

Cerberus Analysis

written by Mert SARICA | 1 December 2020

In February 2020, I received a SMS on my cell phone that made me quite suspicious. When I visited the [https://ko\[.\]tc/hediyekazani](https://ko[.]tc/hediyekazani) web address mentioned in the message, I found that I was redirected to the [http://www-bedavainternethediyeuygulama\[.\]com](http://www-bedavainternethediyeuygulama[.]com) web address. A short time after receiving the SMS, when I visited the website again, I saw that the images on the site had changed. Saying “suspicion is the whip of a cyber security researcher,” I decided to take a closer look at this situation.

18:07

VoLTE LTE 70%



+90 212 985 05 78



14:41

SN; [REDACTED] SARICA Tüm operatorlerde
gecerli 1000 DK, 5 GB INTERNET 3 Ay
bedava! Hemen uygulamayı indir, kur ve
kazan; <https://ko.tc/hediyekazani>
B187

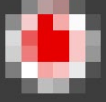
|





thediyeliuygulama.com

16



5G Uygulaması 5GB İnternet Veriyor



1)5G Uygulaması İnternet
Kazandırıyor 5GB İnternet
Kazanmak İçin Hemen
İndir

İndir



Bu türden dosyalar cihazınıza zarar
verebilir. Yine de 5GBeta.apk adlı
dosyayı saklamak istiyor musunuz?



İptal

Tamam





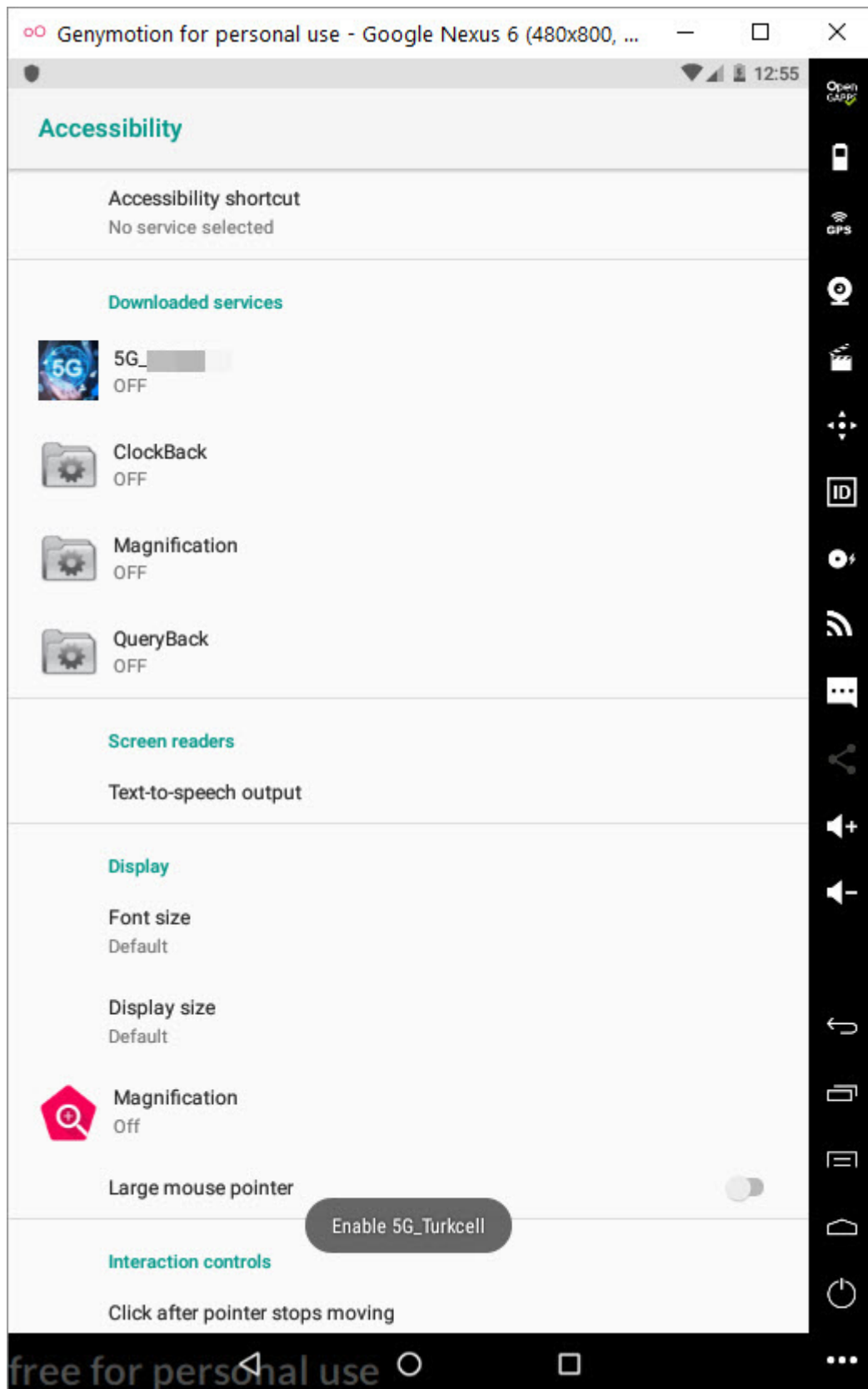
5G Beta Test

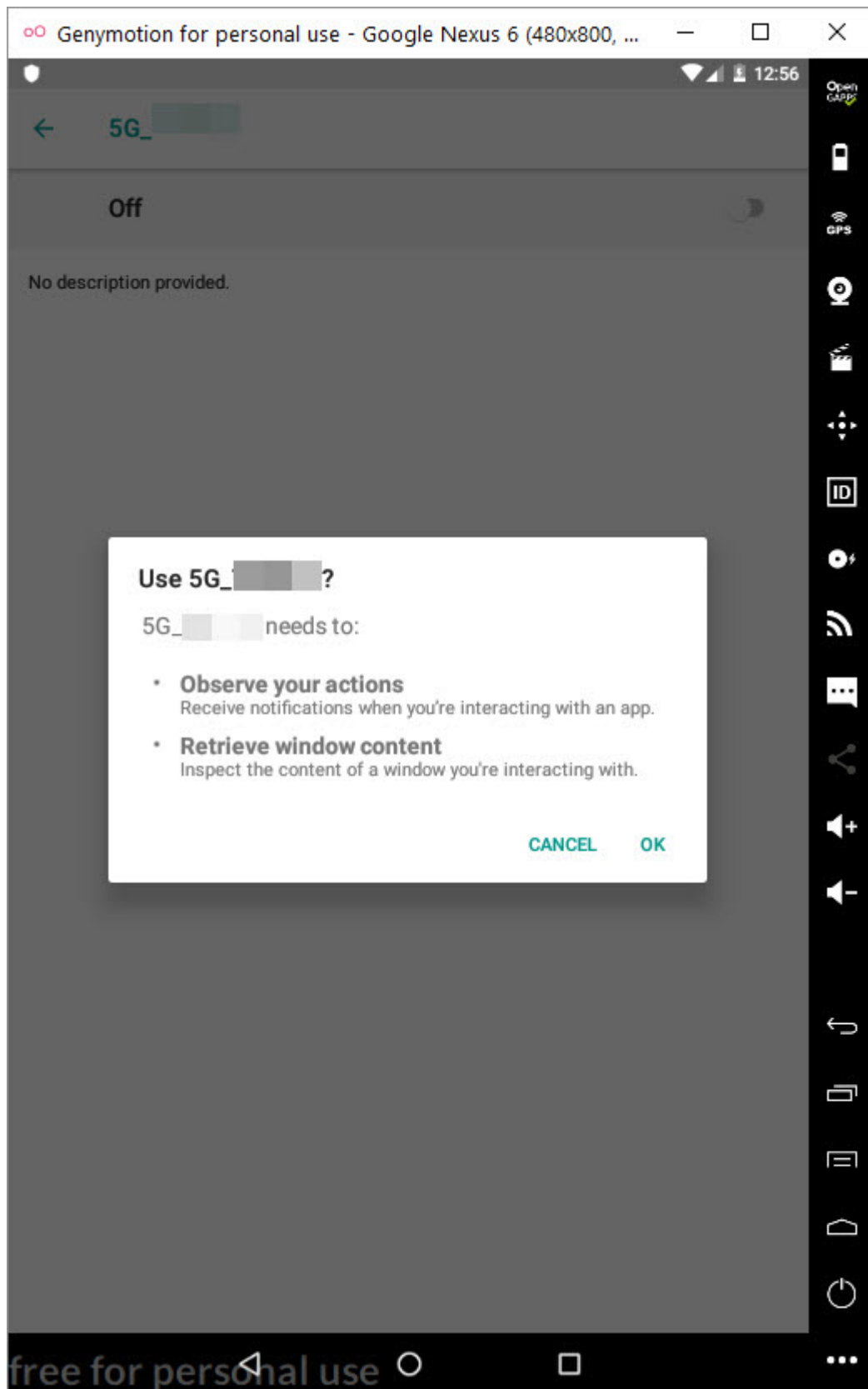
5G Uygulamasini İndirip
Yükleyen Tüm Operatör
Müşterilerine 5GB İnternet
Hediye Sizde Hemen İndirin
Yükleyin 5GB İnterneti
Hemen Kazanın

