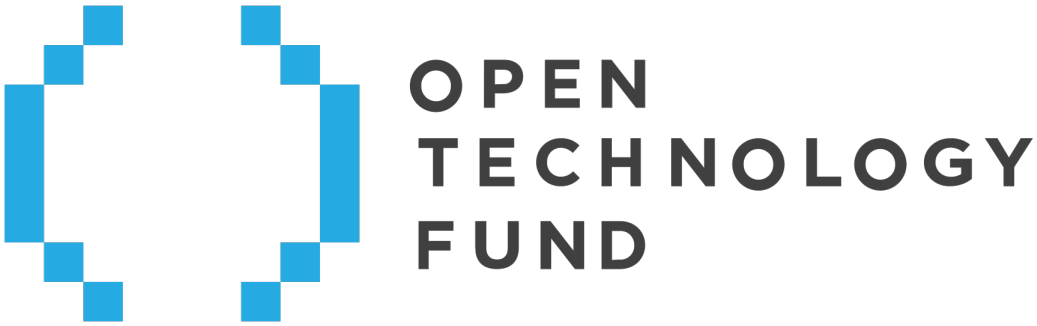
RIT | ESL Global Cybersecurity Institute



Partisan Telegram

Application and Operational Security Assessment

Prepared for:

- Open Technology Fund
- Sarah Aoun – Vice President of Security

Eaton Cybersecurity SAFE Lab Assessors:

- Robert Olson
- Jason Ross
- Forrest Fuqua

Table of Contents

Executive Summary

Project Overview

Partisan Telegram is an Android application intended for use by political dissidents to facilitate coordination using Telegram in regions controlled by hostile forces. Partisan Telegram presents itself as the traditional application known as Telegram but it includes additional functionality intended to support its mission, such as fake passcodes and profiles, automated deletion of data, notification of likely detainment, and remote wiping. It is intended to be resistant to casual inspection by technically unsophisticated opposition forces, but it is not intended to be resistant against dedicated forensic analysis.

The RIT SAFE Lab ("Contractor") was engaged to perform an operational and application security assessment of Partisan Telegram version 2.16.10 for its developers ("Client"). For security purposes, specifics of the Client's identity will not be included in this report. The assessment was scheduled to occur between February 28, 2022 and March 31, 2022. During this period, the Contractor experienced unforeseen delays at which time several project timeline extensions were signed and modified to allow for completion by July 31, 2022 at no additional cost to the Client. At the time this extension was provided, the Client requested that the version of P Telegram to be assessed be shifted to 2.16.8.

Test Scope

- Partisan Telegram Android App – The Contractor performed static and dynamic tests to assess application and operational security on version 2.18.6 of the Partisan Telegram Android application. This was retrieved from the following GitHub release: https://github.com/wrwrabbit/Partisan-Telegram-Android/releases/tag/2.18.6.
  Raw data collected from these tests will be provided separately from this report.

- Telegram Android Application – Version 8.7.0 of Telegram was utilized as a baseline against which Partisan Telegram was compared. This version of Telegram was retrieved from the popular APK hosting site APKPure who hosts that APK at the following link:
  https://m.apkpure.com/telegram/org.telegram.messenger/variant/8.7.0-APK

Assumptions

The Client indicated to RIT SAFE Lab that the primary purpose of Partisan Telegram is to improve the operational security of local activists by providing defense mechanisms

against casual phone searches by opposition forces, which should include service providers. Based on this description, the Contractor made several assumptions about what findings were relevant.

- Evidence of Partisan Telegram use or evidence that the Partisan Telegram client is not standard would be sufficient to present risk to Partisan Telegram users.
- Partisan Telegram is not intended to be resistant to intensive forensic analysis or secure on devices that allow privileged access to the underlying file system.[1]
- Findings that could be used to identify Partisan Telegram users should not be considered if they require that the device be rooted.
- Although Partisan Telegram is not intended to be resistant to intensive forensic analysis, forensic evidence that could be collected from the device using automated tools without privileged access or from network traffic that could be captured at scale by a service provider should be considered.
- Traditional application security findings that were introduced by Partisan Telegram code and were not a part of the Telegram code base are the focus of the test.

Testing Environment

Static security assessment was largely performed using automated code analyzers. MobSF[2], an open-source automated mobile application source code analyzer, was used to identify application security vulnerabilities in both Partisan Telegram 2.18.6 and Telegram 8.7.0. OWASP Dependency Check[3] was utilized on both applications to look for application components with known vulnerabilities. Finally, APKLeaks[4] was used to identify potential secrets - such as API keys - within both applications' code bases.

Several different environments were utilized for dynamic testing of Partisan Telegram and for comparing Partisan Telegram to the original version of Telegram. Testing occurred with Telegram and Partisan Telegram installed on both physical devices and virtual

[1] It would be trivial to identify Partisan Telegram users under these conditions if opposition forces have physical access to the mobile device on which Partisan Telegram is running.

