DATE: 01/02/2019

# KAI OS Forensics for Money and Profit
# By Richard F. McQuown

If you are reading this and are expecting to find a nice easy tool to parse your KAI OS extractions you might as well stop reading. The KAI OS is unique operating system. KAI OS uses some data obfuscation techniques that I haven't seen previously "in the wild". Hopefully this guide will help you manually find usable data from an extraction. You will definitely be more likely to find usable data if you have some other external evidence, like toll records or another person's phone extraction, to correlate and confirm your forensic findings from a KAI OS device forensic examination. You have to be willing to get your bytes dirty.

BAD NEWS UP FRONT : I haven't figured out how to convert the Date/Time stamps from the SQLite databases. I will show you what I believe are the Time/Date Stamps for SMS Messages and Calls. Maybe you can figure them out!

The data in this document was taken from an Alcatel A405DL, running KAI OS 2.5, which I purchased locally for $9.99.

History:
The KAI OS is a cell phone operating system for low end feature phones. The KAI OS had many previous names like FireFox OS and GAIA. See the Wikipedia page for more historical information https://en.wikipedia.org/wiki/Firefox_OS.

I believe one thing is certain. KAI OS will not be going away. It actually appears to be gaining market share and traction in the cell phone arena especially after Google's recent 22 million dollar investment. https://www.theverge.com/2018/6/28/17513036/google-kaios-investment-feature-phones-firefox-os-apps-services-strategy

Extraction Methods:
You can currently chip-off the feature phones running KAI OS. The file system will parse using any decent forensic tool but I don't know of anything that will parse the user data. I have been able to connect via ADB to one KAI OS phone but was unable to get to the USERDATA. The only way I've been able to get the data out of locked KAI OS device is with Chip-off. I haven't looked for ISP/JTAG options. There is some documentation, from overseas, about putting the phone into ADB Mode but I haven't see that option in the Settings Menu in USA releases. FUTURE GOAL: ADB access to the USERDATA to side-load SQLite databases on to lab devices.

KAI OS Versions and Builds:
Just like forensic investigations on other operating systems like Android or Windows, the investigator needs to know which KAI OS Version they are working on. I have seen Versions 1 and 2.5 of the KAI OS. The file system appears to be very similar but there are some variations. This document is specific to KAI OS Version 2.5 unless specifically stated. You also may see many remnants of the words "Android" and "SELinux" in the data.

SQLite File Naming Convention:
The first thing that is unique about KAI OS is that the data we want is saved by obfuscated file names.
I call this "Up Odd/Down Even" naming convention. The first example is the file "226660312ssm.sqlite".

