Hunting for Android 1-days

Analysis of the rooting ecosystem

Eugene Rodionov
Richard Neal
Lin Chen

android

Agenda

- Rooting providers on Android
- Kingroot technical details
  - Architecture & modus operandi
  - Analysis of network communication protocol
- Payload analysis
- Conclusion

Android Rooting Providers

- A rooting application
  - Gain privileged access to the device by breaking the security model
    - Manipulate software
      - Cheat at games
    - Customise the OS / treat it like a general-purpose computer

- Android “1-click rooting” rooting providers:
  - Kingroot
  - QihooRoot (aka 360 Root)
  - BaiduRoot (aka Easy Root)
  - Kingo Root



Kingroot

About Kingroot

- One of the biggest Android rooting providers to date
- Claimed to support more than 2500 device models in 2015
- Developed by KingRoot Studio, China
- It uses Local Privilege Elevation exploits
  - If we can detect them, we can look for malicious use
  - Understand what exploits are actually used in-the-wild

android

Kingroot: Architecture

- Kingroot provides two rooting options: pcRoot and mobileRoot
- The rooting engine (rksdk) is implemented in an additional jar module
  - stored encrypted in the application's assets
  - dynamically loaded during execution of the application
- Krsdk is a rooting SDK:
  - comes with a license and checks signer and package name of the app