Uygulamayı
Yükle

I downloaded the 5GBeta.apk file from the website and uploaded it to the Koodous web application, which is used for mobile malware analysis. The analysis failed. Then I uploaded this application to the VirusTotal web application and, although I encountered a clue that it was a banking malware (Cerberus), I did not see the address of the command and control center in the behavioral analysis output. I couldn't find answers to the questions that came to my mind, so I decided to quickly analyze the 5GBeta.apk application dynamically using the Genymotion Android emulator.

As soon as the malicious application was installed on Android, it began to request permissions one by one to achieve its bad intentions. After obtaining the permissions and successfully completing the installation, it hid its icon and started working in the background and communicating with the command and control center with the kryll[.]ug (8[.]208.19.185) web address. When I searched the 8[.]208.19.185 IP address on VirusTotal, it was clear from the passive DNS information that it was not innocent at all.





Accessibility

Accessibility shortcut

No service selected

Downloaded services



5G_
OFF



ClockBack
OFF



Magnification
OFF



QueryBack
OFF

Screen readers

Text-to-speech output

Display

Font size

Default

Display size

Default



Magnification
Off

Large mouse pointer

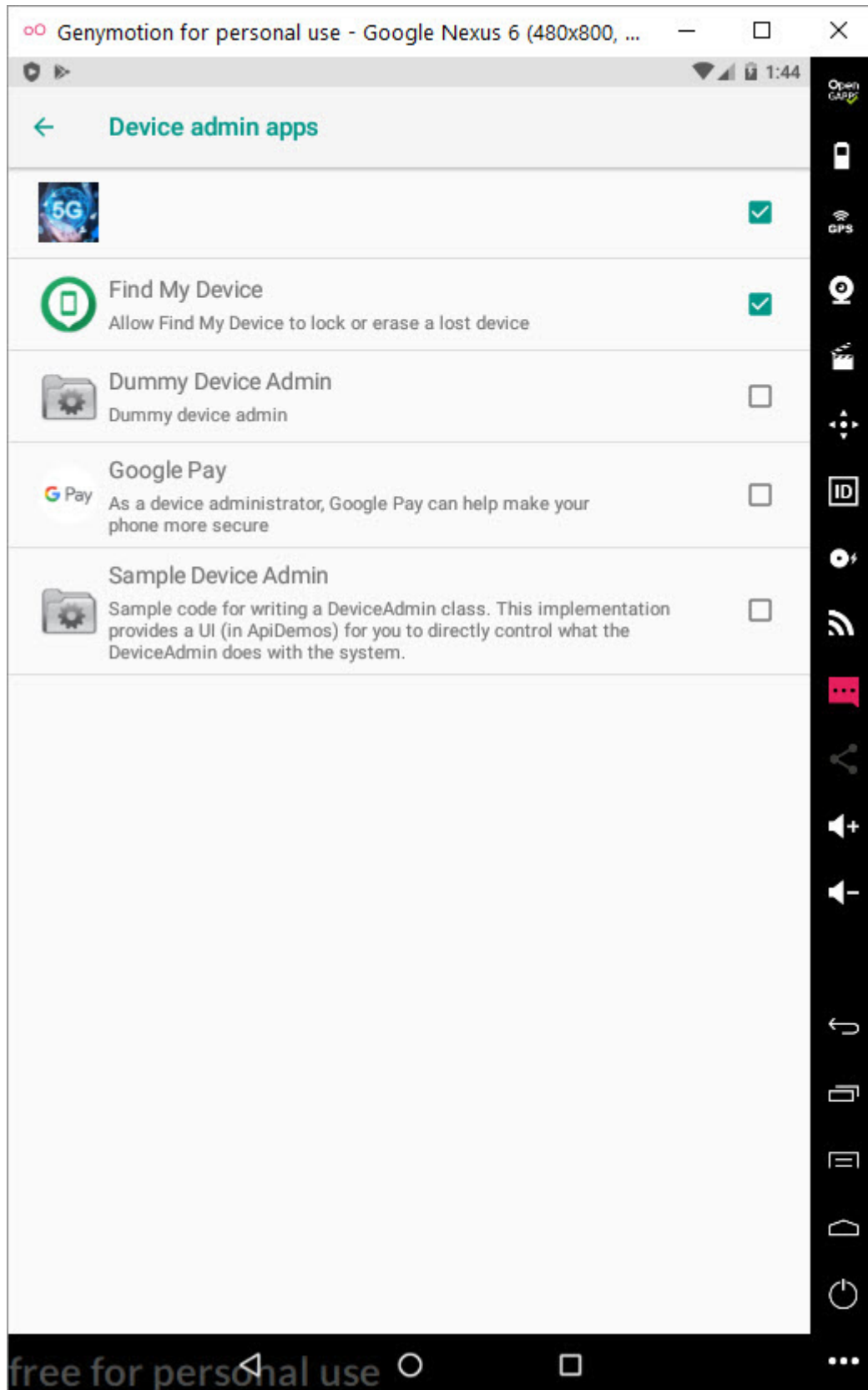


Enable 5G_

Interaction controls

Click after pointer stops moving





In my virtual Android operating system, a screen of Google Play, which was created to steal my credit card information, appeared as there was no banking application installed. When I entered a 16-digit credit card number created for testing, I saw that the save button (SAVE) was not activated. When I made the credit card field 19, the SAVE button became active. Probably the malicious person has made the control of the 3-digit CVV2 number specific to

the form where the credit card number is entered and such an error has occurred. After seeing that all the information I entered went to the command and control center in encrypted form, I decided to pursue the encryption key.

Genymotion for personal use - Google Nexus 6 (480x800, ...)

12:59

Google Play

Cardholder name


Street address

City

State

ZIP code

Telephone number

G Pay 

Continue

By continuing, you agree to the Google Payments [Privacy Notice](#) & [Terms of Service](#).

free for personal use



Osman|

Turkey

Istanbul

Kadikoy

34878

+902163534466



Continue


By continuing, you agree to the Google Payments [Privacy Notice](#) & [Terms of Service](#).




VISA Credit Card Generator, 100% x

creditcardgenerator.com/visa-credit-card-generator/

Other bookmarks



Search





 [Test Generator / Validator](#)

Who do not carry at least one credit card (CC) in their wallet nowadays? Few. Especially with the boom of online shopping, a credit card is a must buying anything online. One of the most popular CC brands is from a company called VISA. Almost every bank and small-mom shop out there try to issue a VISA card to their customers; whether it's a credit, debit, prepaid, or just a charge card.