1 2 3
s s m is read "Up Odd / Down Even" 1,3,2 or SMS

The file 3406066227csotncta.sqlite

1 2 3 4 5 6 7 8 1 3 5 7 8 6 4 2
c s o t n c t a c o n t a c t s

The 2584670174dsitanleecreR.sqlite

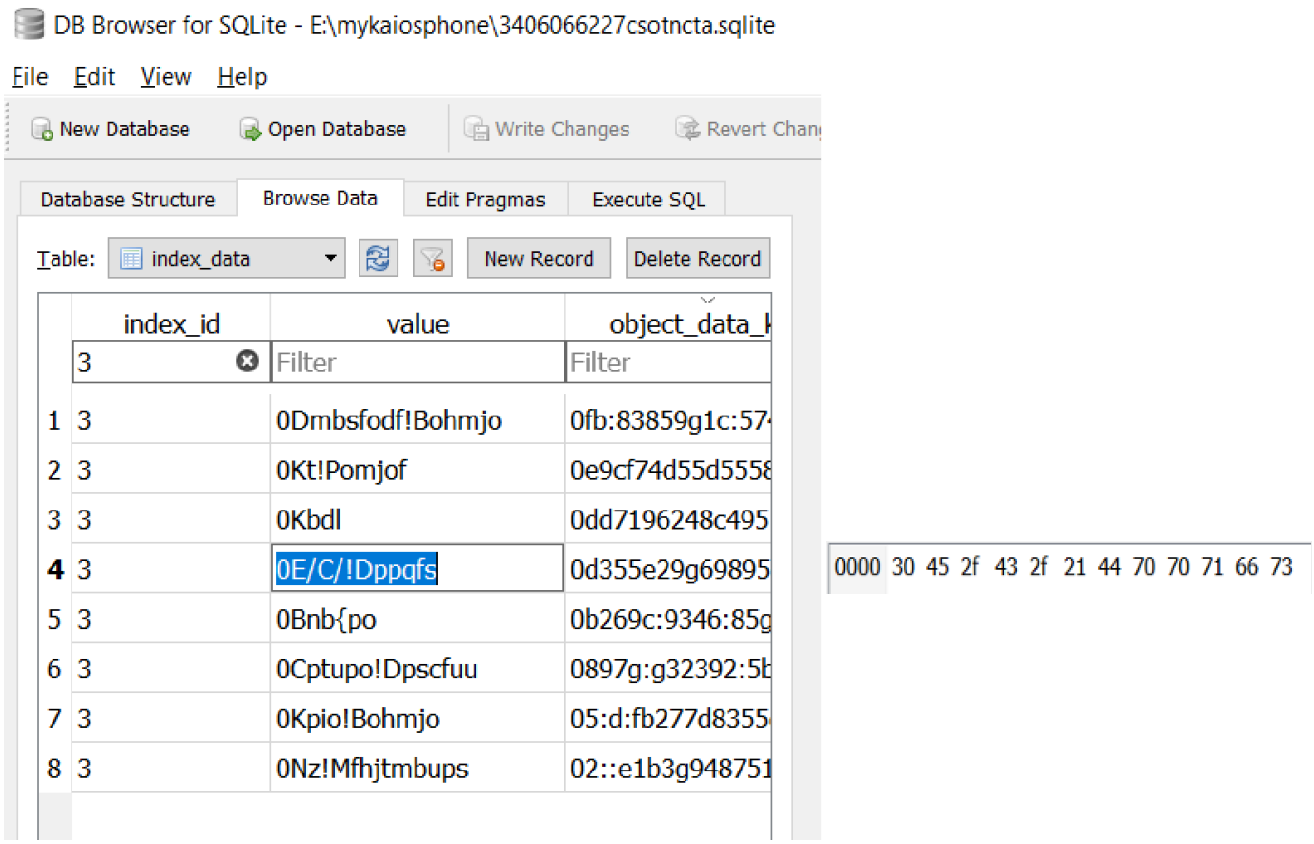
1 2 3 4 5 6 7 8 9 10 11 12 13 1 3 5 7 9 11 13 12 10 8 6 4 2
d s i t a n l e e c r e R d i a l e r R e c e n t s

The path to the above listed SQLite files is "root\local\storage\permanent\chrome\idb\" or "root\local\storage\default\107+f+app+++communications.gaiamobile.org\". I have no idea why the previous FireFox OS uses a directory with the name of "chrome".

There is a list of un-obfuscated SQLite Database Names on the CheetSheet accompanying this document. When discussing the databases in the rest of this document I will refer to them by their un-obfuscated names without the leading numbers. It is my educated guess that the number in the file names have something to do with numerating different versions of the databases.

SQLITE DATA:
You can easily open the dialerRecent.sqlite file with any SQLite Browser but when you rummage through the tables it doesn't appear to be any useful information. Make sure you open you extracted databases in your SQLite Browser as "Open Database Read Only".