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
  <entry key="channel_id">105006</entry>
  <entry key="package_name">com.kingroot.kinguser</entry>
  <entry key="cert_md5">191240FCB048127DB9110D1B30537FDE</entry>
</properties>
```

Kingroot: Rooting Device

- Drop rooting tools in the app's internal directory
  - su manager (SuperSU)
- Fingerprint device and request a list of 'solutions' for rooting
  - the request is done using SharkNetwork communication protocol
  - the downloaded 'solutions' are XML files with information on exploit location
- Download 'solution' and execute it
  - exploit a privilege escalation vulnerability
- Install rooting tools in /system (and recovery) partition
  - and optionally some other undesired software...

SharkNetwork Library

android

SharkNetwork: Overview

- Proprietary network library used by a number of apps on Play and off-market:
  - com.tencent.mobileqq, com.tencent.qqpimsecure
  - com.samsung.android.messaging, com.samsung.android.sm.devicesecurity.tcm
- Capable of sending data over raw TCP and HTTP protocols
- Employs encryption and compression
  - XTEA cipher
  - inflate/deflate (zlib)
- Designed to handle network communication in mobile environments
  - sudden loss of connection
  - reduced bandwidth

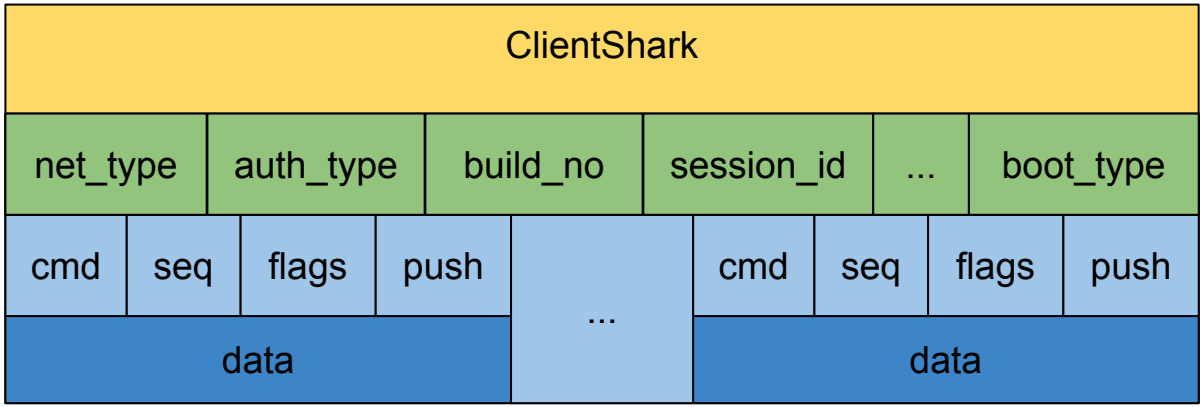
SharkNetwork: RE Challenges

- Heavy utilization of abstract types and inheritance
- Inherently multi-threaded and event-driven framework
- Kingroot uses simple obfuscation: method and field name mangling
- But…
  - there is some debugging/logging data left in dex files
  - there are other APKs that use an unobfuscated version of SharkNetwork library

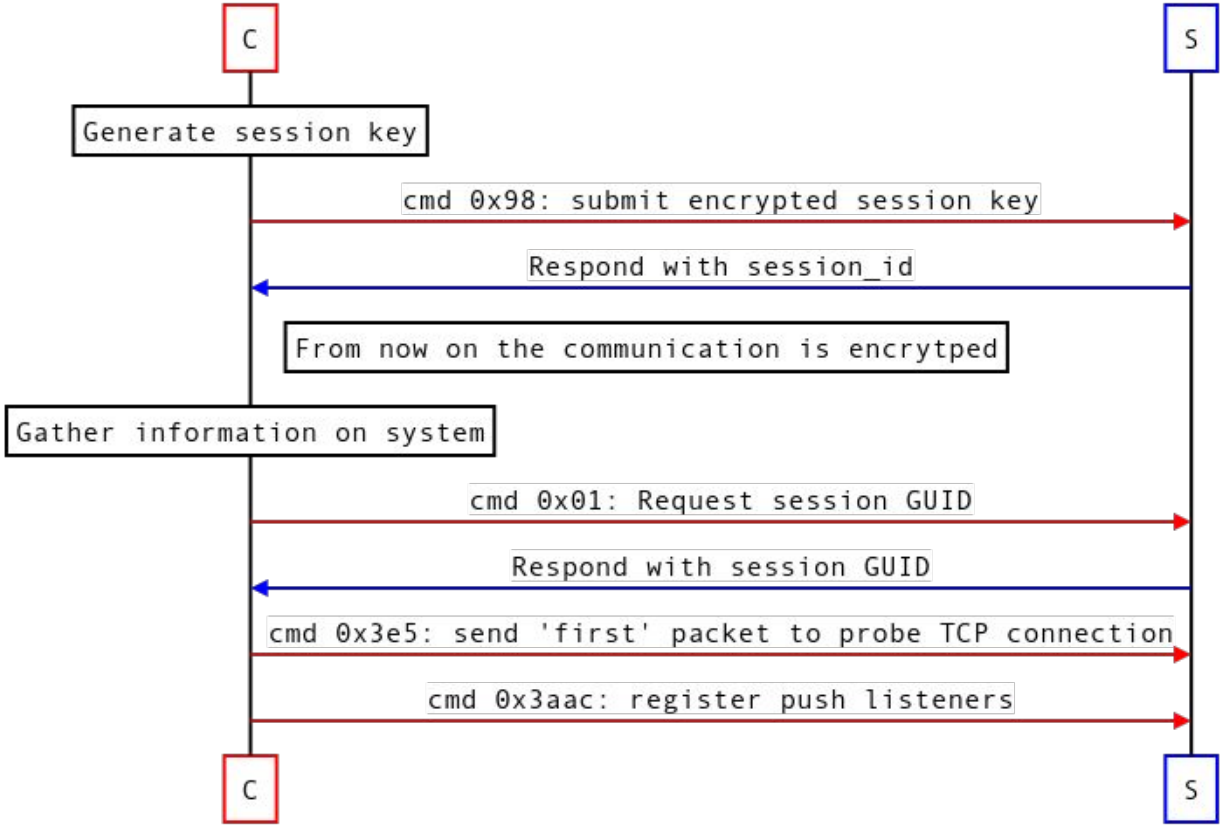
```
kgod.yn.c(new StringBuilder().append(p9).toString(), new
StringBuilder("[ocean][shark_funnel]|seqNo|seq_").append(p11).append("|step|").append(p13).append("|c
mdId|cmd_").append(p10).append("|stepTime|").append((System.currentTimeMillis() -
v0_2.l)).append("|retCode|").append(p14).append("|flow|").append(p15).toString());
```

SharkNetwork: Data Format

- **Below are names of data structures extracted from the unobfuscated version of the lib**
  - Client(Server)Shark -- the outermost object encapsulating everything else
  - Sharkfin -- an object to configure properties of the channel
  - Client(Server)Sashimi -- represents a unit of information exchange



SharkNetwork: Handshake



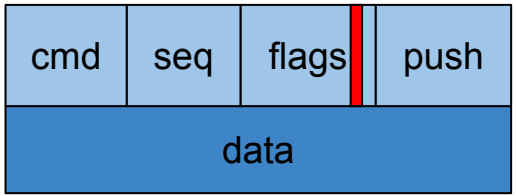
SharkNetwork: Encryption #1

- A session key is a 16-character randomly generated string
- SharkNetwork encrypts the session key with a hard-coded 1024-bit RSA key

