Nexus Android Trojan Analysis Report

Archived: 2026-04-05 18:34:02 UTC

Overview

On the evening of March 21, 2023, a novel Android Trojan was detected by the monitoring system jointly developed by LianSecurity and Zhongrui Tianxia. After capturing samples through the RuiShi sandbox system, it was determined that this Android Trojan is highly likely to be a variant of the original Android banking Trojan, SOVA. Concurrently, the Italian security company Cleafy published a report titled "Nexus: A New Android Botnet?", confirming that the virus is indeed a variant of SOVA and renaming it as Nexus.

Sample Analysis

- **Sample Name:** Chrome.apk
- **Sample SHA256:** 376d13affcbfc5d5358d39aba16b814f711f2e81632059a4bce5af060e038ea4
- **Sample File Size:** 4,792,032 KB

Main Behavior List

- Delete specified applications and their data
- Install and launch arbitrary applications
- Hide its own app icon
- Uninstall protection
- Upload user SMS data and contact list
- Use SmsManager to send, delete, and cancel SMS notifications
- Make phone calls
- Retrieve and upload user cookie information, inject cookies, etc.
- Read and upload digital wallet information
- Record and upload keyboard input logs
- Query sensitive mobile data (such as stored emails, app account data, IMSI, and other phone information)
- Set device to silent mode
- Unlock screen
- Access specified URLs
- Attempt to disable administrator user
- Enable accessibility features
- Monitor phone reboot events
- Use DownloadManager to download files

Installation Test

Upon the Trojan's installation, a Chrome browser-like icon appears on the main screen of the mobile device, with slight differences from the actual Chrome icon. The Trojan's icon is smaller, but it is challenging to recognize this difference without a side-by-side comparison.

Once the Trojan is launched, the interface prompts the user to enable "Accessibility Features." When the user clicks anywhere on the interface, it automatically redirects to the system's "Accessibility Features" settings and enables this feature.

After enabling the "Accessibility Features," the program automatically pops up and requests to obtain "Device Administrator Privileges."

Once the malicious app gains device administrator privileges, it continuously collects user information in the background, making it difficult for users to detect its presence. Once the device administrator privileges are granted, users attempting to access the device administrator settings interface will find it quickly closes, making it impossible to revoke the permissions. Similarly, when performing operations through adb, the same issue occurs, and the interface instantly closes. This is because the malicious app has already monitored the opening action of the device administrator settings interface, preventing users from revoking its privileges. As a result, users need to enable root access to successfully uninstall this malicious app.

```
adb shell am start -S "com.android.settings/.Settings\$DeviceAdminSettingsActivity"
```

In-depth Sample Analysis

Basic Information

Before manually analyzing with Incinerator, through the "Basic Analysis" module, we discovered that the sample program has an encrypted shell. This implies that the malicious app's developer used an encryption method to protect their code, preventing analysis and reverse engineering. Simultaneously, we noticed that the signature information used "CN=Android Debug," which is inconsistent with a regular Chrome certificate. This might suggest that the malicious app's developer is attempting to disguise as a normal Chrome app, making it easier to deceive users and gain their trust.

Thanks to Incinerator's Apk permission analysis capabilities, we can obtain the corresponding permissions list in the Apk's detailed information.

In the app's permissions list, 13 of the permissions obtained by the sample are classified as "dangerous." Among them, a few permissions are particularly hazardous:

- Send SMS (SEND_SMS)
- Read SMS (READ_SMS)
- Receive SMS (RECEIVE_SMS)
- Read contacts (READ_CONTACTS)
- Write contacts (WRITE_CONTACTS)
- Read phone numbers (READ_PHONE_NUMBER)

Typically, common apps do not request permissions involving sensitive operations, such as modifying the contact list, reading, and sending text messages. These permissions are usually reserved for specialized communication software. However, when a malicious app obtains accessibility permissions, it can leverage this feature to automatically enable other permissions, including those that pose potential threats to user privacy and security.