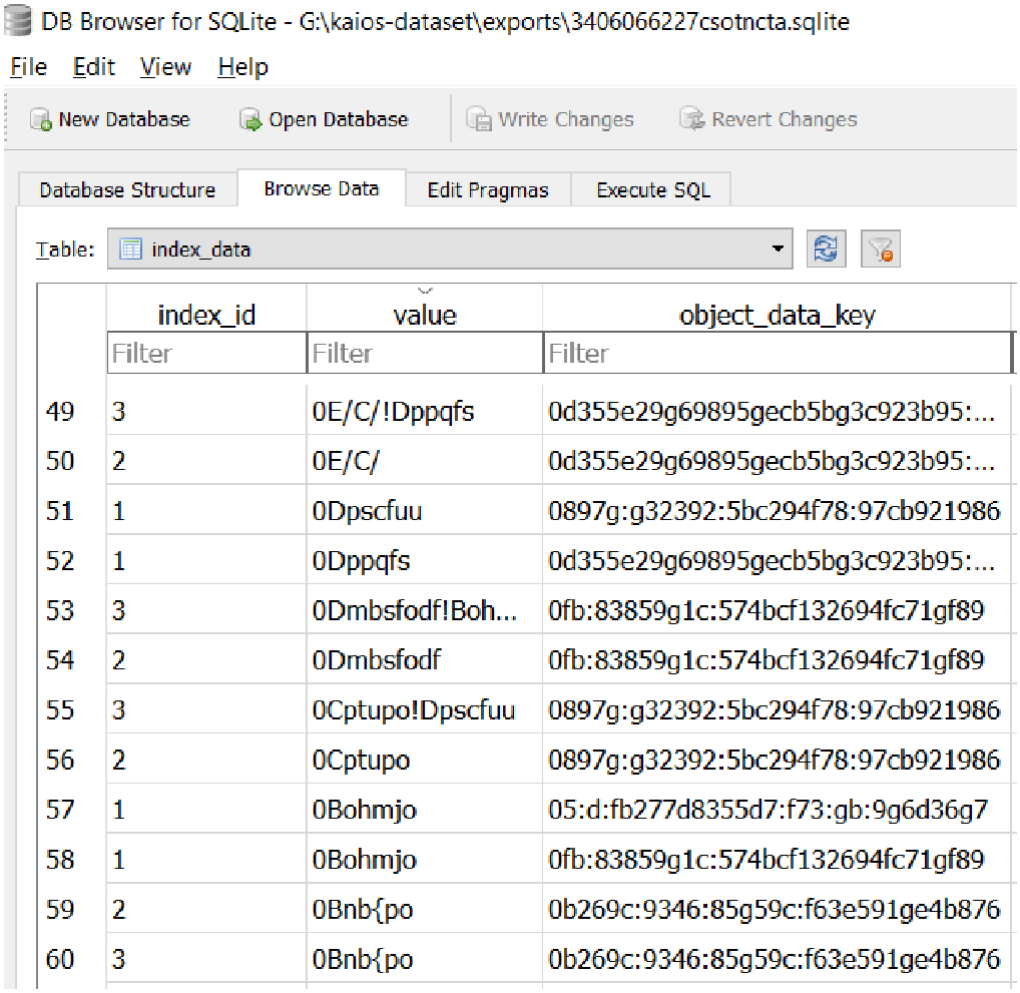
Much of the data in the SQLite databases are obfuscated with ROTATION-1 (ROT-1).




Record 4 value (highlighted above) is ”D.B. Cooper”. The leading “0” is really just the browser representing \x30. Next to Record 4 is the corresponding hexadecimal value of the field. Notice the punctuation of \x2F and \x21 If you UN-ROT -1 them you will have of ASCII \x2E “.” and \x20 (space)).

In the above picture I have the index_id field filtered to 3.