```
private byte[] encryptSessionKey(String sessionKey) {
    if (!android.text.TextUtils.isEmpty(sessionKey)) {

        java.security.PublicKey v1_3 = java.security.KeyFactory.getInstance("RSA").generatePublic(new
java.security.spec.X509EncodedKeySpec(kgod.adc.a("MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDb49jFnNqMDLdl87Ut
Y5jOMqqdMuvQg65Zuva3Qm1tORQGBuM04u7fqygA64XbOx9e/KPNkDNDmqS8SlsAPL1fV2lqM/phgV0NY62TJqSR+PLngwJd2rhYR8wQ1N
0JE+R59a5c08EGsd6axStjHsVu2+evCf/SWU9Y/oQpEtOjGwIDAQAB", 0)));

        javax.crypto.Cipher v2_4 = javax.crypto.Cipher.getInstance("RSA/ECB/PKCS1Padding");
        v2_4.init(1, v1_3);
        v0_0 = v2_4.doFinal(sessionKey.getBytes());
    }
    return v0_0;
  }
```

SharkNetwork: Encryption #2

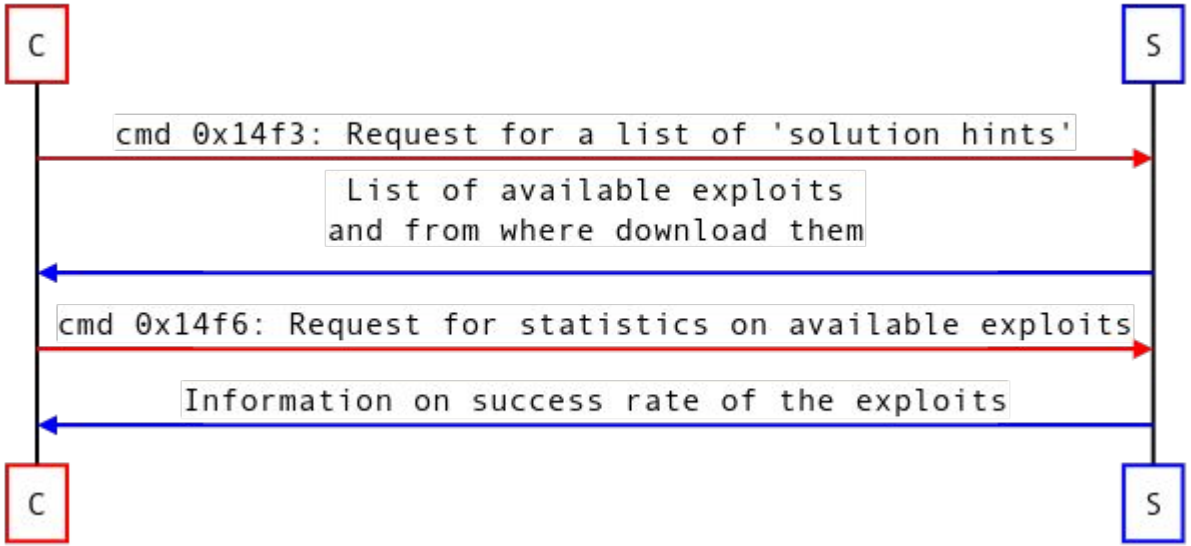
- It is possible to spoof server-to-client messages
  - Client(Server)Sashimi has a flag for data encryption



- The lib uses java.util.Random to generate session keys which provides only 48-bit entropy
  - they seem to switch to java.security.SecureRandom in recent versions of the SDK used in other apps

```
private String a(int p6) {
   java.util.Random v2_1 = new java.util.Random();
   StringBuffer v3_1 = new StringBuffer();
   while (v0_0 < p6) {
     v3_1.append("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789".charAt(v2_1.nextInt(62)));
     v0_0++;
   }
   return v3_1.toString();
 }
```

SharkNetwork: Requesting Exploits