Valid VISA Credit Card Generator that Work

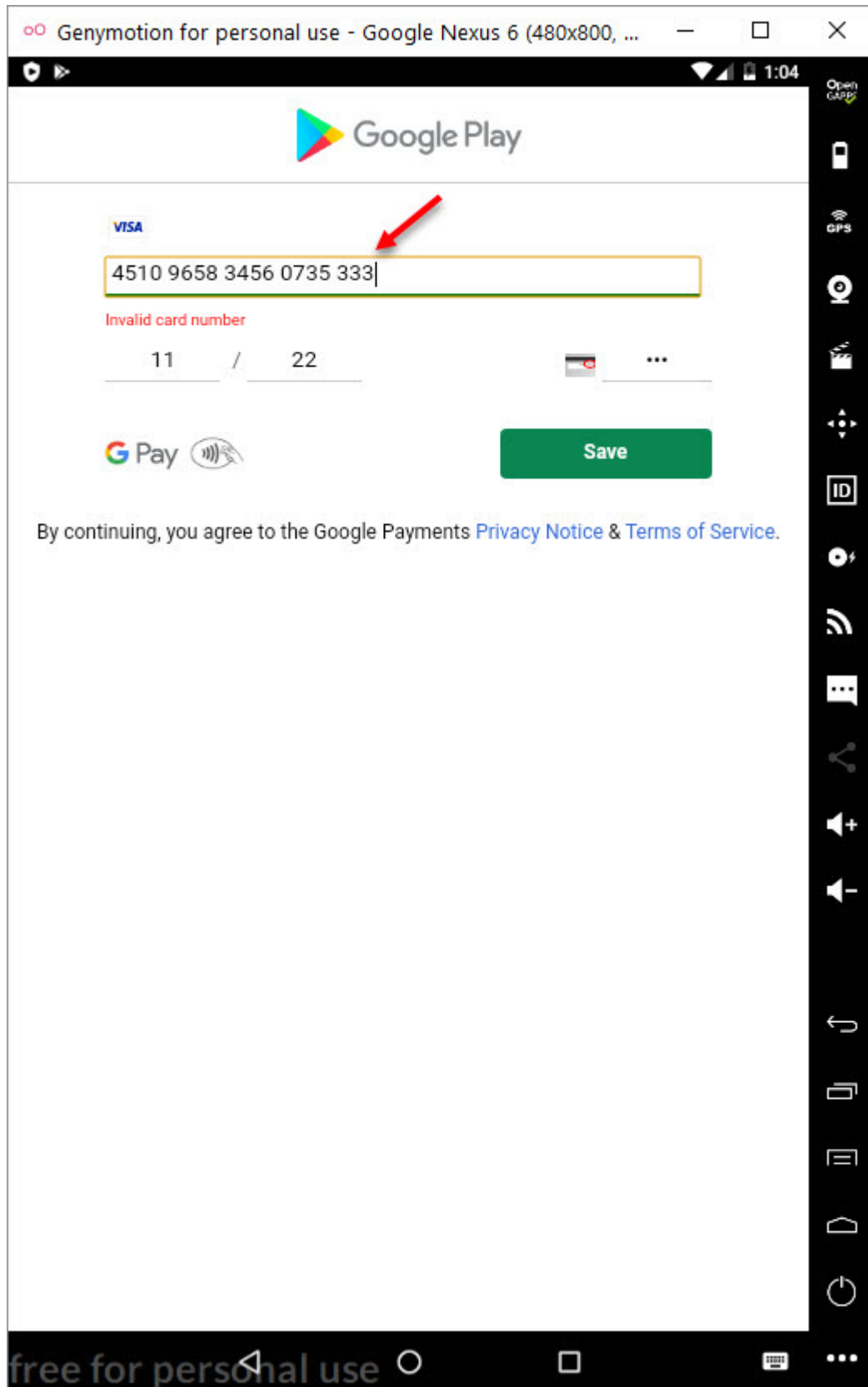
Generator:	Test VISA Credit Cards
Issuing network:	Visa
Card number:	4510 9658 3456 0735
Pin:	1537
Name:	Josh Lunar
Address:	9007 Mountaintrail Way
Country:	France
CVV:	938
Expiration date:	11 / 2022