[2] https://github.com/MobSF/Mobile-Security-Framework-MobSF

[3] https://owasp.org/www-project-dependency-check/

[4] https://github.com/dwisiswant0/apkleaks

machines. In addition to Telegram and Partisan Telegram, the Drozer[5] agent mobile application - used for viewing resources of Partisan Telegram as any other app could view them — was installed on all devices. The Drozer agent was not operational during any network traffic analysis.

Physical Android devices were utilized to assess Partisan Telegram under intended conditions. A Samsung Galaxy A13 was utilized to run Partisan Telegram and a Motorola Moto e5 Play was used to run Telegram. These devices were not rooted and were used to examine the applications without privileged access. The Samsung Galaxy ran Android 12 and the Moto e5 Play ran Android 8.0.

Android Studio Android Virtual Devices (AVD) were utilized for dynamic testing of the application and for analysis of the files created by Telegram and Partisan Telegram. The AVDs utilized were rooted to allow Contractor to fully explore the applications' file systems. Because the rooted devices were used, Google Play Services was not available

Finally, Genymotion Android virtual machines were used to facilitate network capture. Genymotion was selected to allow virtual devices to have direct network access through bridged virtual adapters. This allowed the Contractor to isolate the Android virtual machine's network traffic from the underlying host traffic. These devices were rooted. Google Play Services are not available on Genymotion Android virtual devices, but Open GApps was used to emulate these utilities.

Key Findings

SMS Permission and **App Disk Utilization**

The Contractor observed that the Partisan Telegram Android application requested SMS Permissions which were not requested by the standard version of Telegram. Additionally, the Contractor observed that Partisan Telegram occupies significantly more storage space than the standard version of Telegram. Both differences can be found using the standard Android settings menu. While there were multiple findings that could allow an observer to identify users of Partisan Telegram, these particular artifacts could be discovered by a casual observer with no technical knowledge and no specialized equipment.

Application Signing Key and Password in GitHub

[5] https://github.com/FSecureLABS/drozer

The Contractor observed that the Java keystore containing the key used to sign official release versions of Partisan Telegram could be found in two GitHub repositories. The passwords to that keystore and to the code signing key within were also found in those two GitHub repositories. With access to the code signing key, opposition forces could create a malicious version of Partisan Telegram and social engineer users into installing it. This is particularly likely as Partisan Telegram users are primed to download updates from social media or from GitHub. Additionally, this becomes more likely if Telegram opts or is coerced to cooperate with opposition forces, given that a Telegram channel is one distribution mechanism for these updates. To address this risk, it is recommended that the Client rotate their code signing key, make improvements to the associated certificate, change the passwords on the code signing key and the keystore containing it, and adjust development processes to ensure that it is not pushed to GitHub in the future.

Strategic Recommendations

Overall, the Contractor was impressed with the thoughtfulness that went into the development of Partisan Telegram. Partisan Telegram provides many features that would likely be very beneficial to political dissidents. As far as the Contractor observed, those features all performed as expected and the Contractor did not observe any additional application security vulnerabilities introduced by the addition of these features. The Contractor would also like to commend the Client for containing their operations to Telegram thus minimizing the attack surface of their mobile app. The Contractor observed few operational security leakages that could easily identify Partisan Telegram users on a large scale.

The Contractor, however, noted some operational security deficiencies. These deficiencies ranged from artifacts that may be easily identified via the Android user interface to issues with software development practices to potential clarifications needed within the Partisan Telegram documentation.

The Contractor found there were several possible scenarios where Partisan Telegram users may be identified. The Contractor finds it plausible that a casual inspection of a mobile device running Partisan Telegram 2.18.6 would indicate that a non-standard version of Telegram was in use. To the Contractor' knowledge, Partisan Telegram is the only 3rd party Telegram client that represents itself as Telegram. The Contractor also finds it plausible that a malicious low-privilege application installed on the mobile device would be able to identify that a non-standard version of Telegram is in use. Finally, the Contractor finds it plausible that a device where the user is forced or socially engineered into trusting an ADB connection would specifically indicate that Partisan Telegram is in use.

To address these risks, the Contractor recommends that Partisan Telegram developers first address issues related to their release code signing key to minimize the likelihood of users being socially engineered into installing a malicious version of Partisan Telegram. Second, the Contractor has recommended several changes to the application code base. The Contractor mentions several checks with warning messages that could be added to remind users to fix insecure settings and several mechanisms by which the differences between Telegram and Partisan Telegram may be minimized. Unfortunately, some points of exposure are likely risks that Partisan Telegram users must be willing to accept. Notably, it is likely that the application's disk utilization will always indicate a non-standard version of Telegram is in use. The Contractor also notes that it is very likely that some functionality will necessarily generate identifiable network traffic patterns.

The Contractor would recommend that the network traffic of Partisan Telegram be further studied. The Contractor performed a preliminary analysis to look for differences in network traffic between Telegram and Partisan Telegram over a 4-day period. The Contractor did not observe any anomalous communications during that 4-day period. There is concern, however, that there may be statistical differences in traffic bursts or over longer periods of time that might identify Partisan Telegram users at scale. The Contractor did not perform any statistical analysis on network traffic as it was out of scope for this assessment.