```
PcAndMobileRootInfos v8 = (PcAndMobileRootInfos)arg8;
this.solutionsInfo.pcRoots = v8.solutionsInfo_pcRoots;
this.solutionsInfo.mobileRoots = v8.solutionsInfo_mobileRoots;
this.solutionsInfo.pcRoots.canRoot =  this.solutionsInfo.solutionListToDownload.length > 0 ? 1 : 0;
this.solutionsInfo.mobileRoots.canRoot = this.solutionsInfo.mobileRoots.succRate > 0 &&
                                         this.solutionsInfo.mobileRoots.succUsers > 0 ? 1 : 0;
```

SharkNetwork: Fingerprinting Device

- **To request exploits Kingroot & SharkNetwork submits the following info**
  - unique device ID -- IMEI for GSM and MEID or ESN for CDMA
  - Wi-Fi MAC address
  - Android ID
  - CPU information -- cat /proc/cpuinfo, number of cores, maximum CPU frequency
  - screen size of the device
  - amount of available memory -- /proc/meminfo
  - Total size and amount of available space on system and data partitions
  - Total size and amount of available space on the external storage
  - Device build information
  - Version of baseband firmware
  - device brand, manufacturer, product name and release version number
  - etc.

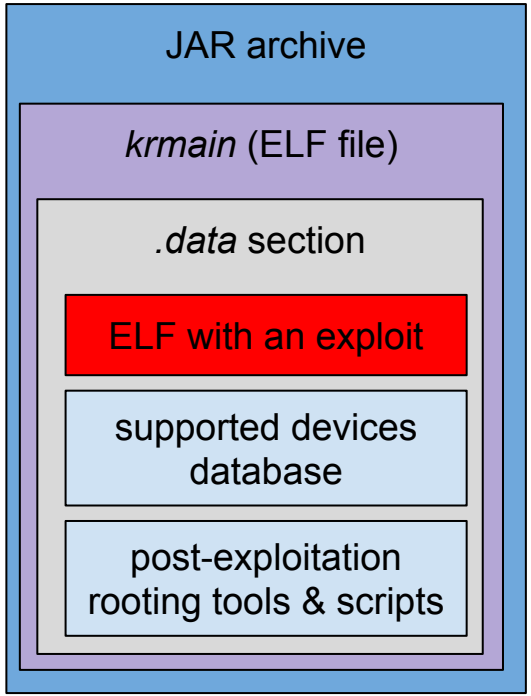
SharkNetwork: Solution Info