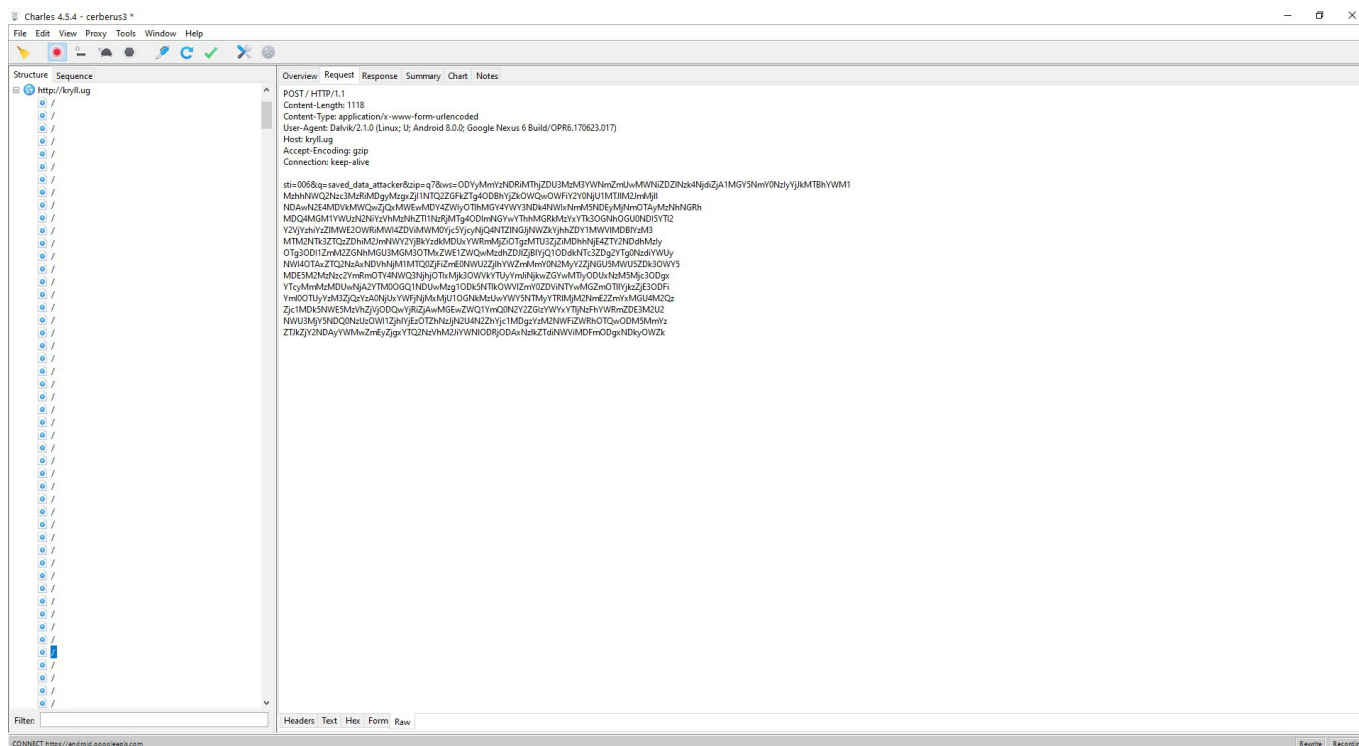
Generate VISA Credit Card



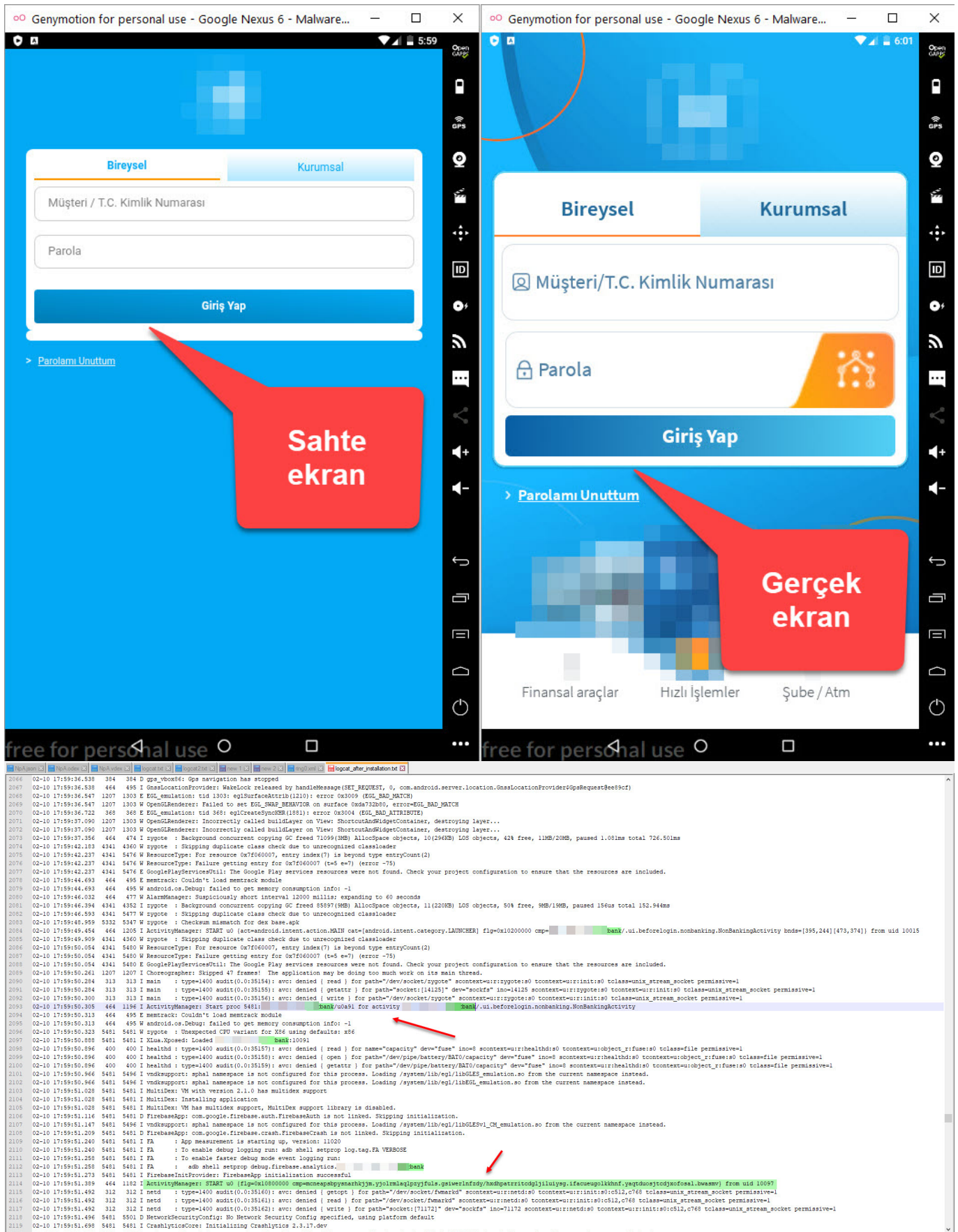
Take a look at the legendary [VISA company](#) if you don't know who they are duh.

People hesitate to share their VISA CC details for an online purchase. Those working as developers or quality assurance engineers for software companies may need to have thousands of credit card numbers to feed through their applications. They need a tool to generate these kinds of valid VISA card numbers in bulk. They should use a **VISA Credit Card Generator 2020** for getting these test numbers regularly.



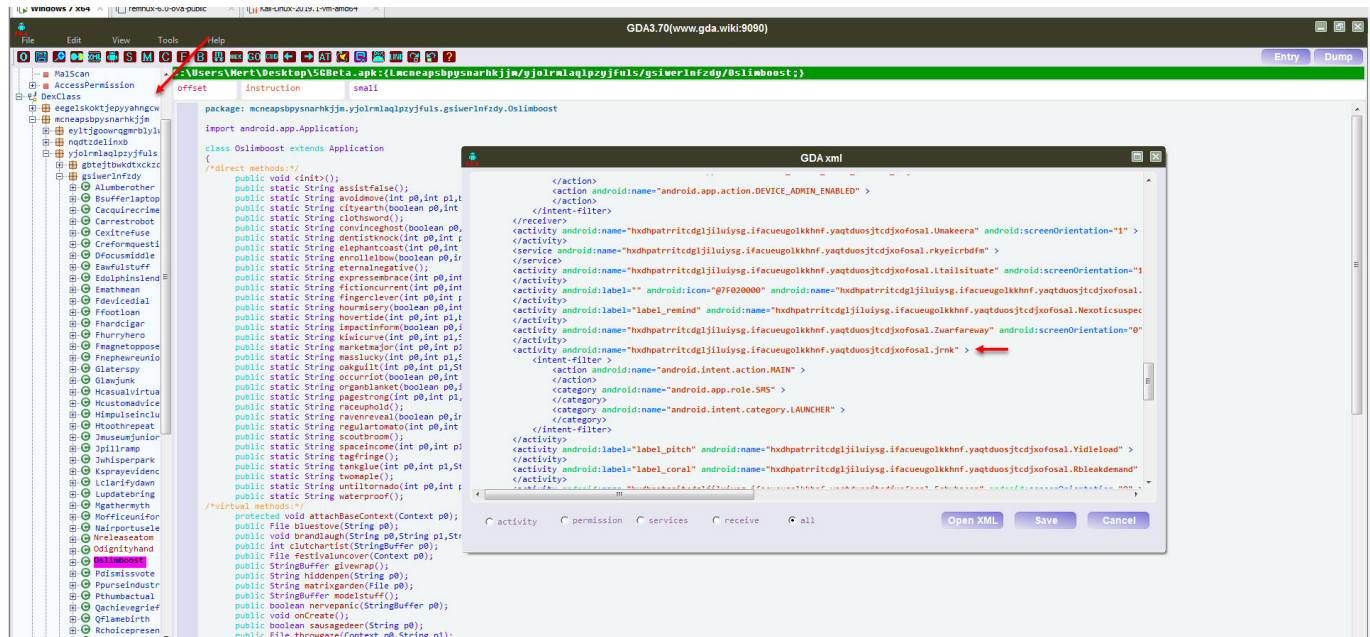


Before pursuing the encryption key, I installed and ran 10 mobile apps of different banks on my virtual system to confirm if the malicious application had stolen any banking information. As a result of my tests, the malicious application was able to steal all the information entered by the user by opening a fake screen over the login screen of the targeted mobile banking application when it was running.

