving to the malware internals, we were able to extract multiple features of Oscorp which are mainly achieved by abusing the Android Accessibility services, a well-known technique used by the other families as well (e.g. Anubis, Cerberus/Alien, TeaBot [2],etc..).

The following snippet of code contains all the remote commands found in the Oscorp source code:

Figure 2 – List of Oscorp commands

All the commands are encrypted through an AES routine, a well-known technique used by malware authors for slowing down analysts.

The complete list of commands found in Oscorp is available on *Appendix 1*.

Oscorp evolves into UBEL

After an apparent stop of the initial activities, during May/June 2021, new Oscorp samples have been found in the wild, with some minor changes; at the same time, on multiple hacking forums, a new Android botnet known as **UBEL** started being promoted.

By analyzing some related samples, we found multiple indicators linking Oscorp and UBEL to the same malicious codebase, suggesting a fork of the same original project or just a rebrand by other affiliates, as its source-code appears to be shared between multiple TAs.

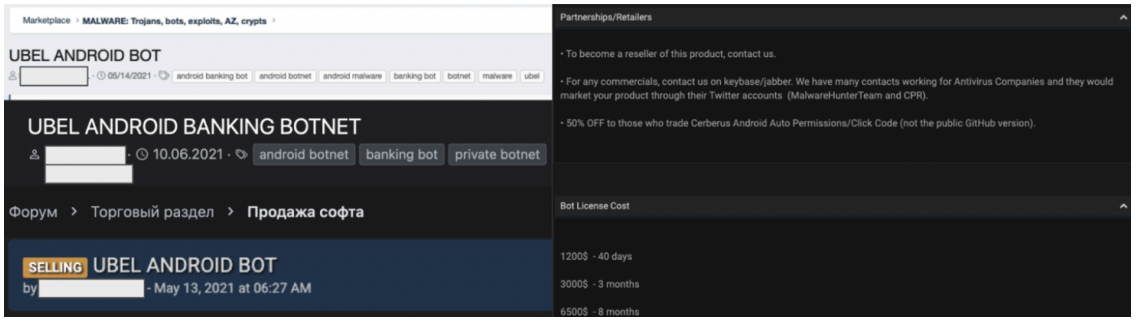

Figure 3 – UBEL private Android botnet threads found on multiple hacking forums

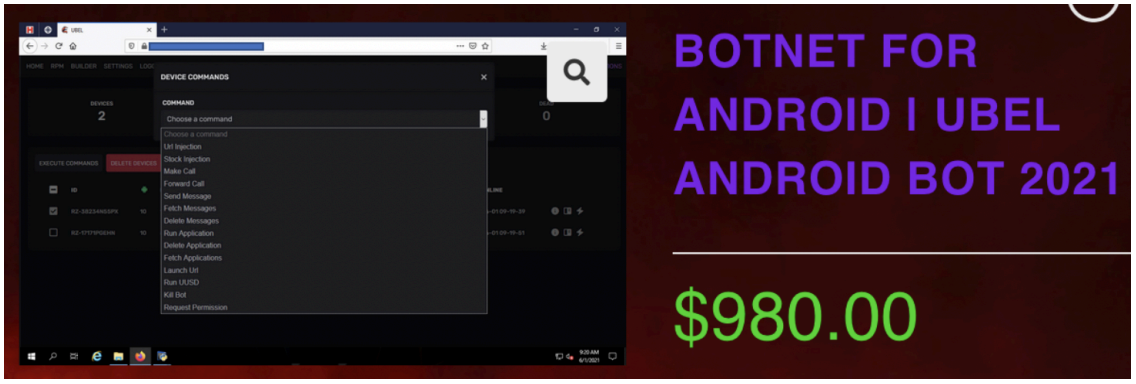

Figure 4 – Video demo of UBEL botnet and its C2 interaction

After a couple of weeks, we also noticed that the multiple UBEL clients started accusing them of scamming, as it appeared not to work on some specific Android devices, contrary to what the TA claimed initially.

One of those clients, after some debate, released some videos as proof of its claims without properly anonymize them, exposing a valid C2 addresses, as shown:

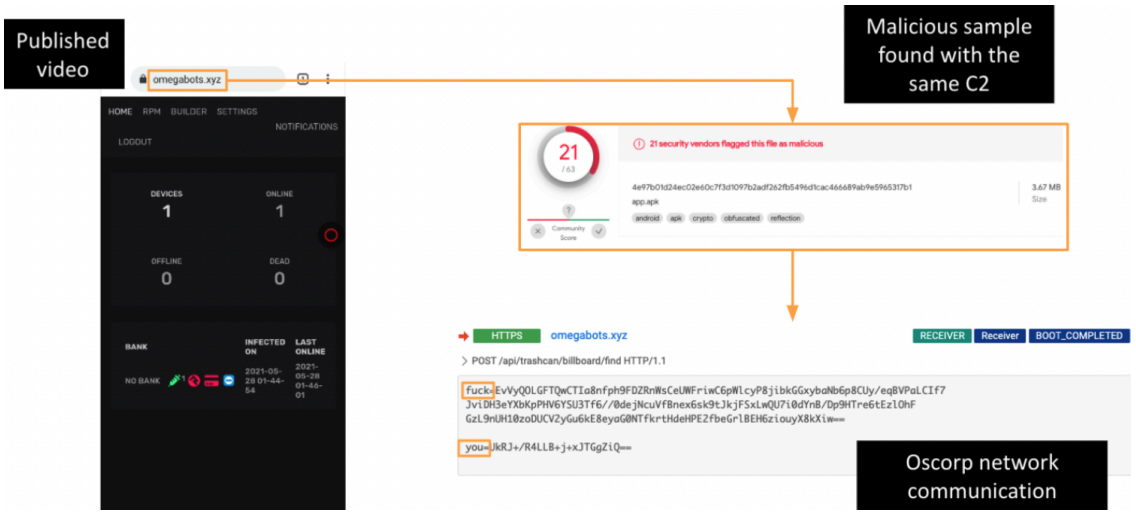

Figure 5 – Oscorp sample communicating with omegabots[.xyz

Another interesting links between Oscorp and UBEL, is the “bot id” string format, which consist in an initial “RZ-” substring followed by some random alphanumeric characters, as shown in another demo video posted online:

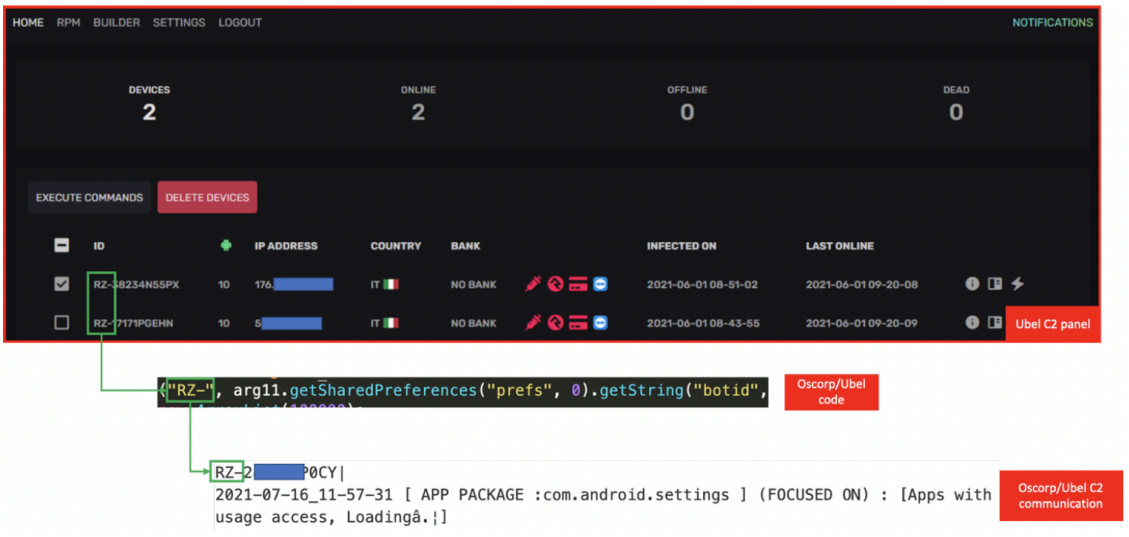

Figure 6 – Same “bot id” string prefix “RZ-” shared between Oscorp and UBEL

Also, on those newer Oscorp samples (linked to UBEL) we were able to identify different API endpoints and different AES keys compared to the initial waves spotted at the very first of 2021, which will be described in the next section.

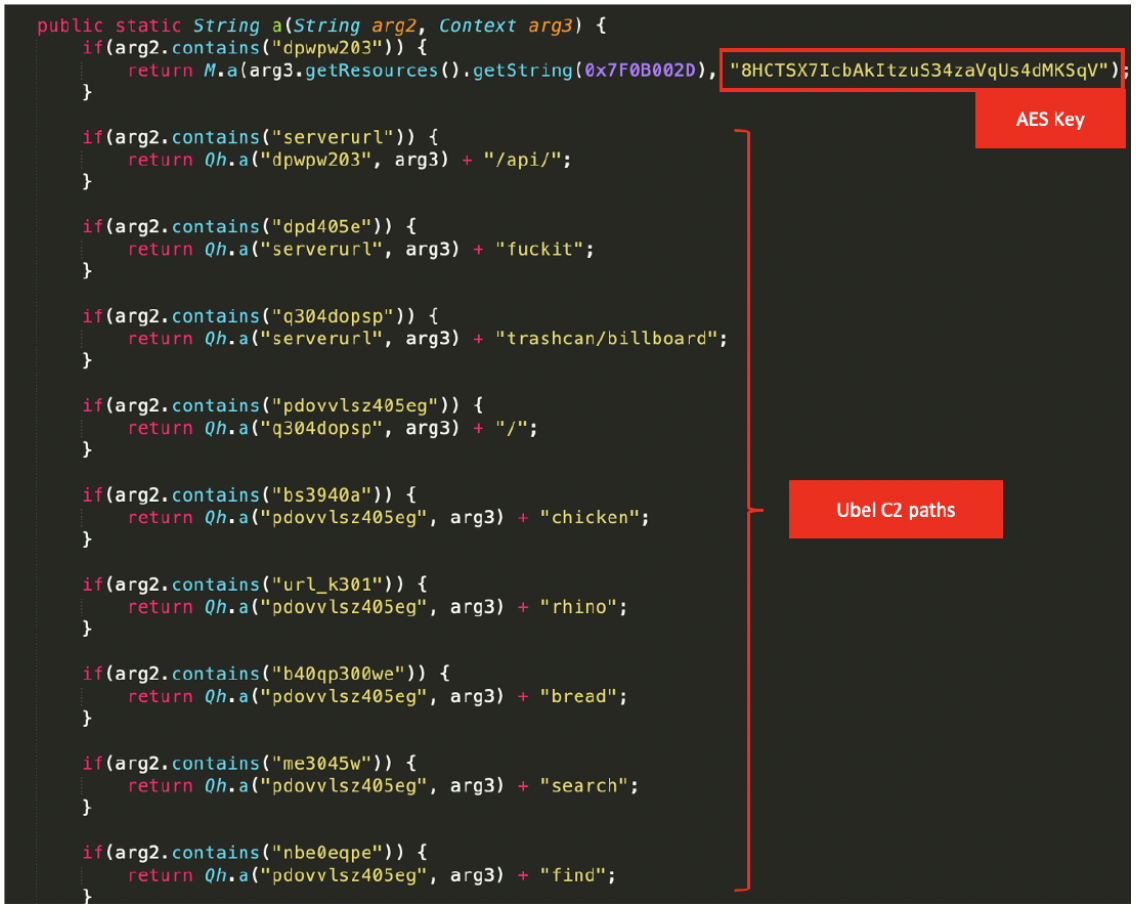

Figure 7 – Some new C2 path used by UBEL

Static Analysis

The following image shows a snippet of the **AndroidManifest** file:

Figure 8 – List of permissions declared in theAndroidManifest.xml file

In the following table we included the most interesting permissions requested by Oscorp for getting access to restricted parts of the Android system (e.g. **READ_SMS, SEND_SMS**) or other legitimate applications (e.g. **BIND_ACCESSIBILITY_SERVICE**):

- **SYSTEM_ALERT_WINDOW:** Allows an app to create windows shown on top of all other apps. Very few apps should use this permission; these windows are intended for system-level interaction with the user. Oscorp uses this permission during the installation phase to force the user to accept the Accessibility permission.
- **RECORD_AUDIO**: Allows an app to record audio
- **READ_SMS**: Allows an app to send SMS messages
- **SEND_SMS**:Allows an app to send SMS messages
- **RECEIVE_SMS**: Allows an app to receive SMS messages
- **REQUEST_INSTALL_PACKAGES**: Allows an application to request installing packages
- **REQUEST_DELETE_PACKAGES**: Allows an application to request deleting packages
- **RECEIVE_BOOT_COMPLETED:** Allows an app to launch itself automatically after system boot. Oscorp uses this permission to achieve persistence on the device and run in the background as an Android service.
- **BIND_ACCESSIBILITY_SERVICE**: *"Accessibility services should only be used to assist users with disabilities in using Android devices and apps. They run in the background and receive callbacks by the system when AccessibilityEvents are fired. Such events denote some state transition in the user interface, for example, the focus has changed, a button has been clicked, etc. Such a service can optionally request the capability for querying the content of the active window."*[3] However, Oscorp abused this permission to observe and retrieve information on the compromised device

Oscorp implements a couple of techniques to slow down static analysis, such as:

- all the strings are obfuscated using an open-source implementation [4] but some strings (e.g. bot's commands, API endpoints, etc.) are also encrypted with AES and base64 encoding.
- network communication between Oscorp and C2 are encrypted using only the AES algorithm and base64 encoding on top of regular HTTP(s).

Figure 9 – Oscorp encryption routine

Moreover, strings obfuscation appears to be introduced only on certain samples of Oscorp[5], sharing the same routine used by Cabassous (Flubot), another Android banking malware.

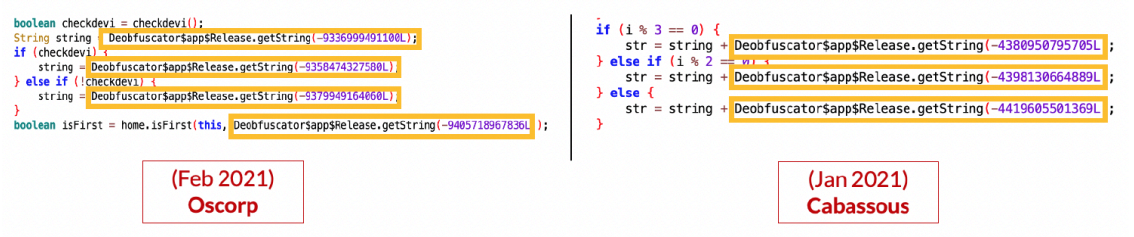

Figure 10 – Comparing encryption routines of Oscorp and Cabassous

Figure 11 – Network traffic encryption routines (AES algorithm)

WebRTC – Web Real-Time Communication

> *"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. The technologies behind WebRTC are implemented as an open web standard and available as regular JavaScript APIs in all major browsers. For native clients, like Android and iOS applications, a library is available that provides the same functionality." [6]*

We assume that Oscorp integrated WebRTC for achieving a real-time interaction with the compromised device combined with the abuse of Android Accessibility Services bypassing the need of a "new device enrollment" to perform an Account Take over scenario (ATO).

In fact, the authors named this feature as '**Reverse VNC**' (or RPM) on their C2 web-panel since a reverse connection is necessary for bypassing NAT or firewall restrictions and live interaction with the device can be achieved via Android Accessibility Services.

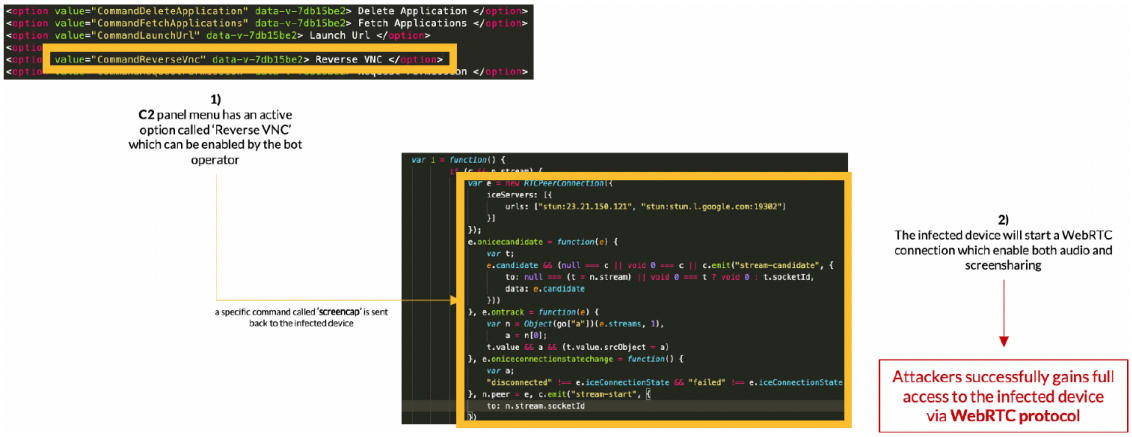

Figure 12 – 'Reverse VNC' function which enable WebRTC remote connection

The main goal for this TA by using this feature, is to avoid a "new device enrollment", thus drastically reducing the possibility of being flagged 'as suspicious' since device's fingerprinting indicators are well-known from the bank's perspective.