```
<root>
  <root_zip>
    <name>00000000000000000001490768428987.jar</name>
    <sid>2909</sid>
    <version>428166</version>
    <type>1</type>
    <size>2306184</size>
    <verified>1</verified>
    <md5>0a14d62d7c0795e4c8edbe759e0c3ee6</md5>
    <encrypt>1</encrypt>
    <url>
      hxxp://king.myapp.com/myapp/Kingroot/webapp_kingroot/solution_test/00000000000000000001490768428987.jar
    </url>
    <backup_url>
      hxxp://king.myapp.com/myapp/Kingroot/webapp_kingroot/solution_test/00000000000000000001490768428987.jar
    </backup_url>
    <exploit_type>0</exploit_type>
    <interface_type>3</interface_type>
  </root_zip>
</root>
```

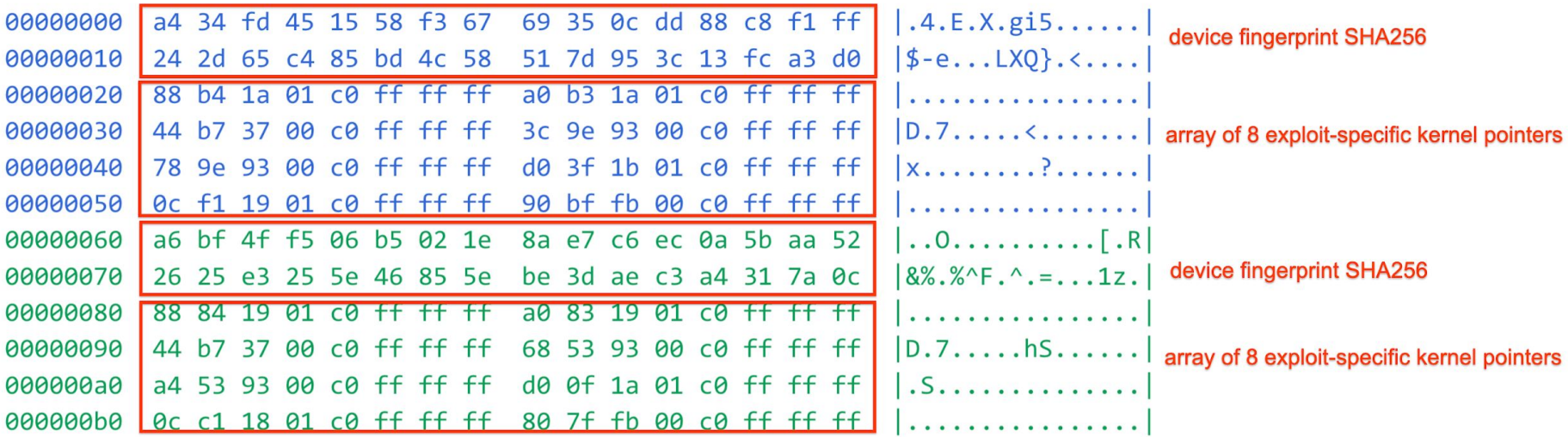
Payload analysis

Exploit Payload Structure

- **A JAR file, which contains**
  - An ELF file, which contains (lightly scrambled)
    - The exploit supported-device DB
    - ELF file(s), (one of) which contains (dynamically-generated substitution cipher)
      - Post-exploitation ELFs and scripts
      - Sometimes Obfuscator-LLVM has been used, sometimes not



Supported-Device DB



Fingerprint Value

- SHA256 over a string from the device:
  - ro.product.brand | ro.product.model | cat /proc/version
- … so make a lookup table

google	pixel XL	Linux version 3.10.73-g368902f (android-build@kpfj9.cbf.corp.google.com) (gcc version 4.9.x-google 20140827 (prerelease) (GCC)) #1 SMP PREEMPT Fri Mar 24 18:12:38 UTC 2017	cb062aa1d378
