SharkBot: a “new” generation Android banking Trojan being distributed on Google Play Store

By Joost Jansen

Published: 2022-03-03 · Archived: 2026-04-05 22:59:58 UTC

Authors:

- Alberto Segura, Malware analyst
- Rolf Govers, Malware analyst & Forensic IT Expert

NCC Group, as well as many other researchers noticed a rise in Android malware last year, especillay Android banking malware. Within the Treat Intelligence team of NCC Group we’re looking closely to several of these malware families to provide valuable information to our customers about these threats. Next to the more popular Android banking malware NCC Group’s Threat Intelligence team also watches new trends and new families that arise and could be potential threats to our customers.

One of these ‘newer’ families is an Android banking malware called SharkBot. During our research NCC Group noticed that this malware was disturbed via the official Google play store. After discovery NCC Group immediately notified Google and decided to share our knowledge via this blog post.

Summary

SharkBot is an Android banking malware found at the end of October 2021 by the Cleafy Threat Intelligence Team. At the moment of writing the SharkBot malware doesn’t seem to have any relations with other Android banking malware like Flubot, Cerberus/Alien, Anatsa/Teabot, Oscorp, etc.

The Cleafy blogpost stated that the main goal of SharkBot is to initiate money transfers (from compromised devices) via Automatic Transfer Systems (ATS). As far as we observed, this technique is an advanced attack technique which isn’t used regularly within Android malware. It enables adversaries to auto-fill fields in legitimate mobile banking apps and initate money transfers, where other Android banking malware, like Anatsa/Teabot or Oscorp, require a live operator to insert and authorize money transfers. This technique also allows adversaries to scale up their operations with minimum effort.

The ATS features allow the malware to receive a list of events to be simulated, and them will be simulated in order to do the money transfers. Since this features can be used to simulate touches/clicks and button presses, it can be used to not only automatically transfer money but also install other malicious applications or components. This is the case of the SharkBot version that we found in the Google Play Store, which seems to be a reduced version of SharkBot with the minimum required features, such as ATS, to install a full version of the malware some time after the initial install.

Because of the fact of being distributed via the Google Play Store as a fake Antivirus, we found that they have to include the usage of infected devices in order to spread the malicious app. SharkBot achieves this by abusing the '**Direct Reply**' Android feature. This feature is used to automatically send reply notification with a message to download the fake Antivirus app. This spread strategy abusing the Direct Reply feature has been seen recently in another banking malware called **Flubot**, discovered by ThreatFabric.

What is interesting and different from the other families is that SharkBot likely uses ATS to also bypass multi-factor authentication mechanisms, including behavioral detection like bio-metrics, while at the same time it also includes more classic features to steal user's credentials.

Money and Credential Stealing features

SharkBot implements the four main strategies to steal banking credentials in Android:

- *Injections (overlay attack)*: SharkBot can steal credentials by showing a WebView with a fake log in website (phishing) as soon as it detects the official banking app has been opened.
- *Keylogging*: Sharkbot can steal credentials by logging accessibility events (related to text fields changes and buttons clicked) and sending these logs to the command and control server (C2).
- *SMS intercept:* Sharkbot has the ability to intercept/hide SMS messages.
- *Remote control/ATS*: Sharkbot has the ability to obtain full remote control of an Android device (via Accessibility Services).

For most of these features, SharkBot needs the victim to enable the **Accessibility Permissions & Services**. These permissions allows Android banking malware to intercept all the accessibility events produced by the interaction of the user with the User Interface, including button presses, touches, TextField changes (useful for the keylogging features), etc. The intercepted accessibility events also allow to detect the foreground application, so banking malware also use these permissions to detect when a targeted app is open, in order to show the web injections to steal user's credentials.

Delivery

Sharkbot is distributed via the Google Play Store, but also using something relatively new in the Android malware: '**Direct reply**' feature for notifications. With this feature, the C2 can provide as message to the malware which will be used to automatically reply the incoming notifications received in the infected device. This has been recently introduced by **Flubot** to distribute the malware using the infected devices, but it seems **SharkBot** threat actors have also included this feature in recent versions.