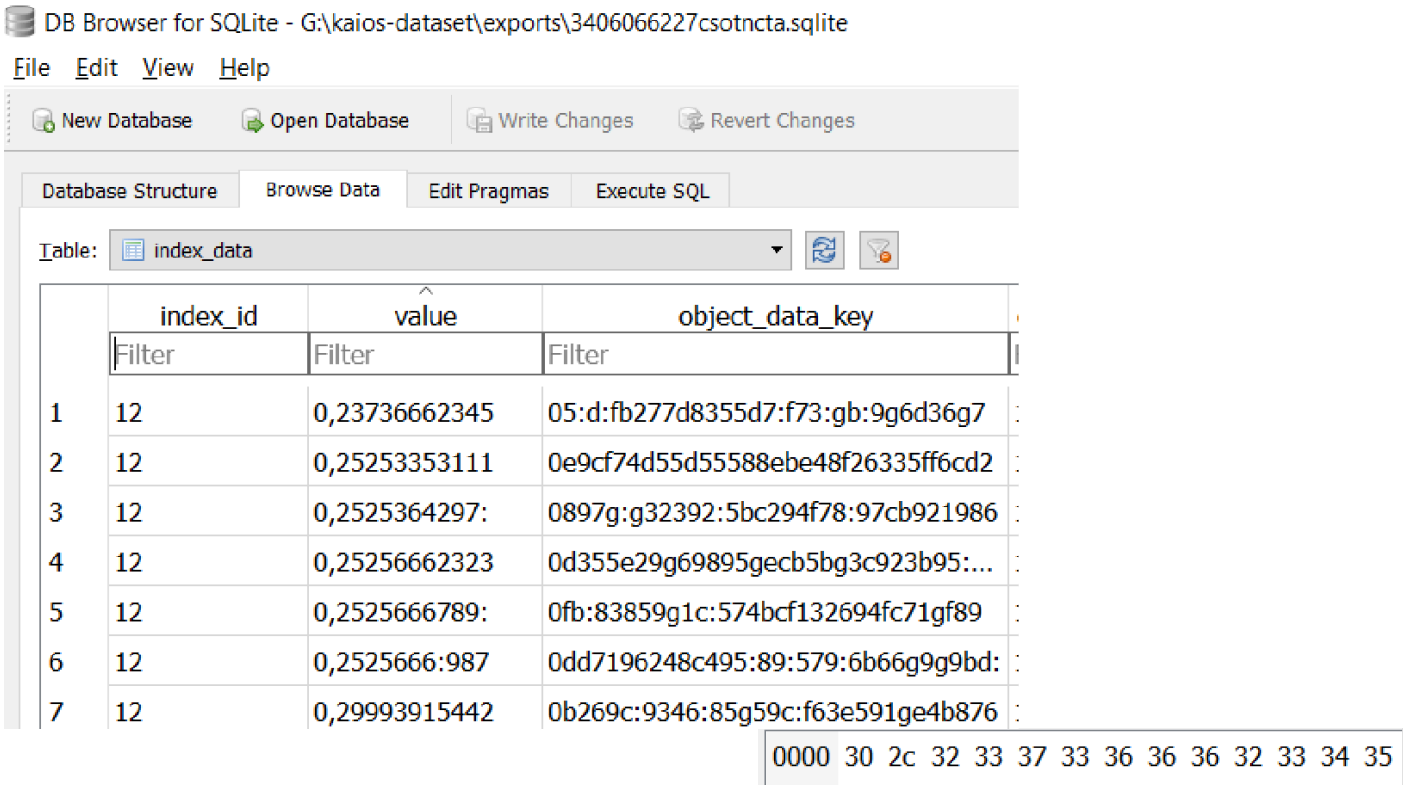
Decipher Value Field in Record 5 ( ASCII “Z” is represented as “}”) and order something geeky for yourself.




| | index_id | value | object_data_key |
|---|---|---|---|
| | Filter | Filter | Filter |
| 49 | 3 | 0E/C/!Dppqfs | 0d355e29g69895gecb5bg3c923b95:... |
| 50 | 2 | 0E/C/ | 0d355e29g69895gecb5bg3c923b95:... |
| 51 | 1 | 0Dpscfuu | 0897g:g32392:5bc294f78:97cb921986 |
| 52 | 1 | 0Dppqfs | 0d355e29g69895gecb5bg3c923b95:... |
| 53 | 3 | 0Dmbsfodf!Boh... | 0fb:83859g1c:574bcf132694fc71gf89 |
| 54 | 2 | 0Dmbsfodf | 0fb:83859g1c:574bcf132694fc71gf89 |
| 55 | 3 | 0Cptupo!Dpscfuu | 0897g:g32392:5bc294f78:97cb921986 |
| 56 | 2 | 0Cptupo | 0897g:g32392:5bc294f78:97cb921986 |
| 57 | 1 | 0Bohmjo | 05:d:fb277d8355d7:f73:gb:9g6d36g7 |
| 58 | 1 | 0Bohmjo | 0fb:83859g1c:574bcf132694fc71gf89 |
| 59 | 2 | 0Bnb{po | 0b269c:9346:85g59c:f63e591ge4b876 |
| 60 | 3 | 0Bnb{po | 0b269c:9346:85g59c:f63e591ge4b876 |

In the above picture is from “Index_Data” table from the“Contacts” database. The “Value” Column contains the ROT-1 Name of the Contacts. The “object_data_key” value appears to contain a unique key for variations of the name values. There is also a lead \x30 represent by a “0” in the browser. There are multiple entries for the same name.