google	pixel xl	Linux version 3.18.52-g06de20685970 (android-build@wphl8.hot.corp.google.com) (gcc version 4.9.x 20150123 (prerelease) (GCC)) #1 SMP PREEMPT Fri Jun 23 19:50:11 UTC 2017	e65de8ab8523
google	pixel	Linux version 3.18.52-g99dda0323132 (android-build@wprf7.hot.corp.google.com) (gcc version 4.9.x 20150123 (prerelease) (GCC)) #1 SMP PREEMPT Fri Aug 18 00:56:04 UTC 2017	fa4d4d4ffc7ad
google	pixel XL	Linux version 3.10.73-g38c3e2f (android-build@vpef9.mtv.corp.google.com) (gcc version 4.9.x-google 20140827 (prerelease) (GCC)) #1 SMP PREEMPT Mon Sep 19 22:06:43 UTC 2016	115f5268a02bc
google	pixel	Linux version 3.10.73-ga51b1600b7f8 (android-build@xpce6.ams.corp.google.com) (gcc version 4.9.x-google 20140827 (prerelease) (GCC)) #1 SMP PREEMPT Thu Aug 17 22:07:37 UTC 2017	ee939e188c21

Some Exploits

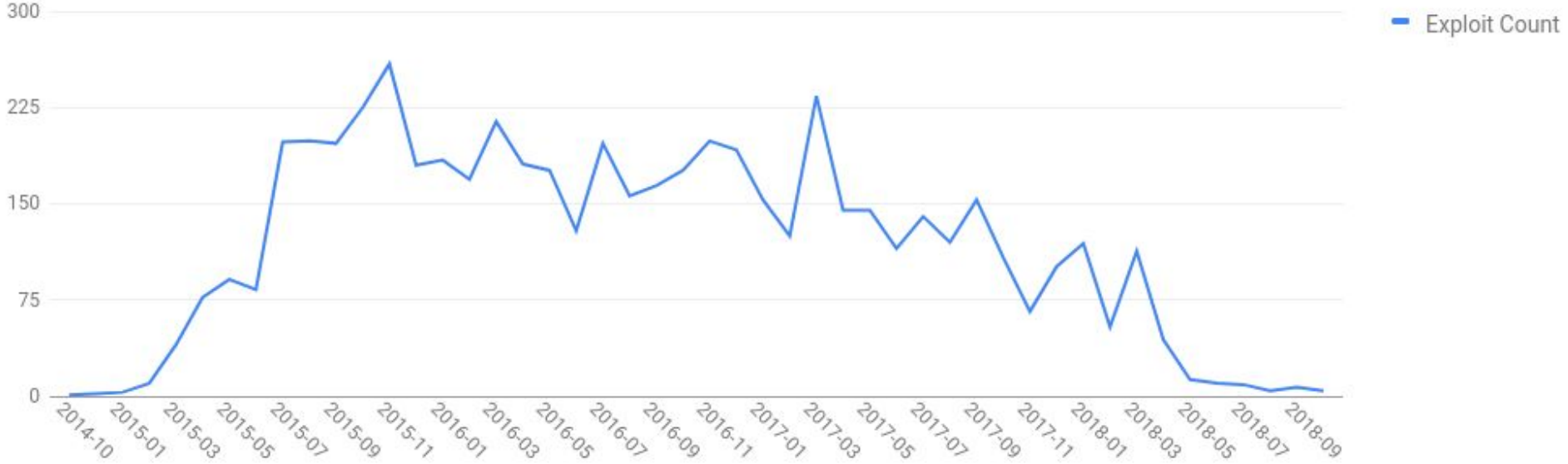
- 909 / 910 - CVE-2015-1805 aka PipeIOV
- 947 - unknown
- 1511 - CVE-2017-0403 aka izanami
- 840 / 1512 - CVE-2016-5195 aka DirtyCow
- 1514 - CVE-2017-0569
- 1523 - CVE-2017-7533
- 1545 - unknown

947

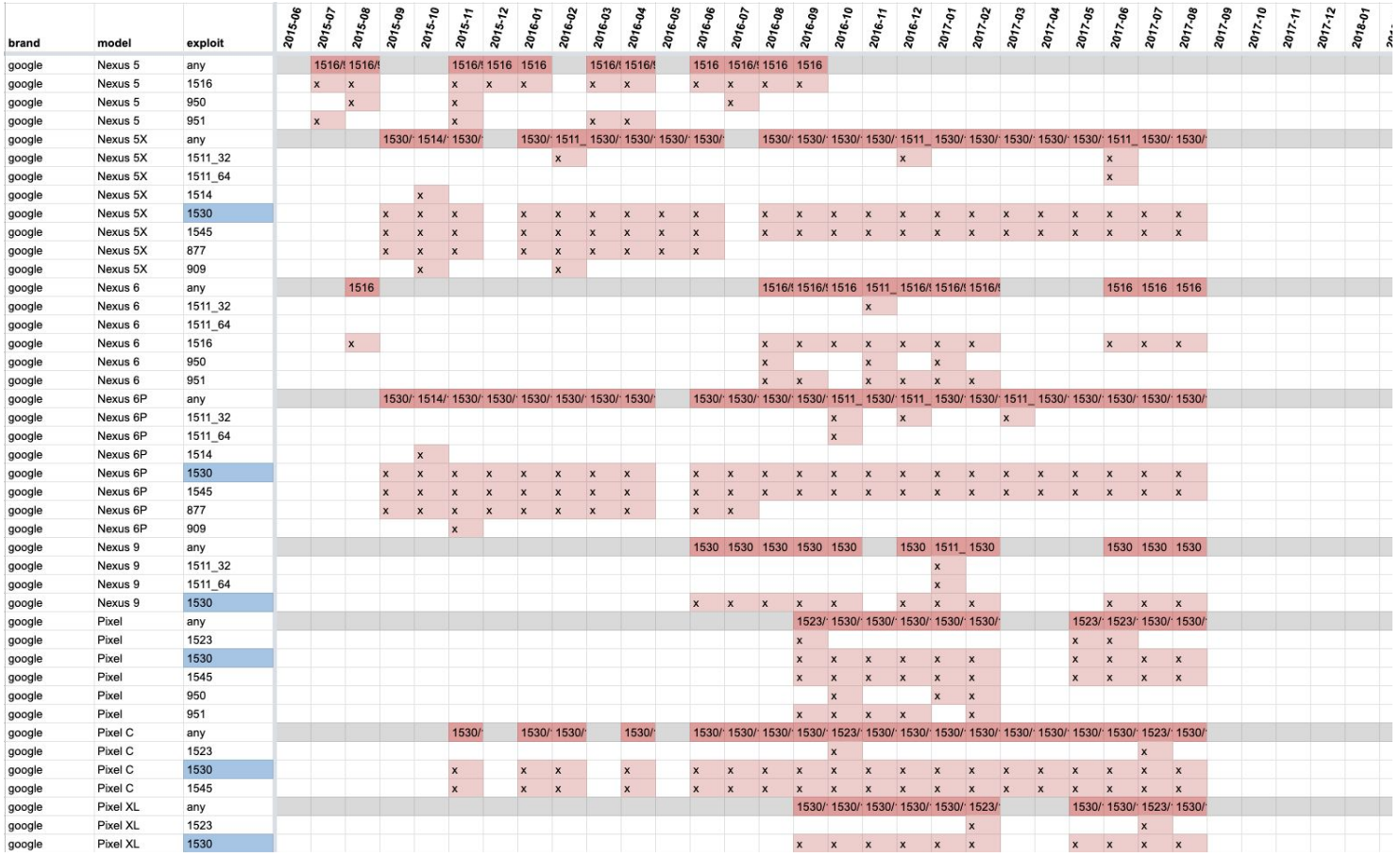
- ioctl MTK_M4U_T_CONFIG_MAU
  - No information found in Google search
  - Seems to allow read/write of kernel memory **after "legitimately" setting some device parameters**
  - Very limited device support
    - 73 kernels in total, 52 identified so far
    - 4 OEMs
- Aug 2016 - May 2017
- CVE-2017-0506 uses MTK_M4U_T_CONFIG_TF
  - Also see CVE-2017-0500/0501/0502/0503/0504/0505