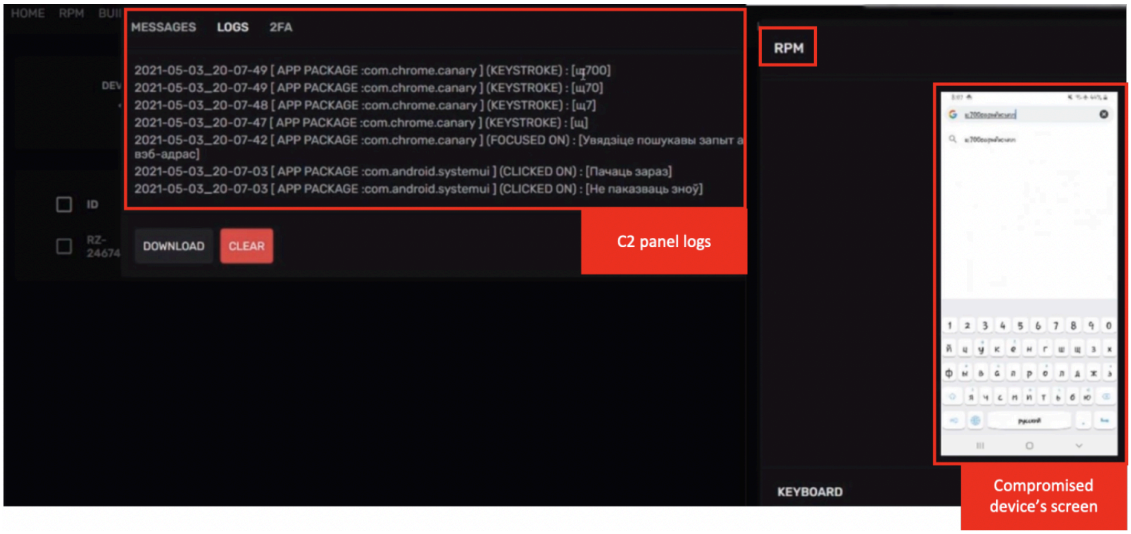

Figure 13 – Views of the UBEL C2 panel during a “Reverse VNC/RPM” attack

Dynamic Analysis

When the malicious application has been downloaded on the device, it tries to be **installed as an “Android Service”**, which is an application component that can perform long-running operations in the background.

This feature is abused by the Oscorp to silently hide itself from the user, once installed, also preventing detection, and ensuring its persistence.

During some campaigns spotted early in 2021, they switched the name of the malicious application from “Android System” to “Protezione Clienti” app (Figure 15):

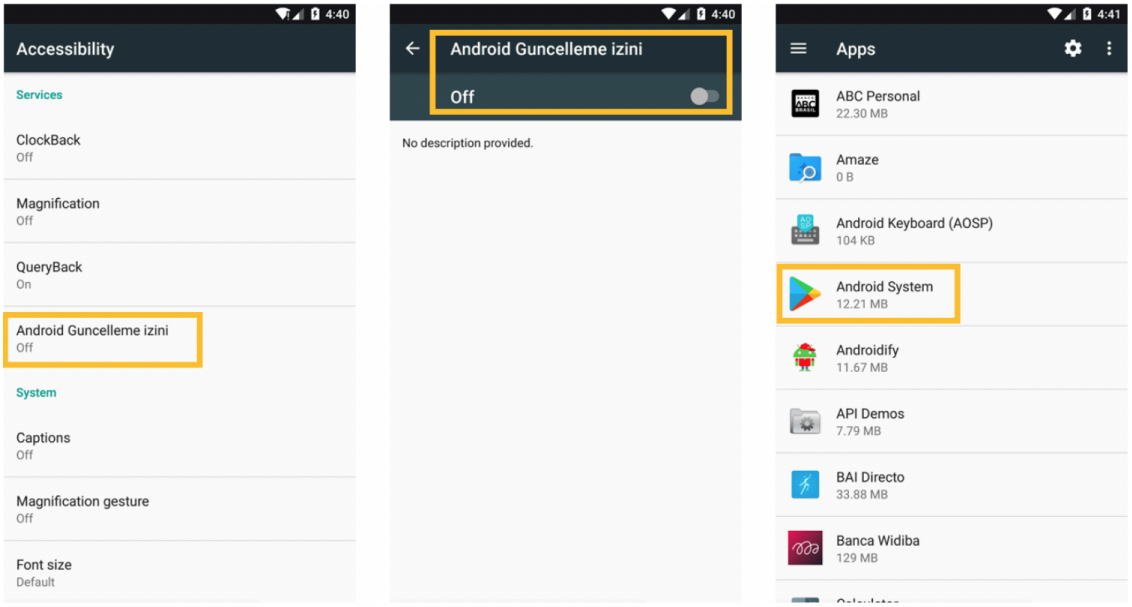

Figure 14 – Screenshots taken during installation phase of Oscorp

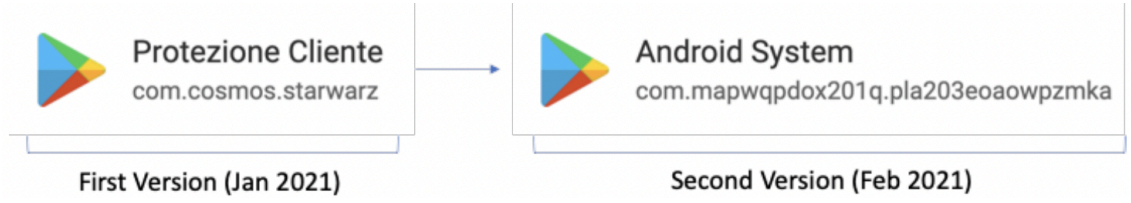

Figure 15 – Application names used by Oscorp

After the installation as “Android Service”, Oscorp will request the following permissions, which are mandatory to perform its malicious behavior:

- **Observe your actions:**
  Used to intercept and observe the user action.
- **Retrieve window content:**
  Used to retrieve sensitive information such as login credentials, SMS, two factor authentication (2FA) codes from authenticators, etc.