Record 50, index_id 2 represents the First Name Field of the Contacts and is “D.B.”
Record 49, index_id 3 represents the First Name and Last Name Field of the Contacts and is “D.B. Cooper”
Notice Record 49 and 50 have the have the same object_data_key(d355e2~). You will see other variations of entries like with all lower case entries. For example index_id 5 contains obfuscated “d.b.” and index_id 6 contains “d.b. cooper”. I like to view the contact names using “Index_Data” table from the “Contacts” database as Index_id 3. But you have to look at all the data to make sure you aren't missing anything. All the different Index_id values can be found in the object_store_index table.




Contact entries without names also appear in the Contacts database as just numbers. The telephone numbers are listed in the Values Field in ROT-1. Record 1 is listed as text "0,23736662345" which is equals \x30 +12625551234. The ASCII symbol ":" (\x58 ) in record 3 value converts to a "9".

SOME EASY WINS:

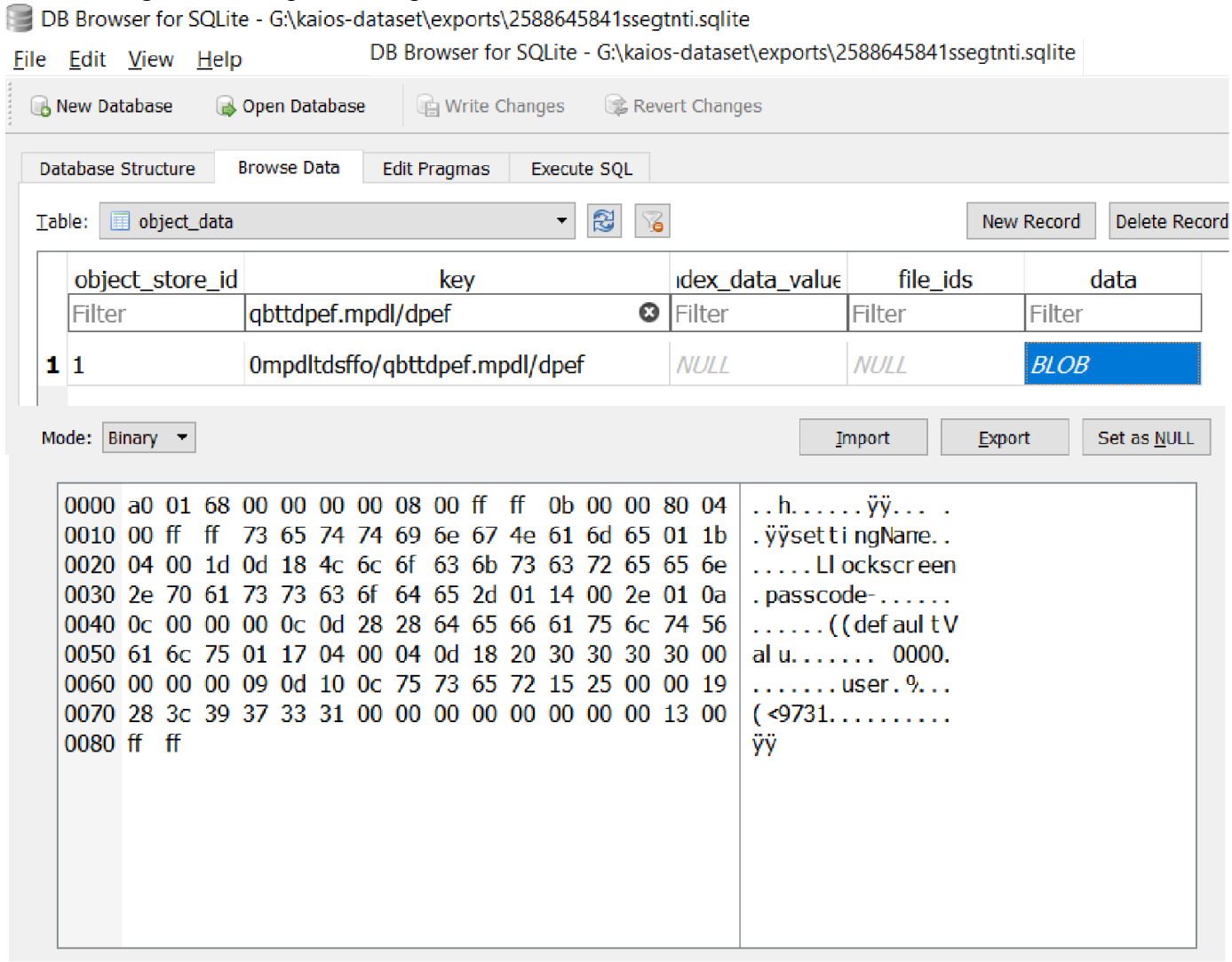
1. PASSCODE
Location: \data\local\storage\permanent\chrome\idb\2588645841ssegtnti.sqlite
Decoded: mpdltdsffo/qbttdpef.mpdl/dpef = lockscreen.passcode.lock.code