```
                        ; CODE XREF: set_m4u_config+C8↑j
LDR         W0, [SP,#0x60+fd_proc_m4u]
MOV         W1, #0x4004671B ; MTK_M4U_T_CONFIG_MAU

MOV         X2, X19
BL          wrap_sys_ioctl
LDR         W8, [SP,#0x60+var_54]
ADD         W9, W8, #1
MOV         W8, #0x9D4C89A9

B           fsm_start
```

Count of Device Models Covered



Vulnerability of Google Devices

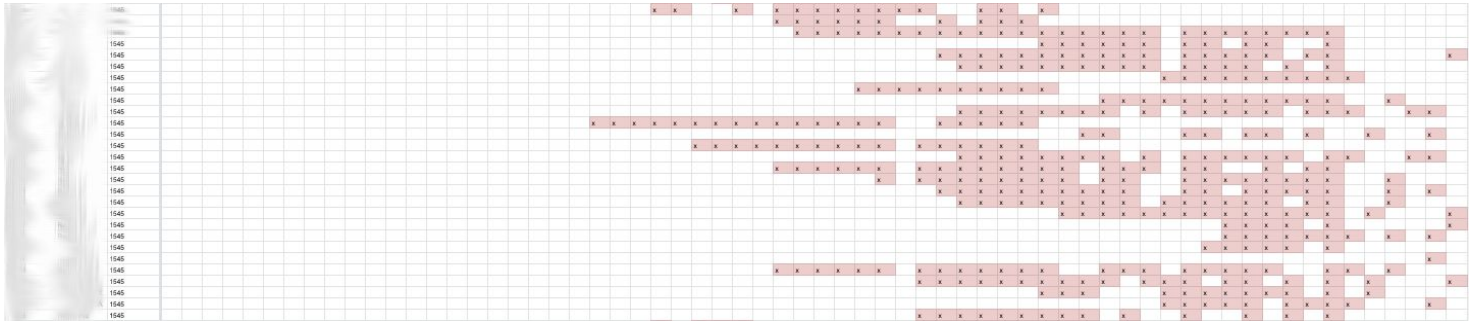


brand	model	exploit	2015-06	2015-07	2015-08	2015-09	2015-10	2015-11	2015-12	2016-01	2016-02	2016-03	2016-04	2016-05	2016-06	2016-07	2016-08	2016-09	2016-10	2016-11	2016-12	2017-01	2017-02	2017-03	2017-04	2017-05	2017-06	2017-07	2017-08	2017-09	2017-10	2017-11	2017-12	2018-01
google	Nexus 5	any		1516/	1516/			1516/	1516	1516		1516/	1516/		1516	1516/	1516	1516															
google	Nexus 5	1516		x	x			x	x	x		x	x		x	x	x	x															
google	Nexus 5	950			x			x								x																	
google	Nexus 5	951		x				x				x	x																				
google	Nexus 5X	any				1530/	1514/	1530/		1530/	1511_	1530/	1530/	1530/	1530/		1530/	1530/	1530/	1530/	1511_	1530/	1530/	1530/	1530/	1530/	1511_	1530/	1530/					
google	Nexus 5X	1511_32									x										x						x						
google	Nexus 5X	1511_64																									x						
google	Nexus 5X	1514					x																										
google	Nexus 5X	1530				x	x	x		x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x					
google	Nexus 5X	1545				x	x	x		x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x					
google	Nexus 5X	877				x	x	x		x	x	x	x	x	x																		
google	Nexus 5X	909					x				x																						
google	Nexus 6	any			1516												1516/	1516/	1516	1511_	1516/	1516/	1516/				1516	1516	1516					
google	Nexus 6	1511_32																		x													
google	Nexus 6	1511_64																															
google	Nexus 6	1516			x												x	x	x	x	x	x	x				x	x	x					
google	Nexus 6	950															x			x		x											
google	Nexus 6	951															x	x		x	x	x	x										
google	Nexus 6P	any				1530/	1514/	1530/	1530/	1530/	1530/	1530/	1530/		1530/	1530/	1530/	1530/	1511_	1530/	1511_	1530/	1530/	1511_	1530/	1530/	1530/	1530/	1530/					
google	Nexus 6P	1511_32																	x		x			x									
google	Nexus 6P	1511_64																	x														
google	Nexus 6P	1514					x																										