In the following image we can see the code of SharkBot used to intercept new notifications and automatically reply them with the received message from the C2.

In the following picture we can see the 'autoReply' command received by our infected test device, which contains a shortten Bit.ly link which redirects to the Google Play Store sample.

We detected the **SharkBot** reduced version published in the Google Play on 28th February, but the last update was on 10th February, so the app has been published for some time now. This reduced version uses a very similar protocol to communicate with the C2 (RC4 to encrypt the payload and Public RSA key used to encrypt the RC4 key, so the C2 server can decrypt the request and encrypt the response using the same key). This SharkBot version, which we can call **SharkBotDropper** is mainly used to download a fully featured SharkBot from the C2 server, which will be installed by using the Automatic Transfer System (ATS) (simulating click and touches with the Accessibility permissions).

This malicious dropper is published in the Google Play Store as a fake Antivirus, which really has two main goals (and commands to receive from C2):

- **Spread the malware using 'Auto reply' feature**: It can receive an 'autoReply' command with the message that should be used to automatically reply any notification received in the infected device. During our research, it has been spreading the same Google Play dropper via a shorten Bit.ly URL.

- **Dropper+ATS**: The ATS features are used to install the downloaded SharkBot sample obtained from the C2. In the following image we can see the decrypted response received from the C2, in which the dropper receives the command '**b**' to download the full SharkBot sample from the provided URL and the ATS events to simulate in order to get the malware installed.

With this command, the app installed from the Google Play Store is able to install and enable Accessibility Permissions for the fully featured **SharkBot** sample it downloaded. It will be used to finally perform the ATS fraud to steal money and credentials from the victims.

The fake Antivirus app, the SharkBotDropper, published in the Google Play Store has more than 1,000 downloads, and some fake comments like 'It works good', but also other comments from victims that realized that this app does some weird things.

Technical analysis

Protocol & C2

The protocol used to communicate with the C2 servers is an HTTP based protocol. The HTTP requests are made in plain, since it doesn't use HTTPs. Even so, the actual payload with the information sent and received is encrypted using RC4. The RC4 key used to encrypt the information is randomly generated for each request, and encrypted using the RSA Public Key hardcoded in each sample. That way, the C2 can decrypt the encrypted key (**rkey** field in the HTTP POST request) and finally decrypt the sent payload (**rdata** field in the HTTP POST request).

If we take a look at the decrypted payload, we can see how SharkBot is simply using JSON to send different information about the infected device and receive the commands to be executed from the C2. In the following image we can see the decrypted RC4 payload which has been sent from an infected device.

Two important fields sent in the requests are:

- **ownerID**
- **botnetID**

Those parameters are hardcoded and have the same value in the analyzed samples. We think those values can be used in the future to identify different buyers of this malware, which based on our investigation is not being sold in underground forums yet.

Domain Generation Algorithm

SharkBot includes one or two domains/URLs which should be registered and working, but in case the hardcoded C2 servers were taken down, it also includes a Domain Generation Algorithm (DGA) to be able to communicate with a new C2 server in the future.

The DGA uses the current date and a specific suffix string ('pojBI9LHGFdfgegjjsJ99hvVGHVOjhksdf') to finally encode that in base64 and get the first 19 characters. Then, it append different TLDs to generate the final candidate domain.

The date elements used are:

- **Week of the year** (v1.get(3) in the code)
- **Year** (v1.get(1) in the code)

It uses the '+' operator, but since the week of the year and the year are Integers, they are added instead of appended, so for example: for the second week of 2022, the generated string to be base64 encoded is: 2 + 2022 + "pojBI9LHGFdfgegjjsJ99hvVGHVOjhksdf" = 2024 + "pojBI9LHGFdfgegjjsJ99hvVGHVOjhksdf" = "2024pojBI9LHGFdfgegjjsJ99hvVGHVOjhksdf".