- **Perform arbitrary gestures:**
  Oscorp uses this feature to accept different kinds of permissions, immediately after the installation phase, for example the REQUEST_IGNORE_BATTERY_OPTIMIZATIONS permission popup, but also perform different actions during the **interaction with the compromised device through the WebRTC protocol.**

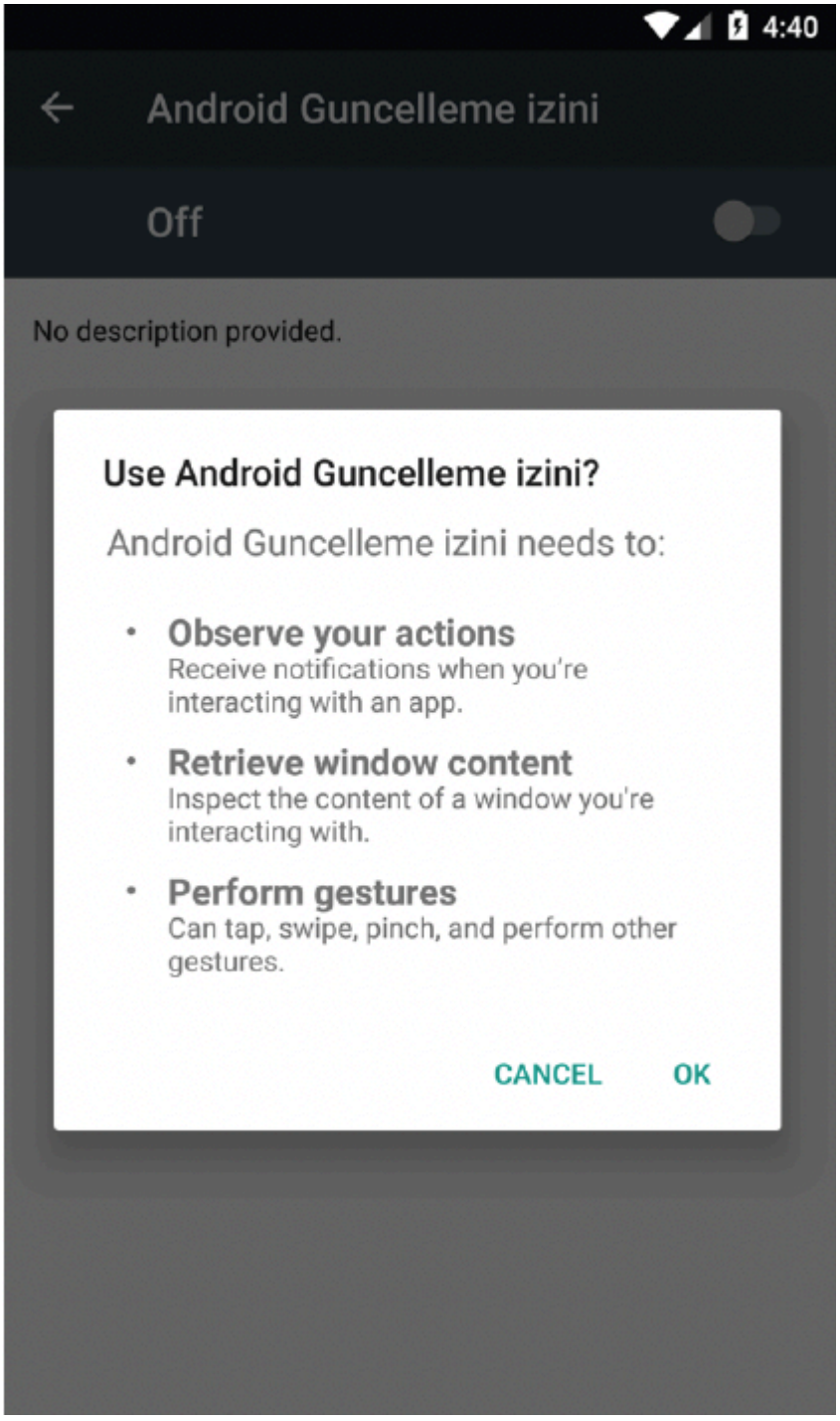

Figure 16 – List of Android permissions requested by Oscorp to the user

Once the requested permissions have been accepted, the malicious application will remove its icon from the device, and it immediately starts communicating with its C2 server in the background.

Network communications performed by Oscorp to its C2 server are encrypted with the AES algorithm and at the very first it tries to send all overall information of the newly infected device, such as vendor, public IP address, list of the installed apps, SMS messages, action performed by the user, etc.

The next figure is an example of a communication intercepted between Oscorp and its C2 server where the list of all the installed application was sent:

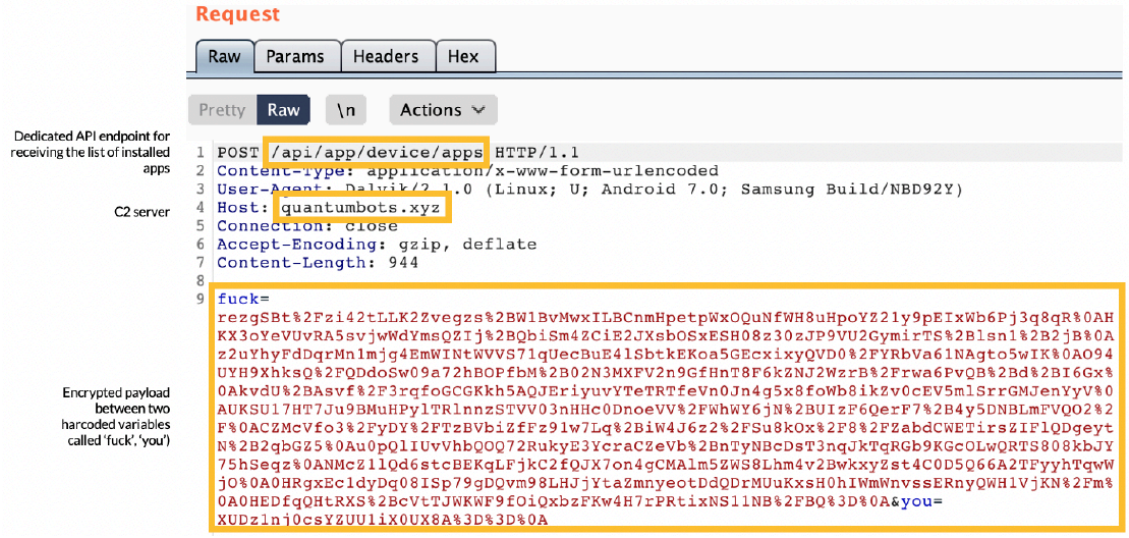

Figure 17 – Example of network communication intercepted between Oscorp and its C2 server

Oscorp can also abuse the Android Accessibility Services to capture and retrieve whatever is on the screen of the device, for example:

- 2FA codes (e.g. OTP) generated by banking applications during login authentication and while signing new bank transfers (e.g. instant payments, SEPA transfers, etc.)
- Intercepting notification and SMS messages
- Performing Overlay attacks (described in the *Appendix 2)*
- Enabling a full interaction with the infected device (e.g. sending arbitrary clicks on screen, opening arbitrary applications already installed, etc.)

The following figure shows how a new SMS received will be intercepted by Oscorp and send back to the designed C2 server:

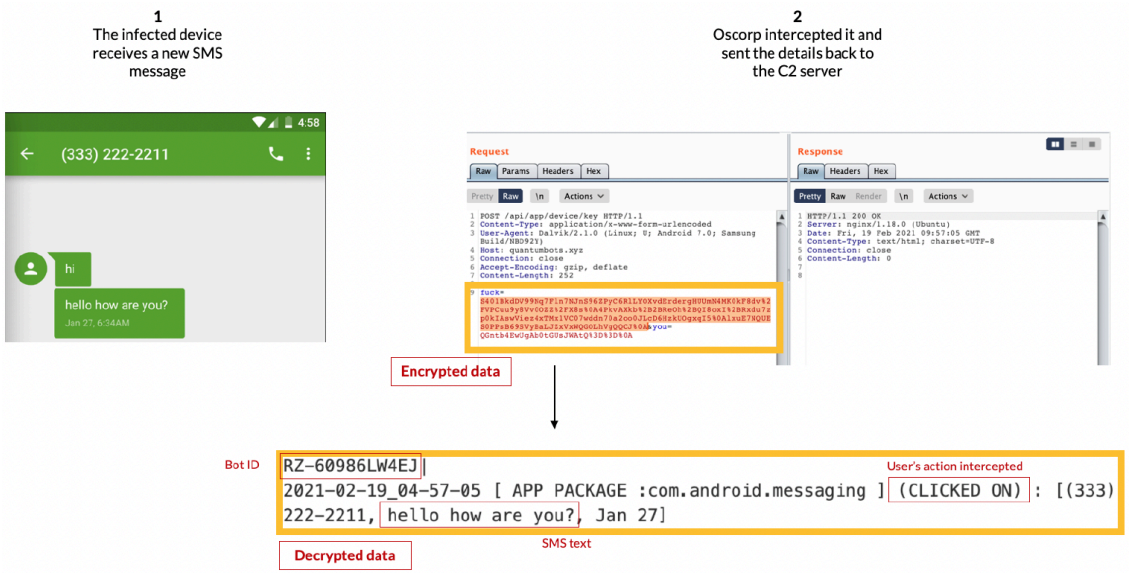

Figure18 – Example of SMS intercepted by Oscorp

Appendix

Appendix 1: list of bots' commands

Below is the summary list of all the bot commands found on Oscorp:

- ***toast***: Show a simple feedback about an operation in a small popup.
- ***send_message***: Send an SMS message
- ***stock_injection***: Save the injections (phishing html payload) provided by C2 in the Jedi / Injections.txt file
- ***forward_call***: Call forwarding through the code *21* + number + ##
- ***run_application***: Run an application
- ***enab_sil***: Mute the device (set to 0 the volume level of the device)
- ***switch_sms***: Change the default SMS application with Oscorp (through android.provider.Telephony.ACTION_CHANGE_DEFAULT)
- ***remove_injection***: Remove an injection
- ***2FA***: Launch the Google 2FA app (then Oscorp is able to steal the codes abusing the Accessibility service)
- ***make_call***: Perform a call to someone
- ***dev_admin***: Set itself as admin app
- ***run_ussd***: Allows itself to initiate a phone call without going through the Dialer user interface for the user to confirm the call
- ***block***: Save the apps to be blocked in Jedi / block.txt and start MyService
- ***launch_url***: Launch and URL
- ***fetch_applications***: Get the list of installed apps
- ***delete_message***: Remove an SMS
- ***delete_application***: Remove an application
- ***batt_opt***: Insert Oscorp app to a list of apps that ignore optimization battery
- ***url_injection***: Start the "ramp" class used to perform stream video of the screen and audio of the compromised device

- ***screencap***: start to record the audio and video through the **WebRTC** and **STUN** protocols (the stun server are embedded in the code)

Appendix 2: Overlay Attack's technique

"The Overlay attack is a well-known technique implemented on modern Android banking trojans (e.g. Anubis, Cerberus/Alien) which consist of a malicious application somehow able to perform actions on behalf of the victim. This usually takes the form of an imitation app or a WebView launched "on-top" of a legitimate application (such as a banking app)."

During our analysis we were able to extract more than 150 targeted applications.

The complete list of the geographical distribution of banks and other app targeted by Oscorp targeted apps is available in the Appendix 4.