google	Nexus 6P	1530				x	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					
google	Nexus 6P	1545				x	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					
google	Nexus 6P	877				x	x	x	x	x	x	x	x		x	x																		
google	Nexus 6P	909						x																									
google	Nexus 9	any													1530	1530	1530	1530	1530		1530	1511_	1530				1530	1530	1530					
google	Nexus 9	1511_32																				x											
google	Nexus 9	1511_64																				x											
google	Nexus 9	1530													x	x	x	x	x		x	x	x				x	x	x					
google	Pixel	any																1523/	1530/	1530/	1530/	1530/	1530/			1523/	1523/	1530/	1530/					
google	Pixel	1523																x								x	x						
google	Pixel	1530																x	x	x	x	x	x			x	x	x	x					
google	Pixel	1545																x	x	x	x	x	x			x	x	x	x					
google	Pixel	950																	x			x	x										
google	Pixel	951																x	x	x	x		x										
google	Pixel C	any						1530/		1530/	1530/		1530/		1530/	1530/	1530/	1530/	1523/	1530/	1530/	1530/	1530/	1530/	1530/	1530/	1530/	1523/	1530/					
google	Pixel C	1523																	x														
google	Pixel C	1530						x		x	x		x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					
google	Pixel C	1545						x		x	x		x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					
google	Pixel XL	any																1530/	1530/	1530/	1530/	1530/	1523/			1530/	1530/	1523/	1530/					
google	Pixel XL	1523																					x					x					
google	Pixel XL	1530																x	x	x	x	x	x			x	x	x	x					

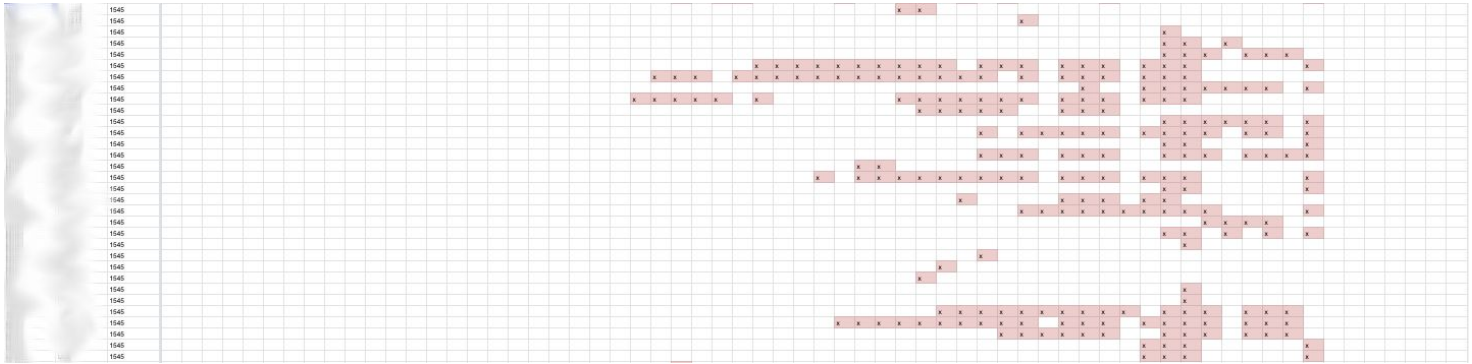
android

1545 - 2013/07 -> 2018-10

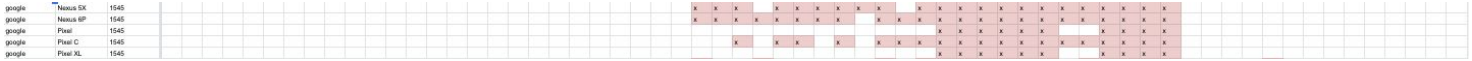
OEM A



OEM B



Google



Thoughts on KingRoot

- **Exploit 1530 supports 5482 kernels**
  - At least 35 device manufacturers
  - At least 300 device models
- **An unknown amount of automation**
  - Obtaining/reversing kernels
  - Testing
- **We only have partial data for exploits and device fingerprints so far**

State of Android Rooting Providers

- There is a decline in popularity of rooting tools for modern Android devices
- Most of the exploits used in the rooting providers target pre-Oreo Android versions
- Rooting providers:
  - Kingroot
    - shut down its operations as of December 2019
  - QihooRoot
    - shut down its operations early in 2020

Thank you! Questions?