In previous versions of SharkBot (from November-December of 2021), it only used the current week of the year to generate the domain. Including the year to the generation algorithm seems to be an update for a better support of the new year 2022.

Commands

SharkBot can receive different commands from the C2 server in order to execute different actions in the infected device such as sending text messages, download files, show injections, etc. The list of commands it can receive

and execute is as follows:

- **smsSend**: used to send a text message to the specified phone number by the TAs
- **updateLib**: used to request the malware downloads a new JAR file from the specified URL, which should contain an updated version of the malware
- **updateSQL**: used to send the SQL query to be executed in the SQLite database which Sharkbot uses to save the configuration of the malware (injections, etc.)
- **stopAll**: used to reset/stop the ATS feature, stopping the in progress automation.
- **updateConfig**: used to send an updated config to the malware.
- **uninstallApp**: used to uninstall the specified app from the infected device
- **changeSmsAdmin**: used to change the SMS manager app
- **getDoze**: used to check if the permissions to ignore battery optimization are enabled, and show the Android settings to disable them if they aren't
- **sendInject**: used to show an overlay to steal user's credentials
- **getNotify**: used to show the Notification Listener settings if they are not enabled for the malware. With this permissions enabled, Sharkbot will be able to intercept notifications and send them to the C2
- **APP_STOP_VIEW**: used to close the specified app, so every time the user tries to open that app, the Accessibility Service with close it
- **downloadFile**: used to download one file from the specified URL
- **updateTimeKnock**: used to update the last request timestamp for the bot
- **localATS**: used to enable ATS attacks. It includes a JSON array with the different events/actions it should simulate to perform ATS (button clicks, etc.)

Automatic Transfer System

One of the distinctive parts of SharkBot is that it uses a technique known as Automatic Transfer System (ATS). ATS is a relatively new technique used by banking malware for Android.

To summarize ATS can be compared with webinject, only serving a different purpose. Rather then gathering credentials for use/scale it uses the credentials for automatically initiating wire transfers on the endpoint itself (so without needing to log in and bypassing 2FA or other anti-fraud measures). However, it is very individually tailored and request quite some maintenance for each bank, amount, money mules etc. This is probably one of the reasons ATS isn't that popular amongst (Android) banking malware.

How does it work?

Once a target logs into their banking app the malware would receive an array of events (clicks/touches, button presses, gestures, etc.) to be simulated in an specific order. Those events are used to simulate the interaction of the victim with the banking app to make money transfers, as if the user were doing the money transfer by himself.

This way, the money transfer is made from the device of the victim by simulating different events, which make much more difficult to detect the fraud by fraud detection systems.

IoCs

Sample Hashes:

- a56dacc093823dc1d266d68ddfba04b2265e613dcc4b69f350873b485b9e1f1c (Google Play SharkBotDropper)
- 9701bef2231ecd20d52f8fd2defa4374bffc35a721e4be4519bda8f5f353e27a (Dropped SharkBot v1.64.1)

SharkBotDropper C2:

- hxxp://statscodicefiscale[.]xyz/stats/

'Auto/Direct Reply' URL used to distribute the malware:

- hxxps://bit[.]ly/34ArUxI

Google Play Store URL:

- https://play.google.com/store/apps/details?id=com.abbondioendrizzi.antivirus.supercleaner

C2 servers/Domains for SharkBot:

- n3bvakjjouxir0zkzmd[.]xyz (185.219.221.99)
- mjayoxbvakjjouxir0z[.]xyz (185.219.221.99)

RSA Public Key used to encrypt RC4 key in ***SharkBot***:

```
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA2R7nRj0JMouviqMisFYt0F2QnScoofoR7svCcjrQcTUe7tKKweDnSetdz1A+PLNtk7v
```

RSA Public Key used to encrypt RC4 Key in the Google Play ***SharkBotDropper***:

```
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAu9qo1QgM8FH7oAkCLkNO5XfQBUdl+pI4u2tvyFiZZ6hMZ07QnlYazgRmWcC5j5H2iV+
```