Accessibility is a powerful feature within the Android system designed to help users with special needs better utilize their devices. However, this feature can also be abused by malicious apps to perform operations beyond user control. Once a malicious app obtains accessibility permissions, it can execute various operations without the user's knowledge, such as enabling other sensitive permissions, leading to user data theft and privacy invasion. Therefore, users should be cautious when granting accessibility permissions and avoid granting them to untrusted apps.

The list of permissions enabled through accessibility in the code is as follows:

- android.permission.READ_SMS: Allows the app to read SMS messages
- android.permission.SEND_SMS: Allows the app to send SMS messages
- android.permission.RECEIVE_SMS: Allows the app to receive SMS messages
- android.permission.READ_CONTACTS: Allows the app to read the contact list
- android.permission.WRITE_CONTACTS: Allows the app to edit the contact list
- android.permission.READ_PHONE_STATE: Allows the app to read the device's phone state and identity information
- android.permission.WRITE_EXTERNAL_STORAGE: Allows the app to write to external storage, such as an SD card
- android.permission.MODIFY_AUDIO_SETTINGS: Allows the app to modify audio settings
- android.permission.READ_EXTERNAL_STORAGE: Allows the app to read external storage, such as an SD card
- android.permission.INSTALL_PACKAGES: Allows the app to install other applications
- android.permission.CALL_PHONE: Allows the app to make phone calls
- android.permission.GET_ACCOUNTS: Allows the app to access the device's account list
- android.permission.READ_PHONE_NUMBERS: Allows the app to read the device's phone numbers
- android.permission.CLEAR_APP_CACHE: Allows the app to clear all cache files

As shown in the images above, the app first hardcodes the list of permissions to be enabled through accessibility and then requests these permissions from the system. In the PermissionsTask phase, the app listens for permission request actions. Once a permission request is detected, the app automatically clicks the "Agree" button on the permission request interface using accessibility.

Static Code Analysis

After using the Incinerator tool to automatically unpack the sample and analyze the malicious behavior code, we discovered the following main features:

1. Delete specified apps and their app data

The malicious app has the capability to delete other apps and their data, potentially affecting users' normal use of their phones and applications.

The `clearApp` method indeed deletes cache data related to a specific app package by executing the `pm clear package` command, including image cache, temporary files, and database cache. This helps clean up junk files on the device and frees up storage space. The `deleteThisApp` method uninstalls the app by triggering the `android.intent.action.DELETE` intent. When the system receives this intent, an uninstall confirmation interface pops up. Typically, the user needs to manually click the "Agree" button on this interface to complete the uninstall. However, since this malicious app has accessibility permissions, it can automatically click the "Agree" button when the uninstall confirmation interface appears, thus completing the uninstall operation without the user's knowledge. This approach further enhances the malicious app's stealth and destructiveness.

2. Install and launch arbitrary apps

The malicious app can install and launch other apps, potentially further spreading malware or directing users to malicious websites.

The installation and uninstallation of apps are indeed achieved through accessibility. This approach can conveniently automate the app installation and uninstallation process for users. The only difference is that to implement this feature, the malicious app needs to adapt to different manufacturers' package names and installation Activity names.

As a result, the malicious app can successfully execute installation and uninstallation operations on various devices, thus more covertly implementing its malicious behavior. This strategy grants the malicious app a broader attack capability on various devices.

3. Hide its app icon

To make it difficult to be discovered and uninstalled, the malicious app hides its app icon.

In this malicious app, the developer uses the `setComponentEnabledSetting` method to disable the Launcher Activity. As a result, users cannot operate or access the malicious app through the app icon (Launcher Icon) on the device's main screen. The `setComponentEnabledSetting` method can be used to enable or disable app components, such as Activity, Service, BroadcastReceiver, etc. In this case, the malicious app achieves the purpose of hiding itself by disabling Launcher Activity, making it harder for users to detect its presence. This approach further enhances the stealth of the malicious app, making it more difficult to be discovered and removed.

4. Upload sensitive information such as phone contacts

The malicious app can steal and upload user contacts, text messages, cookies, etc., potentially leading to user privacy leaks and financial loss.