Method 1. Open 2588645841ssegtnti.sqlite (settings) in a SQLite Browser, Open object_data table, filter key value mpdltdsffo/qbttdpef.mpdl/dpef. Look at the blob data in binary.

Method 2. Use the Search Terms below on extraction:
Search Term: mpdltdsffo/qbttdpef.mpdl/dpef or
Search Term: lockscreen.passcode.lock.code

Method 3. Use the ENCASE GREP Search Terms below on extraction :
ENCASE GREP: Llockscreen.passcode-.{59,60} or
ENCASE GREP: mpdltdsffo/qbttdpef.mpdl/dpef.{117,117}

The following is an Example of using Method 1 to find the Device Passcode.




The plain text passcode “9731” is at offset \x72 in the blob view-able in binary mode.

2. TELEPHONE NUMBER of DEVICE

Location: \data\local\storage\permanent\chrome\idb\2588645841ssegtnti.sqlite

Decoded: sjm/nnt/tfoefs = rin.mms.sender

Method 1. Open 2588645841ssegtnti.sqlite (settings) in a SQLite Browser, Open object_data table, filter key value sjm/nnt/tfoefs. Look at the blob data in binary.

Method 2. Use the Search Term sjm/nnt/tfoefs

Method 3. Use the ENCASE GREP sjm/nnt/tfoefs.{101,101}

3. WIFI AP NAMES and PASWORDS

Location: data\misc\wifi\wpa_supplicant.conf

Method 1: Open file with any tool and read contents.

Example:
ctrl_interface=/data/misc/wifi/sockets disable_scan_offload=1 update_config=1 p2p_disabled=1 p2p_no_group_iface=1 hs20=1 interworking=1 pmf=1 external_sim=1 network={
ssid="WiFiAp4321" psk="password4321" key_mgmt=WPA-PSK frequency=2412 }

4. IMEI

Location: \data\property\persist.product.imei

Method 1: Open file with any tool and read contents.

5. MAC Address

Location 1 \data\persist\data\param\macaddr

Location 2: \data\local\storage\permanent\chrome\idb\2588645841ssegtnti.sqlite

Decode: efwjdfjogp/nbd = deviceinfo.mac

Method 1: Open data\persist\data\param\macaddr with any tool and read contents.

Method 2: Use SQLite viewer to open on 2588645841ssegtnti.sqlite. Select object_data table, filter key value efwjdfjogp/nbd. Look at the blob data in binary.

Method 3: Use Search Term = efwjdfjogp/nbd

Method 4: Use Encase GREP = efwjdfjogp/nbd.{106,106}

6. KaiOS Version

Location: \data\local\storage\permanent\chrome\idb\2588645841ssegtnti.sqlite

Decoded: sefwjdfjogp/tpguxbsf = rdeviceinfno.software

Method 1. Open 2588645841ssegtnti.sqlite (settings) in a SQLite Browser, Open object_data table, filter key value sefwjdfjogp/tpguxbsf . Look at the blob data in binary.

Method 2. Use the Search Term sefwjdfjogp/tpguxbsf.

Method 3. Use the ENCASE GREP efwjdfjogp/tpguxbsf.{100,100}

7. Build Number

Location: \data\local\storage\permanent\chrome\idb\2588645841ssegtnti.sqlite

Decoded: efwjdfjogp/cvjme = deviceinfo.build

Method 1. Open 2588645841ssegtnti.sqlite (settings) in a SQLite Browser, Open object_data table, filter key value efwjdfjogp/cvjme . Look at the blob data in binary.