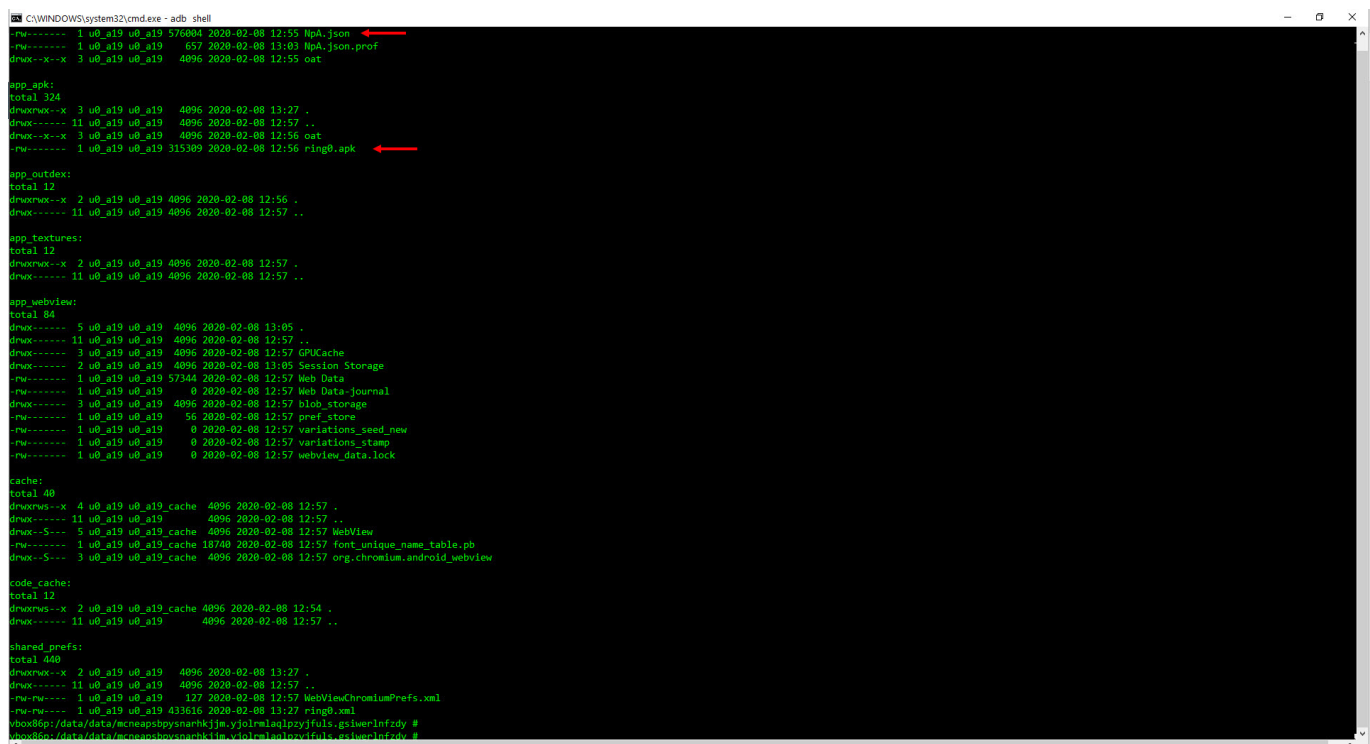


In order to find the encryption key, I used the GDA tool to decompile the 5GBeta.apk application, and by looking at the AndroidManifest.xml file which contains the basic information about the application, I saw that the classes between the MainActivity class, which will run first when the malicious

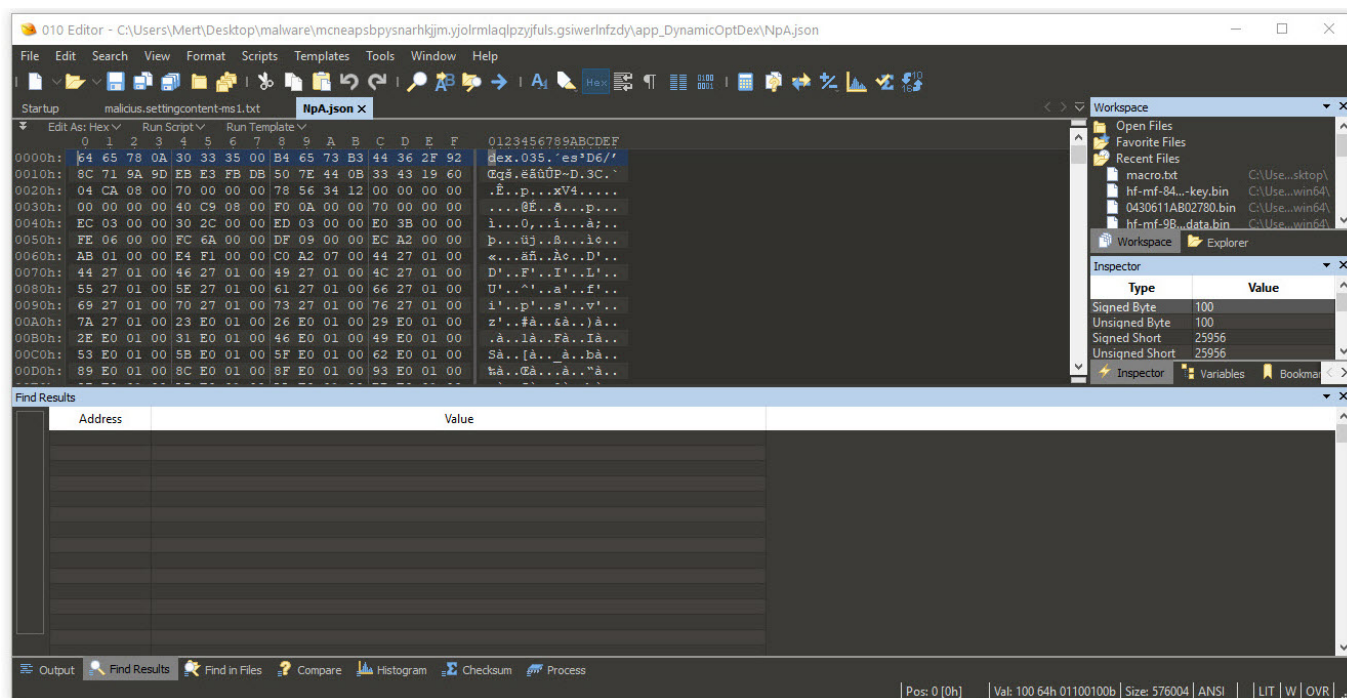
application starts, and the source code were different. This indicated that the malicious code block was loaded dynamically during runtime.