Similarly, the Contractor also recommends that the Client assess the potential risk related network correlation attacks. It is reasonably likely that users of Partisan Telegram are members of public Telegram channels known to be used for activism. It may be possible to correlate activity within these channels to network traffic being sent to Partisan Telegram users. This would be particularly significant if opposition forces were to provoke activity within these channels, such as through spam posting. Assessing the feasibility of network correlation attacks was determined to be outside of the scope of this engagement.

Technical Findings

The following table is a summary of the vulnerabilities identified during this security assessment. Subsequent pages of this report provide detail for each of the vulnerabilities, along with guidance on ways the risks can be mitigated.

High Risk Findings

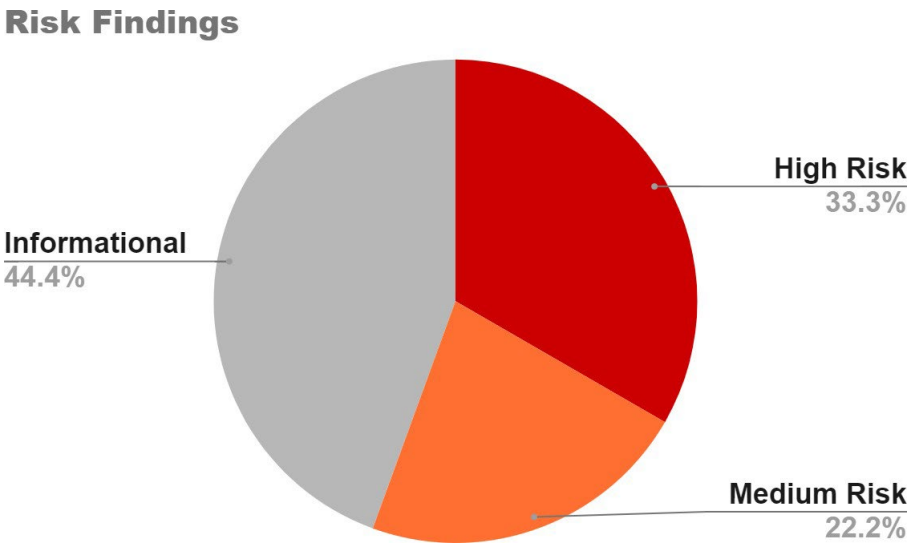
- PTelegram-High-1: SMS Permission
- PTelegram-High-2: App Disk Utilization
- PTelegram-High-3: Application Signing Key and Password in GitHub

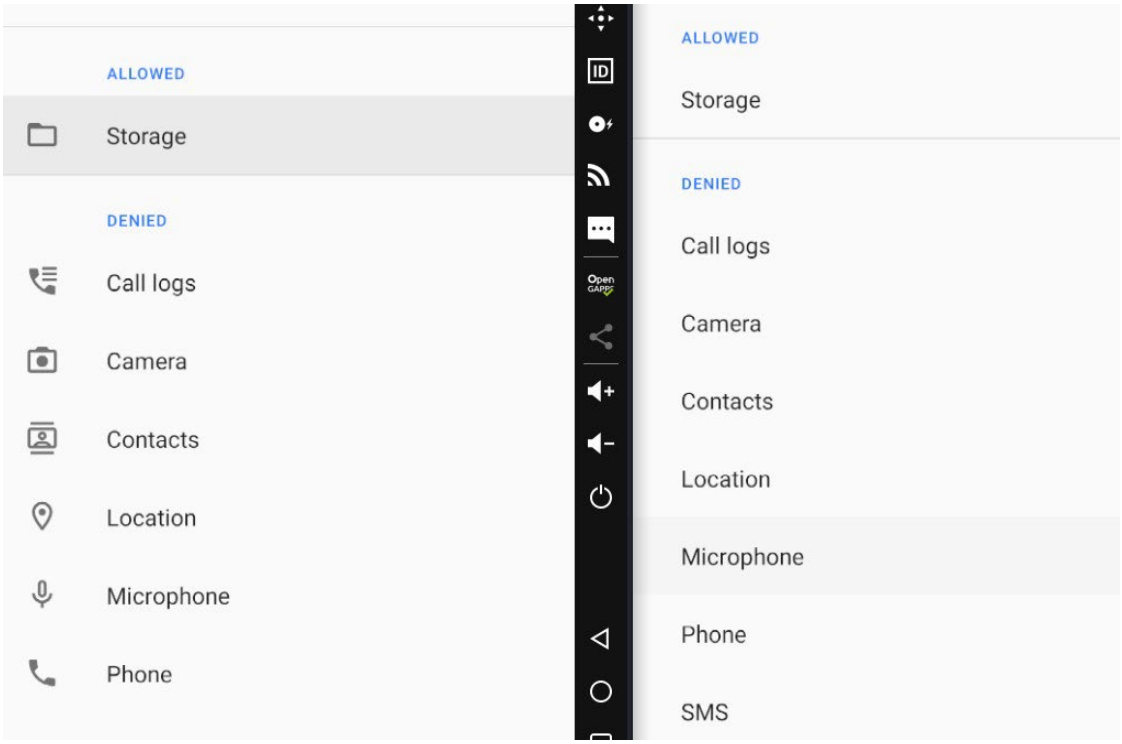
Medium Risk Findings

- PTelegram-Medium-1: Bundled Native Libraries
- PTelegram-Medium-2: Application Manifest Differences