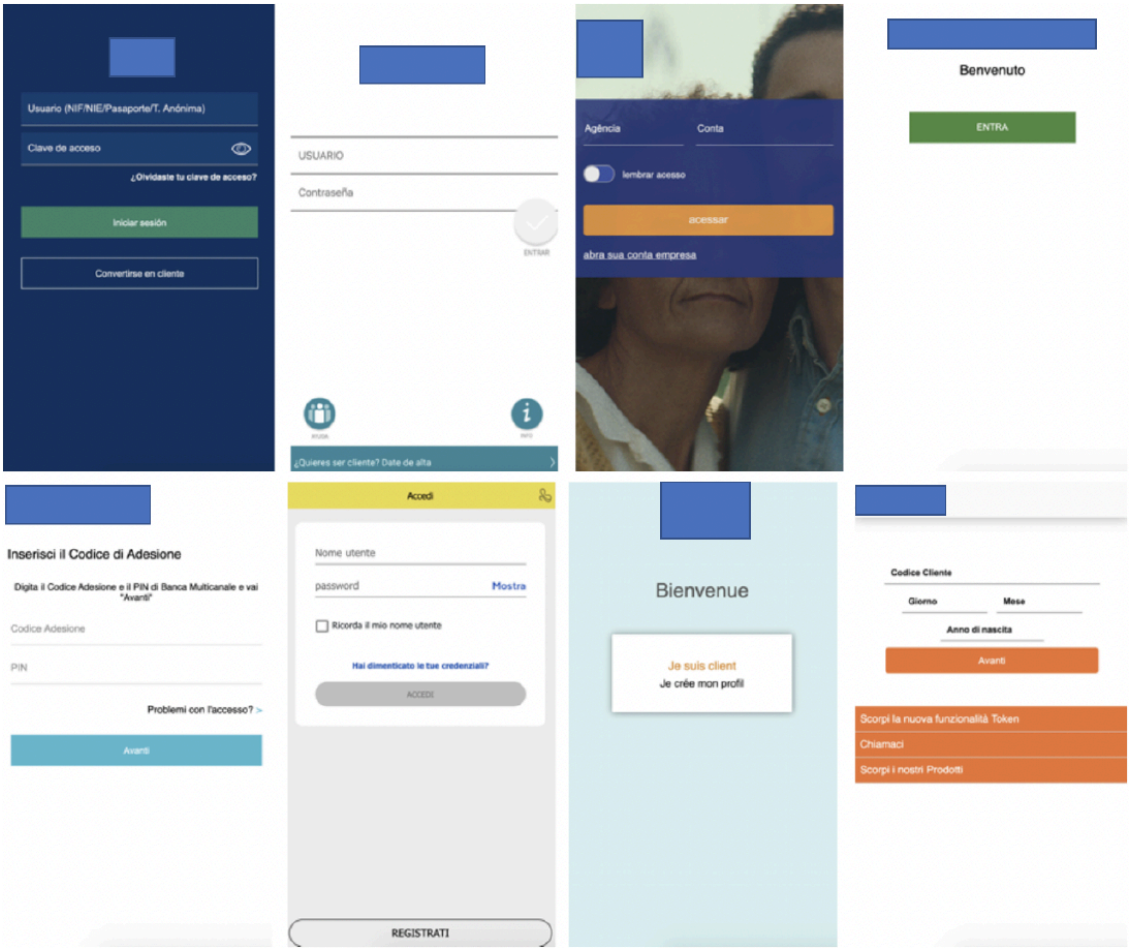

Figure 19 – Some payloads used by Oscorp for "Overlay Attacks"

All the injections payloads which consist mainly of HTML, CSS and JS files, will be downloaded from the C2 server in a specific directory called

_YTrJWNMmHkAPfdWA4QsfPwufCBhpYGbG.

When this feature is requested remotely by the TA, if the victim opens one of the targeted applications, it will get the injection payload shown in a WebView launched ‘on top’ of the legitimate application.

Figure 20 – C2 path used to download stock injections payload for Overlay Attacks

In addition, analyzing one of the web-panel used by this TA, it is also possible to reconstruct this distinction among the different categories of targeted applications, such as:

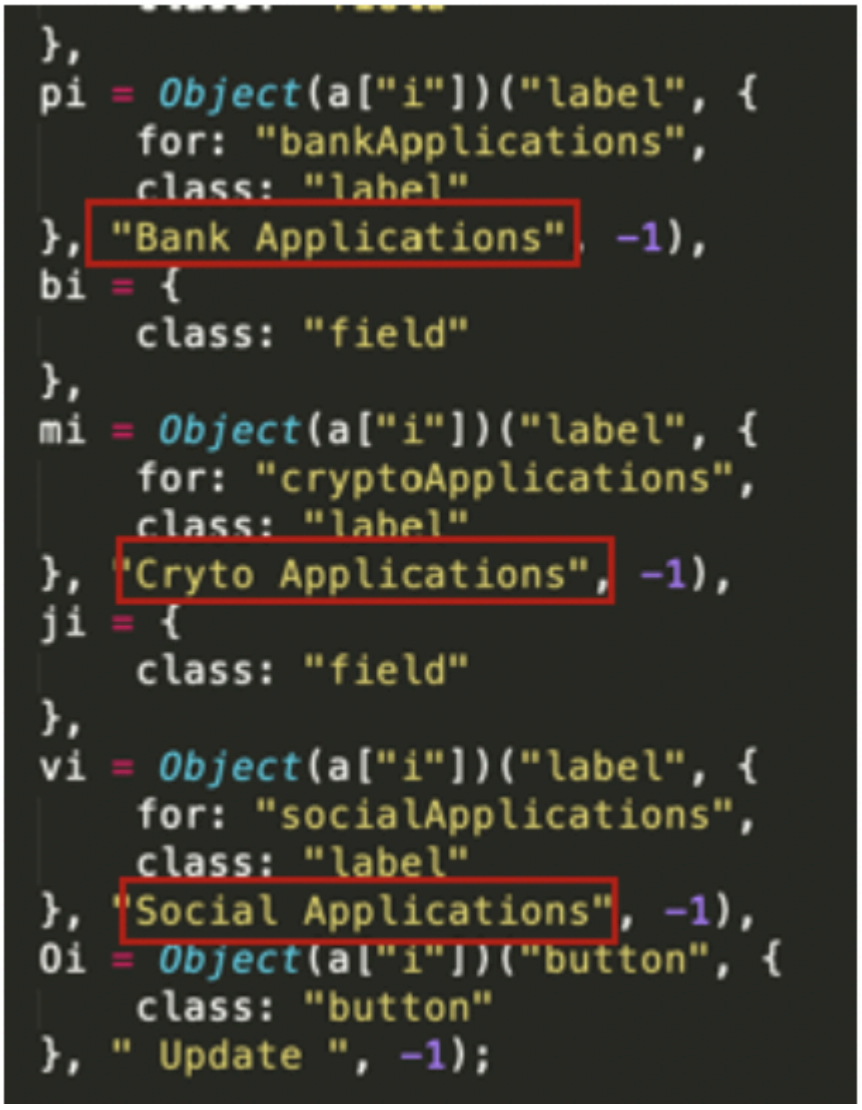

Figure 21 – Different types of targeted applications (Overlay Attacks)

Appendix 3: Extracted IOCs

Md5

0d1df5c35c3c43e1b8bb7daec2495c06
f73ebc6f645926bf8566220b14173df8
eaf0524ba3214b35a068465664963654
daba8377d281c48c1c91e2fa7f703511
1d848ba69a966f9f0ebe46bcb89a10c4
8daf9ba69c0dcf9224fd1e4006c9dad3
de51b859f41b6a9138285cf26a1fad84

App names

Protezione Cliente
Android System
deneme

Package names

com.cosmos.starwarz
com.cosmos.starwarz
com.mapwqpdox201q.pla203eoaowpzmka
ycpgmsxy.rqhfesas

C2 Domains

montanatony[.xyz
marcobrando[.xyz
quantumbots[.xyz
smoothcbots[.xyz
omegabots[.xyz
callbinary.xyz
gogleadser.xyz

Stock injection path

/_YTrJWNMmHkAPfdWA4QsfPwufCBhpYGbG/LFwbkjNthZk9jDtvADjnS7FyUPcjKPpb_/

AES keys

RHBuUXFEhkrbrHaYIZ6VYH3uNIBRnwTe
8HCTSX7IcbAkItzuS34zaVqUs4dMKSqV

In addition, The Android Banking Trojan **Oscorp/Ubel** is already classified and blacklisted in our Threat Intelligence data with the following tags:

- **ASK_BANKER_ANDROID_OSCORP_V1**
- **ASK_BANKER_ANDROID_OSCORP_V2**

Appendix 4: Geographical distribution of banks and other app targeted by Oscorp

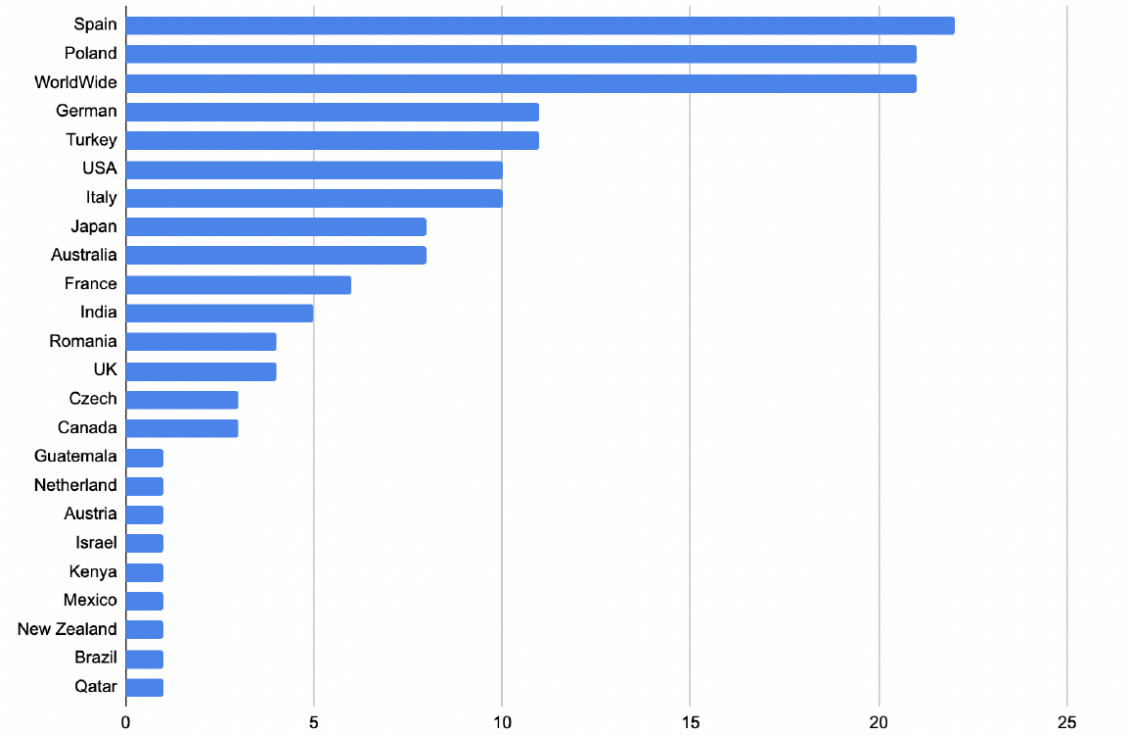

Figure 22 – Geographical distribution of banks and other app targeted by OSCORP

[1] https://cert-agid.gov.it/news/individuato-sito-che-veicola-in-italia-un-apk-malevolo/

[2] https://www.cleafy.com/cleafy-labs/teabot

[3] https://developer.android.com/reference/android/accessibilityservice/AccessibilityService

[4] https://github.com/MichaelRocks/paranoid

[5] Name:“secureapp.apk” MD5: daba8377d281c48c1c91e2fa7f703511

[6] https://webrtc.org/