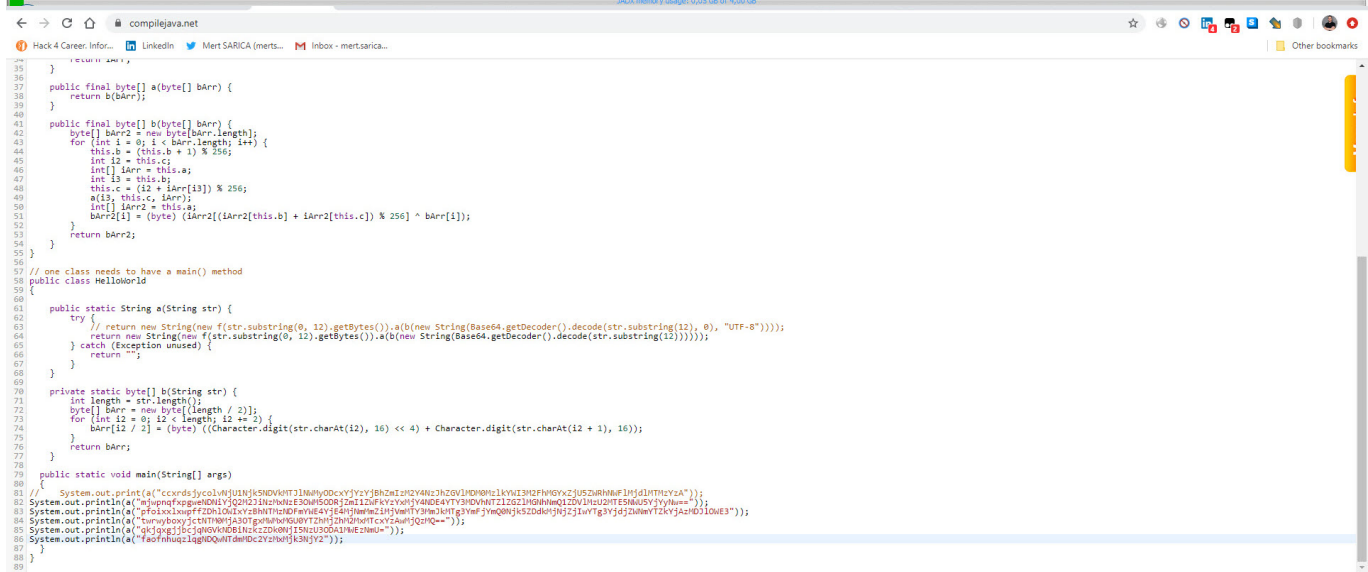
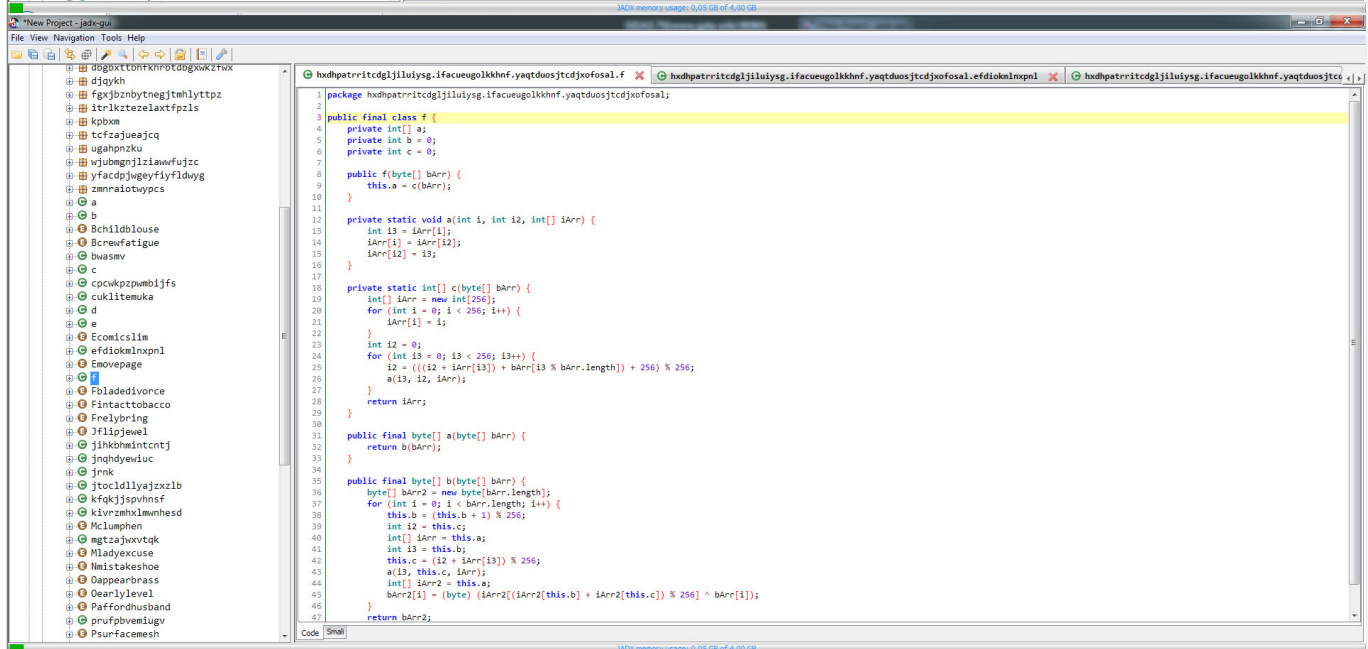
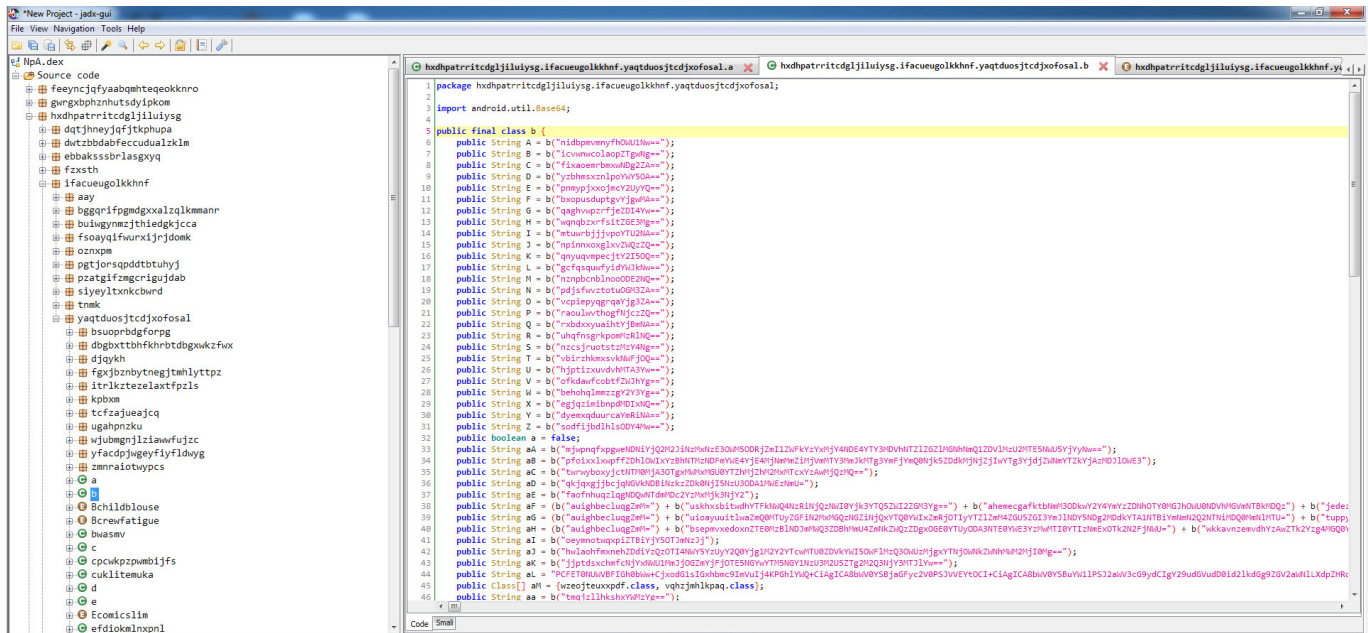


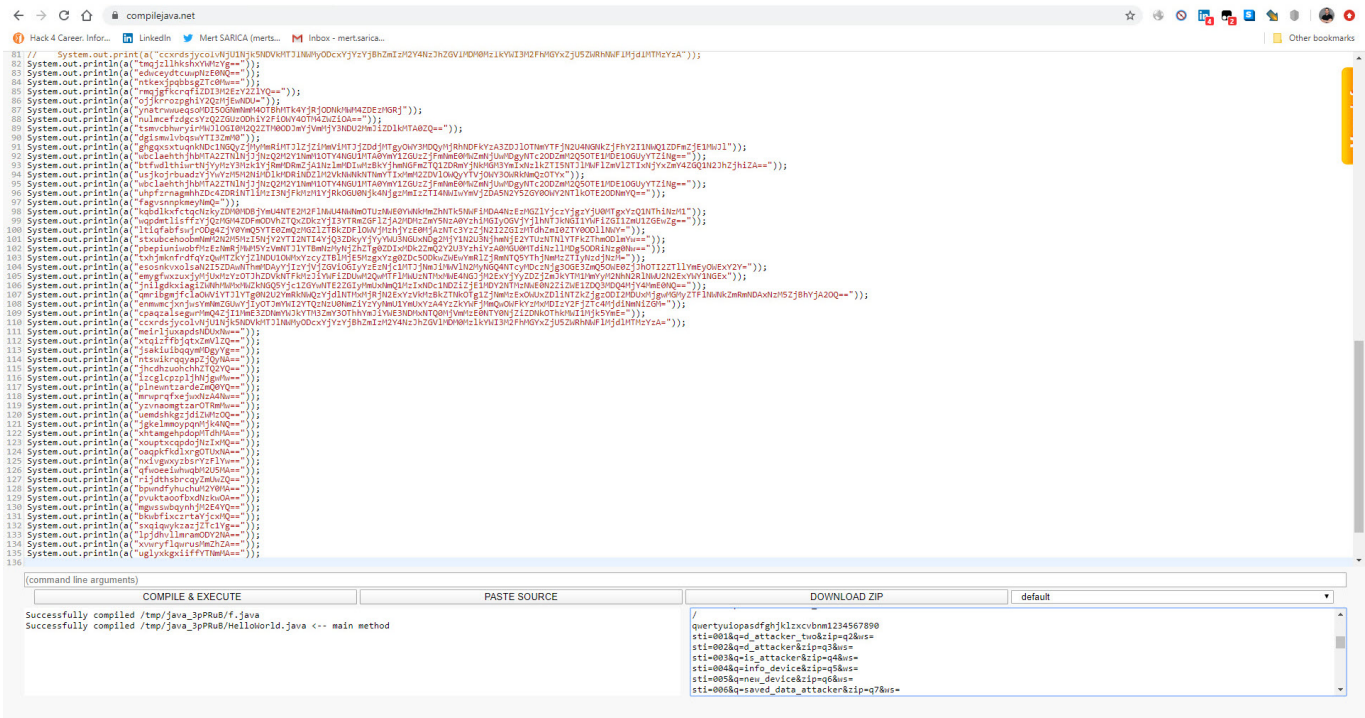
When I looked at the folder `/data/data/mcneapsbypsnnarkhjjm.yjolrmlaqlpzyjfuls.gsiwerlnfzdy` where the malicious application was installed, I noticed the large files `ring0.apk` and `NpA.json`. I learned that `NpA.json` was actually a DEX file, and when I decompiled it using the `jadx` tool to see the source code, I encountered the `MainActivity` class that was present in the `AndroidManifest.xml` file.