Informational Findings

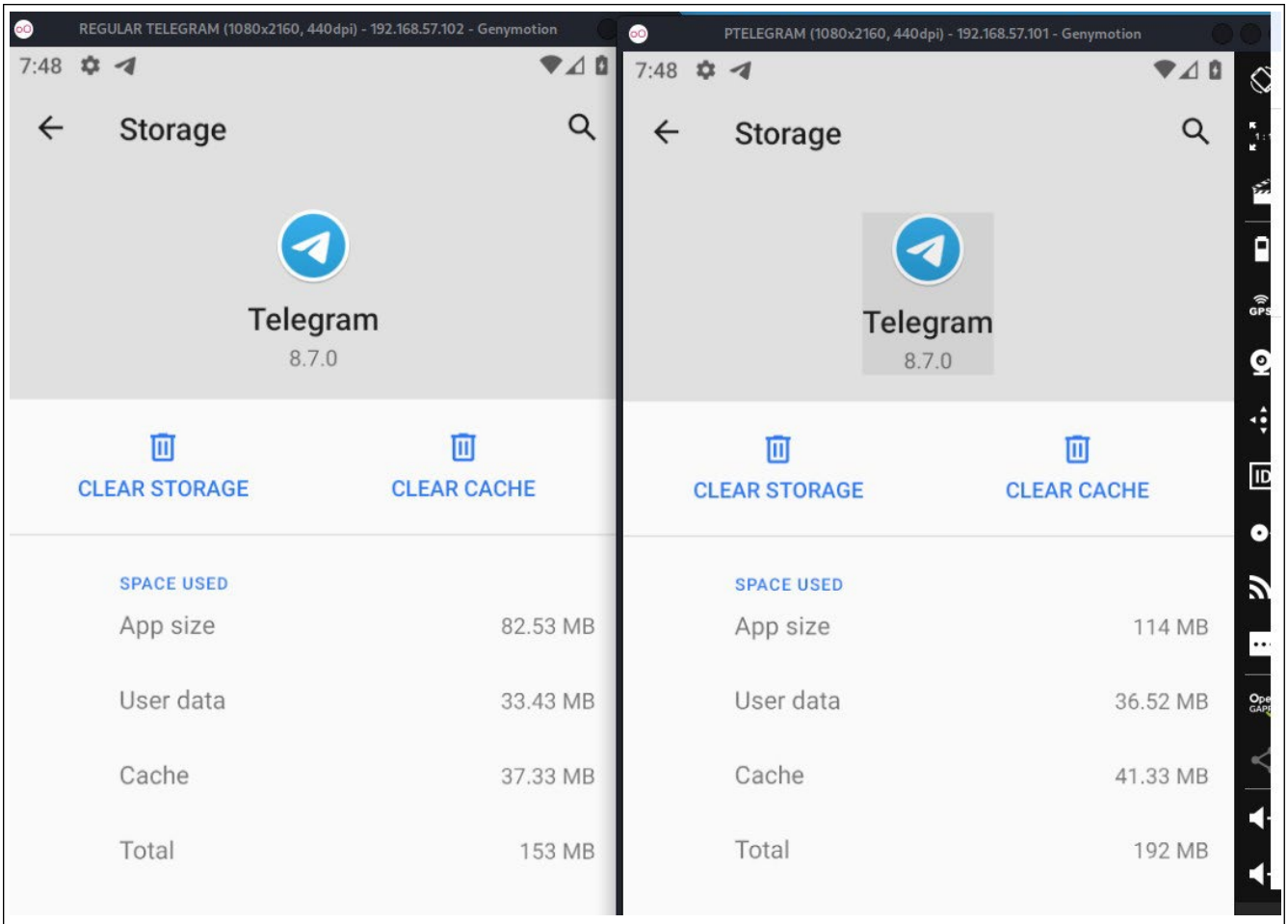
- PTelegram-Info-1: Unknown App Installation Setting
- PTelegram-Info-2: Rooted Device Check
- PTelegram-Info-3: USB Debugging
- PTelegram-Info-4: Missing Domain Reference



SMS Permission		
Risk: High	**Severity**: High	**Difficulty**: Low
Finding ID: PTelegram-High-1		
Component(s): Application Manifest, Android UI		
Impact: Casual inspection of Partisan Telegram's requested permissions using the Android interface will indicate that the Telegram client on the user's phone is not the standard version of Telegram.		
Description: The Partisan Telegram Android application requests the SMS permissions. This permission is not requested by the corresponding Telegram application. In the image below, the right column shows the permissions requested by Partisan Telegram.  		
Solution:		

Discontinue the use of SMS as a secondary notification mechanism, negating the need for the SMS permission. Signal is recommended as an alternative to SMS only if the utilization of Signal would not raise suspicion. Alternative secondary notification systems, such as hang-up phone calls, might be used. However, unless out-of-band communications are necessary, the Contractor recommends containing communications to the Telegram platform in order to minimize the potential exposure of those being notified.