Method 2. Use the Search Term sefwjdfjogp/tpguxbsf.

Method 3. Use the ENCASE GREP efwjdfjogp/cvjme.{111,111}

8. Device Model Number
   Location: \data\local\storage\permanent\chrome\idb\2588645841ssegtnti.sqlite
   Decoded: efwjdfjogp/qspevdu`npefm = deviceinfo.product_model

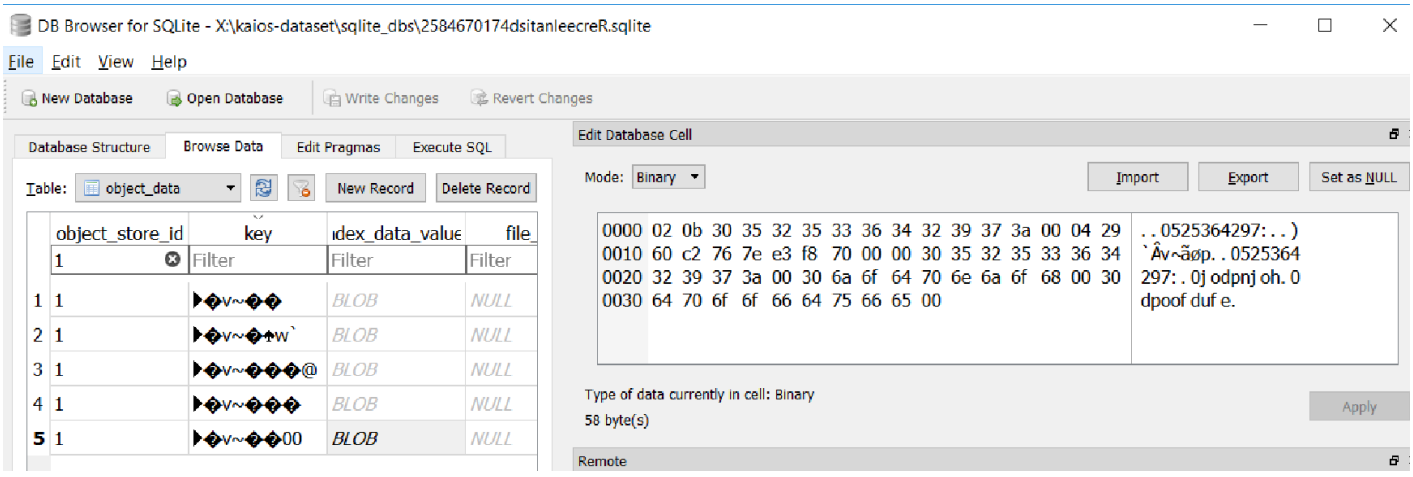
   Method 1. Open 2588645841ssegtnti.sqlite (settings) in a SQLite Browser, Open object_data table, filter key value efwjdfjogp/qspevdu`npefm . Look at the blob data in binary.

   Method 2. Use the Search Term efwjdfjogp/qspevdu`npefm.

   Method 3. Use the ENCASE GREP efwjdfjogp/qspevdu`npefm.{102,102}

CALLS:
You can see the Call History in an SQLite Browser. You will be able to determine if the calls were incoming, incoming connected or outgoing. You should be able to arrange them in chronological order too. I can also point you what I believe is the Time/Date stamp of each individual call. What I don't know is how to parse that Time/Date Stamp.
Location: data\local\storage\permanent\chrome\idb\2584670174dsitanleecreR.sqlite
AKA: dialerRecents.sqlite

(THIS PROCEDURE IS STILL IN BETA TESTING) Open dialerRecents.sqlite in SQLite browser and select object_data table, filter object_store_id to "1" and Sort on Key Column (down caret). Record 1 will have the newest Call and the Last Record (#5 in our example) is the oldest recent call. If review the BLOB you can see the telephone number of the other phone (in ROT-1) and if the incoming, incoming connected, dialing.