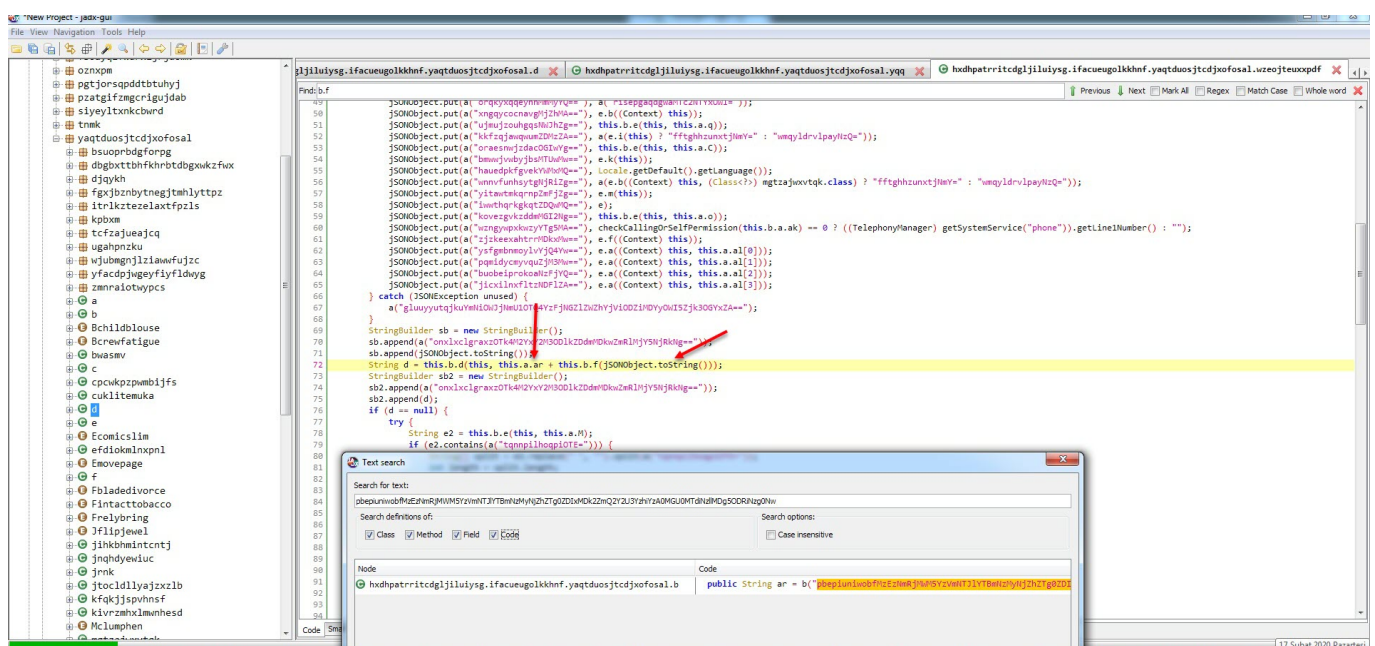
(NDNiYjQ2M2JiNzMxNzE3OWM5ODRjZmI1ZWFKYzYxMjY4NDE4YTY3MDVhNTZlZGZlMGNhNmQ1ZDVlMzU2MTE5NWU5YjYyNw==) are decoded by BASE64 and then decrypted by the encryption key.) I was also able to easily decrypt all the strings by using the Java code in the f class and compiling it with the help of compilejava.net website.