App Disk Utilization		
Risk: High	**Severity**: High	**Difficulty**: Low
Finding ID: PTelegram-High-2		
Component(s): Android UI		
Impact: Casual inspection of Partisan Telegram's storage use using the Android interface will indicate that the Telegram client on the user's phone is not the standard version of Telegram due to differences in "App size".		
Description: Partisan Telegram utilizes significantly more disk space for the application after it is installed. The screenshot below indicates Telegram 8.7.0 occupies 82.53 MB of disk space while Partisan Telegram 2.18.6 occupies 114 MB of disk space.		



Solution:

Unfortunately, it is unlikely there is an effective mechanism for completely mitigating this vulnerability. Partisan Telegram will always be larger than Telegram.

One mitigation may be to base Partisan Telegram on some popular Telegram alternative rather than on Telegram itself. Partisan Telegram will always be necessarily larger than any application it attempts to emulate, though, unless such Partisan Telegram Developers remove functionality from the Telegram client being emulated.

The Contractor also recommends attempting to minimize Partisan Telegram's application size by minimizing code and resource files (images, videos, etc.) added. Smaller differences in application size may appear less suspicious to casual observation.

Application Signing Key and Password in GitHub

Risk: High	**Severity**: Critical	**Difficulty**: Medium
Finding ID: PTelegram-Medium-3		

Component(s):: https://github.com/wrwrabbit/Partisan-Telegram-Android/blob/master/gradle.properties

https://github.com/wrwrabbit/Partisan-Telegram-Android/blob/master/gradle.properties

https://github.com/wrwrabbit/Partisan-Telegram-Android/blob/master/TMessagesProj/config/release.keystore

https://github.com/vivabelarus/Telegram/blob/master/TMessagesProj/config/release.keystore

Impact:

Anyone could generate an APK signed with the same key as official releases. A malicious actor could generate a signed malicious APK and social engineer Partisan Telegram users into installing that malicious app. While social engineering users to download a malicious app can often be challenging, Partisan Telegram users that do not compile the app themselves are *expecting* to download and install app updates, simplifying the process.

Description:

The code signing key likely used to sign official releases and the passwords which protect it were found in GitHub. Android uses these code signing keys as a mechanism to authenticate updates. Any APK signed by the same key and representing itself as Partisan Telegram could be installed on a device and replace the official release. Although there are instructions provided for replacing this key and the associated passwords if individuals want to compile their own version of Partisan Telegram, it is likely that many users make use of the official APKs provided via Telegram and GitHub.

Partisan Telegram gradle.properties:

```
16  RELEASE_KEY_PASSWORD=UCKJJtMyqB!9uGrAw6xu
17  RELEASE_KEY_ALIAS=key0
18  RELEASE_STORE_PASSWORD=LdAaKx_MFWGzL4ix4Jj*
```

Telegram gradle.properties:

```
15  RELEASE_KEY_PASSWORD=android
16  RELEASE_KEY_ALIAS=androidkey
17  RELEASE_STORE_PASSWORD=android
```

Signing Certificate in Official Release:

```
APK is signed
v1 signature: True
v2 signature: True
v3 signature: False
Found 1 unique certificates
Subject: C=BY, ST=CP, L=CP, O=CP, OU=CP, CN=C P
Signature Algorithm: rsassa_pkcs1v15
Valid From: 2021-01-16 12:27:52+00:00
Valid To: 2046-01-10 12:27:52+00:00
Issuer: C=BY, ST=CP, L=CP, O=CP, OU=CP, CN=C P
Serial Number: 0x467a14a5
Hash Algorithm: sha256
md5: 196f25fb9418ad0ab9ead6073df09ddf
sha1: b134df916190f59f832be4e1de8354dc23444059
sha256: 7a7d4936fad1a022f4db2b24b8c9687b80c79099d986c05c541d002308872421
sha512: 859ea402e00aa949fac575bffd59c7471483484fdb833bef3bb2e9be6c212f20519aa3
PublicKey Algorithm: rsa
Bit Size: 2048
Fingerprint: 6c2e052a8924dc9ca524999e4485698c931125e9392de7b96e3e4b58b8240efb
```

Signing Certificate in GitHub KeyStore:

```
APK is signed
v1 signature: True
v2 signature: True
v3 signature: False
Found 1 unique certificates
Subject: C=BY, ST=CP, L=CP, O=CP, OU=CP, CN=C P
Signature Algorithm: rsassa_pkcs1v15
Valid From: 2021-01-16 12:27:52+00:00
Valid To: 2046-01-10 12:27:52+00:00
Issuer: C=BY, ST=CP, L=CP, O=CP, OU=CP, CN=C P
Serial Number: 0x467a14a5
Hash Algorithm: sha256
md5: 196f25fb9418ad0ab9ead6073df09ddf
sha1: b134df916190f59f832be4e1de8354dc23444059
sha256: 7a7d4936fad1a022f4db2b24b8c9687b80c79099d986c05c541d002308872421
sha512: 859ea402e00aa949fac575bffd59c7471483484fdb833bef3bb2e9be6c212f20519aa3d19
PublicKey Algorithm: rsa
Bit Size: 2048
Fingerprint: 6c2e052a8924dc9ca524999e4485698c931125e9392de7b96e3e4b58b8240efb
```