As shown in the images above, the malicious app first accesses text message content through content://sms, then processes it through a series of business logic and integrates it into the data of a network request. In addition to the text message data, this request also contains information such as SIM card information, victim device's IP address, country, city, and device model. Finally, this data is sent to a specified server.

Through this method, the malicious app can steal user text messages and device information and send this data to the attacker. Attackers can use this information for various illegal activities, such as fraud, privacy theft, or even identity theft.

5. Use SmsManager to send text messages, delete text messages, cancel text message notifications, read text messages

5.1 Upload text messages

According to the above description, the malicious app listens for the system broadcast when a text message is received, extracts the received text message content from the broadcast, and then sends each text message to a remote server. After completing this process, the app also terminates the received text message broadcast to avoid being discovered by users or other applications. In the second image, super.execute refers to sending the collected text message data to the remote server.

This behavior indicates that the malicious app has taken a more aggressive approach to stealing user text messages. Users need to strengthen their awareness of such applications to avoid adverse effects on their privacy and security.

5.2 Send text messages

The app calls the system's SmsManager to send text messages.

6. Obtain user cookie information and upload, inject cookies, etc.’

As shown in the image above, read all cookies, upload them to the remote server, and use CookieManager to delete local cookies.

7. Read and upload digital wallet information

7.1 Read balance

Using accessibility features, read the character content displayed by the View representing the balance, which is the user's wallet balance.

7.2 Read seed phrase

Using accessibility features, read the content from the View representing the seed phrase.

7.3 Upload to server

Send encrypted wallet information to the remote server.

8. Record and upload keyboard input records

In the two images above, the first one listens to keyboard input and extracts data through accessibility features, while the second one uploads this data to the remote server.

9. Query sensitive information on mobile data (query stored email and app account data, IMSI, and other mobile information)

Use AccountManager to obtain account information and upload it to the remote server.

10. Mute the phone

Using the audio system server, set the phone to silent mode.

11. Listen for phone restart events

Listen for phone restart events, and the malicious app begins to work after the phone restarts.

12. Use DownloadManager to download APK and install

Download the APK and proceed with the installation.

13. Take photos, record videos

14. Read other documents

15. Network requests

All logs in the code are uploaded, and the server address for uploading comes from an "encrypted" string

Based on "aHR0cDovLzE5My40Mi4zMi44Ny8=" being a Base64 encoded string. After decoding, we obtained the URL "http://193.42.32.87/", which is a server address for receiving collected data. However, this URL cannot

be accessed in mainland China. Therefore, even if domestic users install this malicious app, their data will not be collected because the data cannot be successfully sent to the server. Additionally, the URLs "http://ip-api.com/json" and "https://icanhazip.com" identified by the URL string scanner cannot be accessed due to the GFW.

Conclusion

Overall, the main purpose of this malicious app is to steal user privacy and conduct fine-grained processing on the related privacy data. During the analysis process, a large number of special treatments for Chinese mobile phone manufacturers such as Huawei, Xiaomi, and OPPO were discovered, suggesting that this malicious app may primarily target Chinese users. To hide its malicious behavior, the app uses a simple encryption shell to conceal critical code. This approach is not common in overseas malicious apps. Considering the types of data stolen, this malicious app is highly harmful because it steals very sensitive information such as text messages, encrypted wallets, and cookies, which could impact user's financial security. In terms of technical means, the malicious app mainly utilizes Android's accessibility features and device administrator permissions. This method is relatively common in malicious apps. Users need to be vigilant and guard against the threats posed by such applications to their privacy and security.

IOC Indicators

SHA256: 376d13affcbfc5d5358d39aba16b814f711f2e81632059a4bce5af060e038ea4
724a56172f40177da76242ee169ac336b63d5df85889368d1531f593b658606b
f3fc80a8793e60a901da44b9ab315931699e64a4f3eddb8aba839fe860de46dc

ec5b083c017570f846f6925b7c79d9e5886525a9b7ba7e514dabad0325c0af5e

C2Server：

193.42.32.87

85.31.45.101

Source: https://liansecurity.com/#/main/news/RWt_ZocBrFZDfCElFqw_/detail