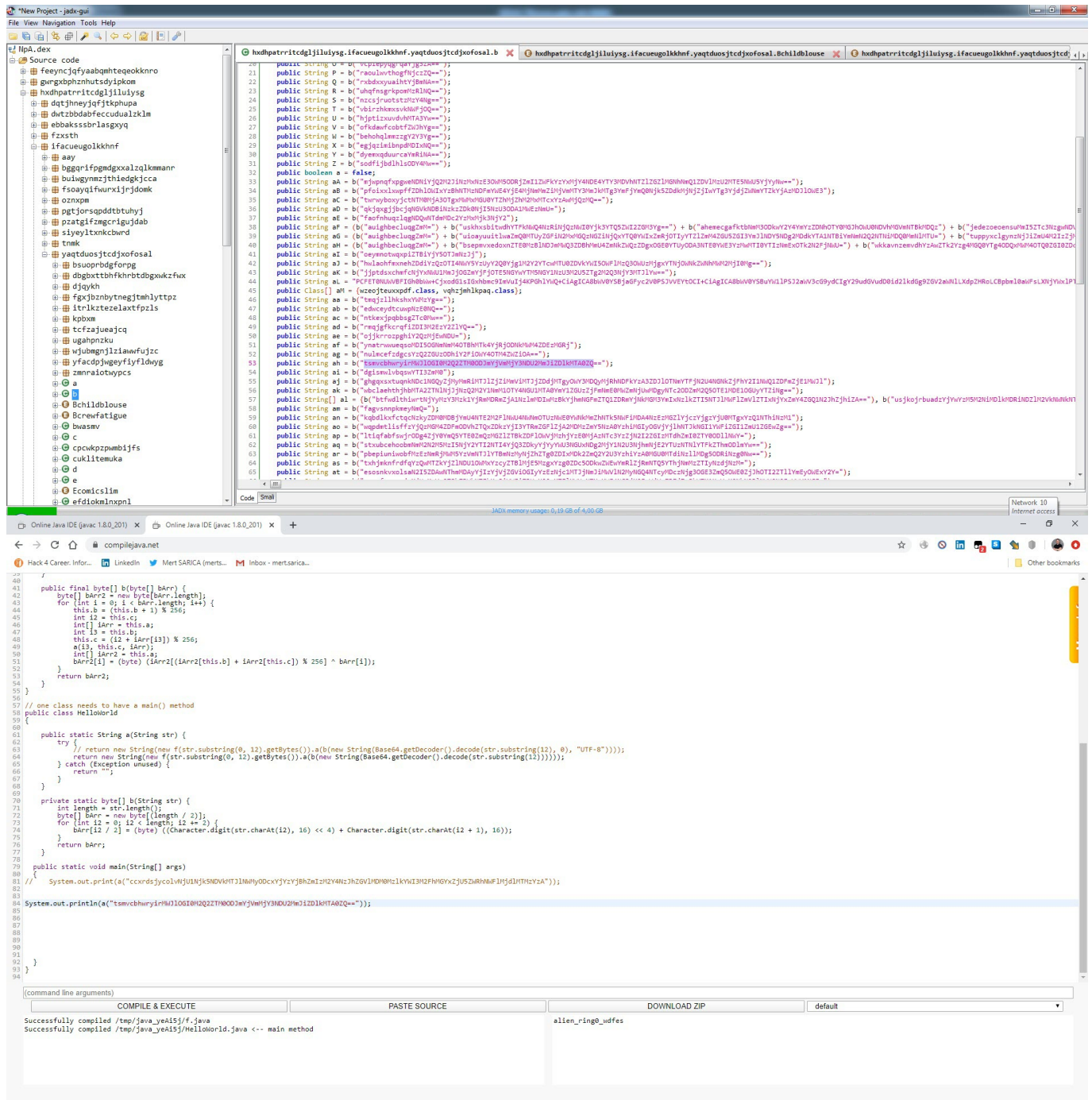




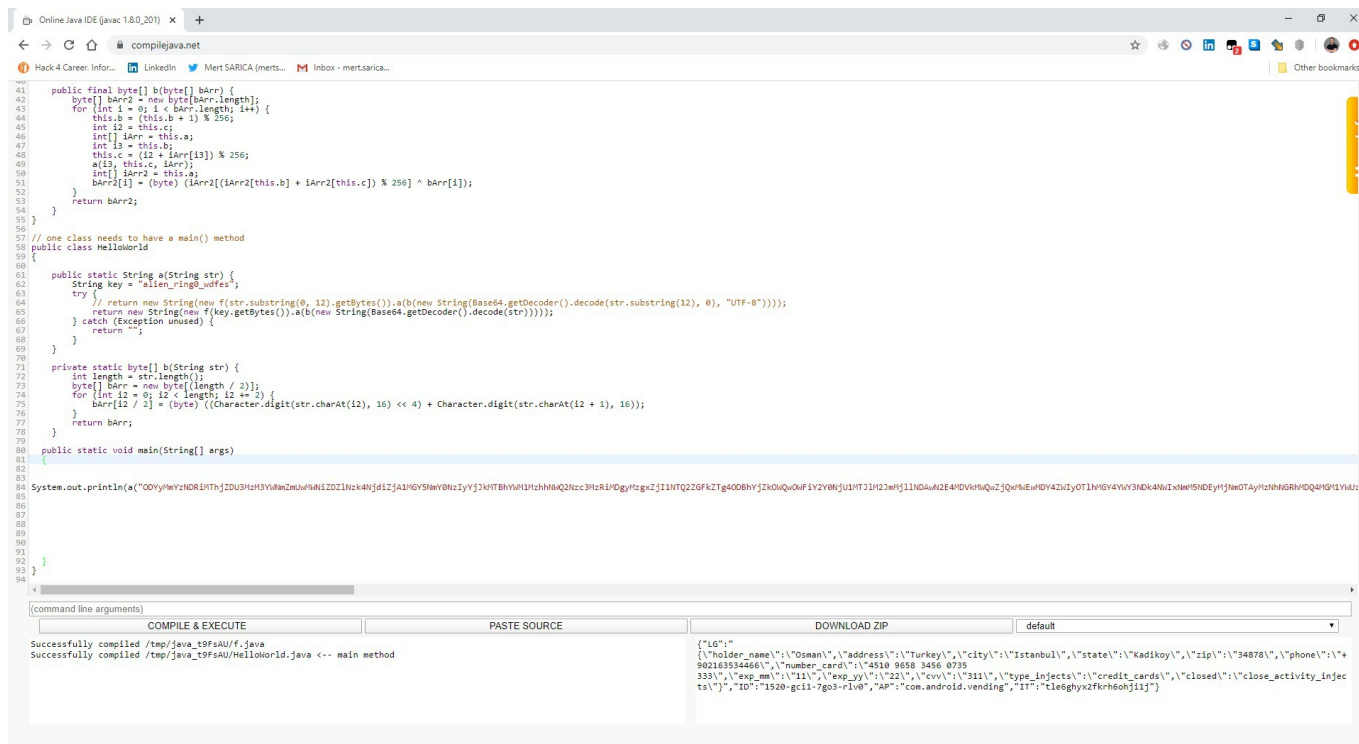


After observing that the encrypted data sent to the command and control center was in JSON format, I focused on the parts of the code where this class was used. Since I knew that the malicious application sent the `sti=004&q=info_device` parameter to the command and control center at certain intervals, and then the encrypted `ws=` parameter, I found the code block where these two values were concatenated. Upon analyzing this code block, I found that the `ws` parameter was encrypted with the `alien_ring0_wdfes` RC4 key.





When it came to decrypting the encrypted data I previously obtained with the alien_ring0_wdfes encryption key, and confirming the validity of the encryption key, I also learned that the fake screens (html) came from the command and control center, as mentioned at the beginning of the text.”



```
41 public final byte[] b(byte[] barr) {
42     byte[] barr2 = new byte[barr.length];
43     for (int i = 0; i < barr.length; i++) {
44         this.b = (this.b + 1) % 256;
45         int i2 = this.c;
46         int[] iarr = this.a;
47         int i3 = this.b;
48         this.c = (i2 + iarr[i3]) % 256;
49         a(i3, this.c, iarr);
50         int[] iarr2 = this.a;
51         barr2[i] = (byte) (iarr2[(iarr2[this.b] + iarr2[this.c]) % 256] ^ barr[i]);
52     }
53     return barr2;
54 }
55 }
56 // one class needs to have a main() method
57 public class HelloWorld
58 {
59     public static String a(String str) {
60         String key = "alien_ing0_w0ffes";
61         try {
62             // return new String(new f(str.substring(0, 12).getBytes()).a(b(new String(Base64.getDecoder().decode(str.substring(12), 0), "UTF-8"))));
63             return new String(new f(key.getBytes()).a(b(new String(Base64.getDecoder().decode(str))));
64         } catch (Exception unused) {
65             return "";
66         }
67     }
68     private static byte[] b(String str) {
69         int length = str.length();
70         byte[] barr = new byte[(length / 2)];
71         for (int i2 = 0; i2 < length; i2 += 2) {
72             barr[i2 / 2] = (byte) ((Character.digit(str.charAt(i2), 16) << 4) + Character.digit(str.charAt(i2 + 1), 16));
73         }
74         return barr;
75     }
76     public static void main(String[] args)
77     {
78         System.out.println(a("00YrWYzNDRIHTJZDU3H2H3YVWwZmWuWnNzDZ1HzK4Njd1ZJA1G0Y5Ww0HzIyY3YHTBHYu1tzhNNAQ2NzC3HzR1PDgyfzgxZj11HTQZ2GFkZTg40DBHYj2K0uQ0uF1Y2Y0NjU1HTj1H23mHj11NDuW4Z4H4DVW9QuZ3Q0rWuW4DY4Z2YzOT1H6Y4YUy3HDK4Nu1xWw5NDEyHjW0TAYtzhNNGRN4Q4H91YU4U;
79     }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
```

Successfully compiled /tmp/java_t9fsAU/f.java
Successfully compiled /tmp/java_t9fsAU/HelloWorld.java <- main method

```
{
  "holder_name": "Osman",
  "address": "Turkey",
  "city": "Istanbul",
  "state": "Kadikoy",
  "zip": "34878",
  "phone": "902163534466",
  "number_card": "4510 9658 3456 0735",
  "exp_mm": "11",
  "exp_yy": "22",
  "cvv": "311",
  "type_injects": "credit_cards",
  "closed": "close_activity_injec",
  "id": "1520-gc11-gn3-r1v8",
  "app": "com.android.vending",
  "t": "11eghyx2fkn6ohj1ijj"
}
```

In conclusion, it was surprising and concerning to learn that the Cerberus mobile banking malware, which has been frequently talked about in recent years for its features and name, has started targeting citizens with SMS containing their name and surname. As always, I stress that Android users should avoid installing apps from unknown sources.

Hope to see you in the following articles.

Note:

1. After this security research (February 2020), it was revealed that (September 2020) this malware, commonly known as Cerberus, later branching off into a malware called Alien from the fork of Cerberus v1 version.
2. This text also includes the solution for the Pi Hediye Var #18 cybersecurity game.