Solution:

Generate a new application code signing key and change the passwords used to protect it. Ensure that code signing key and keystore do not get pushed to GitHub. Treat this as a code signing key for releases, not development. Generate a new release signed by the new key and strongly encourage all users who have downloaded the application to update to the version. Users may have to manually uninstall the previous version.

Additionally, it is recommended that the signing certificate resemble Telegram's signing certificate as closely as possible. Google does not validate the authenticity of signing certificates; they are merely trust-on-first-use and anyone could generate a signing certificate with identical Issuer information. Be aware that the hashes and fingerprints are unlikely to be identical, though, and that the real Telegram key size is smaller than recommended. Although the hashes will never match perfectly, it may improve resilience against casual inspection.

Telegram Signing Certificate:

```
APK is signed
v1 signature: True
v2 signature: True
v3 signature: False
Found 1 unique certificates
Subject: L=Saint-Petersburg, O=VK, OU=VK, CN=Nikolay Kudashov
Signature Algorithm: rsassa_pkcs1v15
Valid From: 2013-08-29 19:13:13+00:00
Valid To: 2038-08-23 19:13:13+00:00
Issuer: L=Saint-Petersburg, O=VK, OU=VK, CN=Nikolay Kudashov
Serial Number: 0x521f9d49
Hash Algorithm: sha1
md5: 26babc62540ef0c20bfc6bacf3d3b1f5
sha1: 9723e5838612e9c7c08ca2c6573b6026d7a51f8f
sha256: 49c1522548ebacd46ce322b6fd47f6092bb745d0f88082145caf35e14dcc38e1
sha512: b47593b965b36396b06296546de386e452f84180968bca894ac90c5dfdbdedae97c3701
PublicKey Algorithm: rsa
Bit Size: 1024
Fingerprint: 5a75db88b7dde612c71b129469a376894dee2c4d8c3266380b658d837ee46677
```

Bundled Native Libraries		
Risk: Medium	**Severity**: High	**Difficulty**: Medium
Finding ID: PTelegram-Medium-1		
Component(s): Compiled APK		
Impact: Inspecting the bundled and shared libraries would indicate that the Telegram client installed on the device is not standard Telegram. Although finding this difference with Drozer would likely require some technical sophistication on the part of opposition forces, it would be trivial to develop a tool that could be used by unsophisticated opposition forces to find this difference.		
Description:		

Partisan Telegram's official release includes versions of the libraries liblanguage_id_jni.so and libtmessages.40.so for all possible release platforms. Additionally, Partisan Telegram uses different shared native libraries than are used in the corresponding version of Telegram.

Partisan Telegram Bundled and Shared Native Libraries:

```
drozer Console (v2.4.4)
dz> run scanner.misc.native -a org.telegram.messenger
Package: org.telegram.messenger
  Bundled Native Libraries:
   - lib/arm64-v8a/liblanguage_id_jni.so
   - lib/arm64-v8a/libtmessages.41.so
   - lib/armeabi-v7a/liblanguage_id_jni.so
   - lib/armeabi-v7a/libtmessages.41.so
   - lib/x86/liblanguage_id_jni.so
   - lib/x86/libtmessages.41.so
   - lib/x86_64/liblanguage_id_jni.so
   - lib/x86_64/libtmessages.41.so

  Shared Native Libraries:
   - /system/framework/org.apache.http.legacy.jar
```

Telegram Bundled and Shared Native Libraries:

```
drozer Console (v2.4.4)
dz> run scanner.misc.native -a org.telegram.messenger
Package: org.telegram.messenger
  Bundled Native Libraries:
   - lib/armeabi-v7a/liblanguage_id_jni.so
   - lib/armeabi-v7a/libtmessages.42.so

  Shared Native Libraries:
   - /system/framework/com.google.android.maps.jar
```

Solution:

If possible, ensure Partisan Telegram's shared native libraries are the same as those used within Telegram. Consider releasing multiple versions of Partisan Telegram compiled for different architectures. This would likely reduce the number of bundled native libraries. However, that would require Partisan Telegram users to ensure they are downloading the correct version. Minimizing bundled native libraries may also help reduce storage space required by application.

Application Manifest Differences

Risk: Medium	**Severity**: High	**Difficulty**: Medium
Finding ID: PTelegram-Medium-2		
Component(s): Application Manifest		

Impact:

Inspection of the application's manifest file would indicate that the application is not standard Telegram. This could be performed by extracting the manifest file from the APK left on the device after installation. This would require a USB connection to the device and the analysis could be scripted for ease of use by unsophisticated opposition forces. This could also be done programmatically from a malicious application using the Android Package Manager.

Description:

Partisan Telegram's application manifest has three differences from Telegram's application manifest. First, the manifest contains the app's request for an SMS permission previously documented above. Second, the Google Maps API key differs. The third difference was observed in the <manifest> tag of the application manifest, in which there were differing attributes between Partisan Telegram and Telegram.

API Key in Partisan Telegram:

```
<meta-data android:name="com.google.android.maps.v2.API_KEY"
android:value="AIzaSyA-t0jLPjUt2FxrA8VPK2EiYHcYcboIR6k"/>
```

API Key in Telegram:

```
<meta-data android:name="com.google.android.maps.v2.API_KEY"
android:value="AIzaSyDdP0nhiLgxWOJc9ddIOx5Kf3SNE18np38"/>
```

Partisan Telegram <manifest>:

```
<manifest android:versionCode="26229" android:versionName="8.7.0"
android:installLocation="auto" android:compileSdkVersion="31"
android:compileSdkVersionCodename="12"
package="org.telegram.messenger" platformBuildVersionCode="31"
platformBuildVersionName="12"
 xmlns:android="http://schemas.android.com/apk/res/android">
```

Telegram <manifest>:

```
<manifest android:versionCode="26221" android:versionName="8.7.0"
android:installLocation="auto" android:compileSdkVersion="31"
android:compileSdkVersionCodename="12"
package="org.telegram.messenger" platformBuildVersionCode="31"
platformBuildVersionName="12"
 xmlns:android="http://schemas.android.com/apk/res/android">
```

Solution:

Remove the SMS permission. This was also recommended as a solution to PTelegram-High-1. Additionally, extract the Google Maps API key from the version Telegram which is being emulated and reuse it in Partisan Telegram. This should likely be checked for each Telegram release to determine if the API key has been rotated. Be aware, however, that this would likely constitute a Google Maps Terms of Service violation. Also, ensure that all attributes of the application manifest are the same between Telegram and Partisan Telegram.

Unknown App Installation Setting

Risk: Info	**Severity**: Info	**Difficulty**: N/A
Finding ID: PTelegram-Info-1		
Component(s): Android UI		

Impact:

A casual observer would be able to notice that users of Partisan Telegram have installed software from outside of standard distribution channels, such as Google Play. Partisan Telegram developers have indicated to the Contractor that the installation of apps from

unknown sources are not unusual in regions where they expect Partisan Telegram to be used, so this may present less risk.
Description: It is likely that most users of Partisan Telegram are going to download the official release APKs using a web browser or Partisan Telegram on their mobile device. Users must grant special permission to install APKs from unknown sources. This usually refers to any source other than Google Play. In older versions of Android this is a system-wide permission and, on newer versions of Android, this permission can be granted to the users on an app-by-app basis. Users of the application may fail to remove this permission once granted.
Solution: Include a software check in Partisan Telegram that, when opened, determines if apps may be installed from unknown sources. This can be done programmatically by checking Secure.Settings.INSTALL_NON_MARKET_APPS on API versions prior to 17 and by calling PackageManager.CanRequestPackageInstalls() on subsequent API versions.

Rooted Devices		
Risk: Info	**Severity**: Info	**Difficulty**: N/A
Finding ID: PTelegram-Info-2		
Targets: Android Application Settings		
Impact: Users of Partisan Telegram may be identified and sensitive information contained in the file system may be disclosed.		
Description: It is trivial to identify users of Partisan Telegram if the device on which it is running is rooted, as there are many differences in the artifacts that Partisan Telegram leaves on the file system. On a rooted device, it is likely that any application could access these files, which may contain sensitive information.		
Solution: Include a warning in the Q&A that Partisan Telegram is not secure on a rooted device and consider implementing programmatic checks that provide the user an appropriate warning		

message if the device is rooted. Unfortunately, there is no single mechanism for identifying if a device is rooted.

USB Debugging		
Risk: Info	**Severity**: Info	**Difficulty**: N/A
Finding ID: PTelegram-Info-3		
Component(s): Android Application Settings		
Impact: Users of Partisan Telegram may be identified, and sensitive information contained in the file system may be disclosed.		
Description: USB Debugging is a setting within a hidden “Developer Options” menu in Android. If this setting is enabled, it allows remote management of the device over the Android Debug Bridge (ADB). Use of the Android Debug Bridge requires that a user of the phone “trust” the computer on which ADB is running through an interface prompt. Once the mobile device trusts the system running ADB, sensitive files - such as APK files - can be pulled from the Android device. This finding was considered within scope because it exclusively relies on settings native to Android without any specialized forensic tools. It is the opinion of the Contractor that this process could be automated in such a way that would allow unsophisticated adversarial forces to carry it out, though it would likely require either forcing or social engineering users into trusting another device. It may also be able to deploy this sort of data collection at scale with malicious public USB charging stations. ADB can operate over a physical USB connection or over TCP/IP, though the latter requires extra configuration on the device at the command line.		
Solution: Consider including a warning in the Q&A that Partisan Telegram is not secure on a device with USB debugging enabled and consider implementing programmatic checks that provide the user an appropriate warning message if this setting is turned on. There is, however, some risk in adding this to the Q&A. USB debugging is a feature native to Android and documenting this attack path would clearly indicate how Partisan Telegram use could be		

detected. The Contractor leaves it to the Partisan developers to determine which option presents less risk to Partisan Telegram users.

This setting may be programmatically checked by inspecting Settings.Secure.ADB_ENABLED on API versions before version 17 and Settings.Global.ADB_ENABLED on subsequent API versions.

Missing Domain Reference		
Risk: Info	**Severity**: Info	**Difficulty**: N/A
Finding ID: PTelegram-Info-4		
Targets: Android Application Settings		
Impact: Mobile device traffic that interacts with Telegram but does not make DNS calls to the domain tmessages2.firebaseio.com may be identifiable as traffic generated by Partisan Telegram.		
Description: Static code analysis identified one domain, tmessages2.firebaseio.com, that is used by Telegram and not Partisan Telegram. It may be that this domain was recently added to the Telegram code base or is in code that Partisan Telegram developers explicitly modified. The Contractor did not observe any DNS calls to this domain in network traffic captures made during this assessment.		
Solution: Determine if use of this domain was explicitly removed from Partisan Telegram. If not, consider reconciling the Partisan Telegram codebase with the Telegram code base to ensure minimal differences in network traffic generated by the applications.		

Appendix A – Vulnerability Definitions

The following sections describe the classes, severities, and exploitation difficulty rating assigned to each identified issue by the security assessment team.

Severities

- **Informational**: The issue does not pose an immediate risk, but is relevant to security best practices or Defense-in-depth.
- **Undetermined**: The extent of the risk was not determined during this engagement.
- **Low**: The risk is relatively small, or is not a risk the customer has indicated is important.
- **Medium**: Individual user information is at risk, exploitation could lead to moderate financial impact or reputational damage.
- **High**: Large numbers of users are affected, exploitation will lead to significant financial impact, reputational damage, or pose serious legal implications by indicating that a non-standard version of Telegram is in use.
- **Critical**: This vulnerability poses similar risks as High severity, however the issue is either actively being exploited in the wild, or is otherwise highly likely to be discovered and used. This likely would indicate that Partisan Telegram is in use and/or disclose should be immediately addressed.

Difficulties

- **N/A**: This finding does not contain a difficulty rating.
- **Undetermined**: The difficulty of the exploit was not determined during this engagement.
- **Low**: Commonly exploited, public tools exist or can be scripted that exploit this flaw. Could be carried out by an individual with little technical knowledge via casual observation.
- **Medium**: Attackers must write an exploit, or need an in-depth knowledge of a complex system. Could be carried out by an individual with little technical knowledge using custom tools.
- **High**: The attacker must have privileged insider access to the system, may need to know extremely complex technical details or must discover other weaknesses in order to exploit this issue.

Appendix B – Project Team

The Contractor's Security Assessment Team consisted of the following primary members:

- Justin Pelletier – Business Director
- Robert Olson – Lead Assessor
- Jason Ross – Assessor
- Forrest Fuqua – Assessor
- Sarthak Mathur - Student Assessor
- Sergei Chuprov - Student Assessor / Translator

This project was facilitated and managed in partnership with the following Open Tech Fund representatives on behalf of the Client:

- Sarah Aoun – Vice President of Security
- Shems Abdelwahab – Project Manager/Liaison

Appendix C – Supplemental Data to be Provided

File Name	Description
MobSF Partisan Telegram 2-18-6 Report.pdf	Static source code analysis report from MobSF for Partisan Telegram version 2.18.6.
MobSF Telegram 8-7-0 Report.pdf	Static source code analysis report form MobSF for Telegram version 8.7.0. This was used as a security baseline for analyzing MobSF Report 2-18-6.pdf
Partisan-Telegram-dependency-check-report.html	This contains the OWASP Dependency Check results for Partisan Telegram version 2.18.6 and documents any known vulnerabilities in detected software dependencies.
Telegram-dependency-check-report.html	This contains the OWASP Dependency Check results for Telegram 8.7.0 and documents any known vulnerabilities in detected software dependencies.
day1-telegram.pcapng	Packet capture of all traffic from Genymotion virtual device running Telegram from 7/15/22 to 7/16/22.
day1-ptelegram.pcapng	Packet capture of all traffic from Genymotion virtual device running Partisan Telegram from 7/15/22 to 7/16/22.
day23-telegram.pcapng	Packet capture of all traffic from Genymotion virtual device running Telegram from 7/16/22 to 7/18/22.
day23-ptelegram.pcapng	Packet capture of all traffic from Genymotion virtual device running Partisan Telegram from 7/16/22 to 7/18/22.
day4-telegram.pcapng	Packet capture of all traffic from Genymotion virtual device running Telegram from 7/18/22 to 7/19/22.
day4-ptelegram.pcapng	Packet capture of all traffic from Genymotion virtual device running Partisan Telegram from 7/18/22 to 7/19/22.