The ROT-1 of the above Record 5 BLOB

| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000 | 01 | 0A | 2F | 34 | 31 | 34 | 32 | 35 | 33 | 31 | 38 | 36 | 39 | FF | 03 | 28 | /4142531869ÿ ( |
| 00000016 | 5F | C1 | 75 | 7D | E2 | F7 | 6F | F5 | FF | 2F | 34 | 31 | 34 | 32 | 35 | 33 | _Áu}â÷oõÿ/414253 |
| 00000032 | 31 | 38 | 36 | 39 | FF | 2F | 69 | 6E | 63 | 6F | 6D | 69 | 6E | 67 | FF | 2F | 1869ÿ/incomingÿ/ |
| 00000048 | 63 | 6F | 6E | 6E | 65 | 63 | 74 | 65 | 64 | FF | | | | | | | connectedÿ |

The above blue highlighted portion probably contains a time/date stamp.

CALLS DIRECTION (ROT-1)

| | |
|---|---|
| ejbmjoh | dialing |
| jodpnjoh | incoming |
| dpoofdufe | connected |

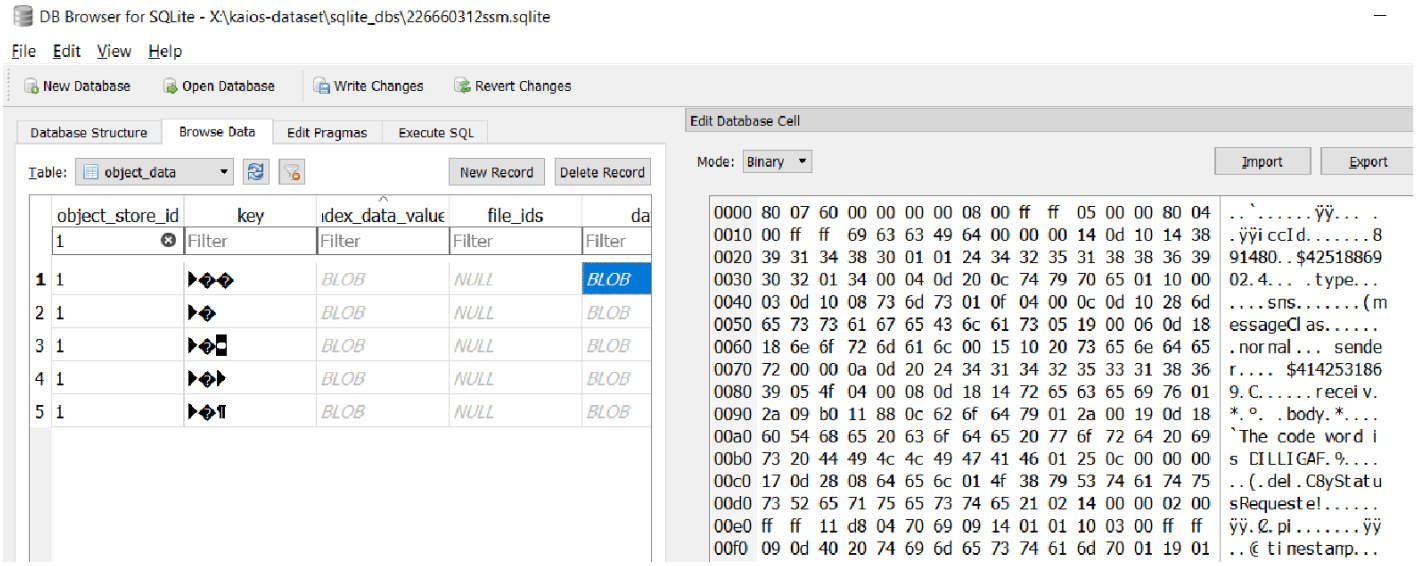
**SMS – Simple Message System:**

You should be able to read every text message stored on the handset. You will also be able to arrange them in order received.

Location: Location: data\local\storage\permanent\chrome\idb\226660312ssm.sqlite

Decode ssm = sms

(*THIS PROCEDURE IS STILL IN BETA TESTING) Open 226660312ssm.sqlite in SQLite browser and select object_data table, filter object_store_id to "1" and Sort on Index_data_value column (^). Review the Data Column BLOBs to see the SMS messages in Binary. Record 1 will have the oldest SMS and the Last Record (#5 in our example) is the most recent SMS. If review the BLOB you can see the body of the message and the other telephone number (sender/receiver) in plain text.

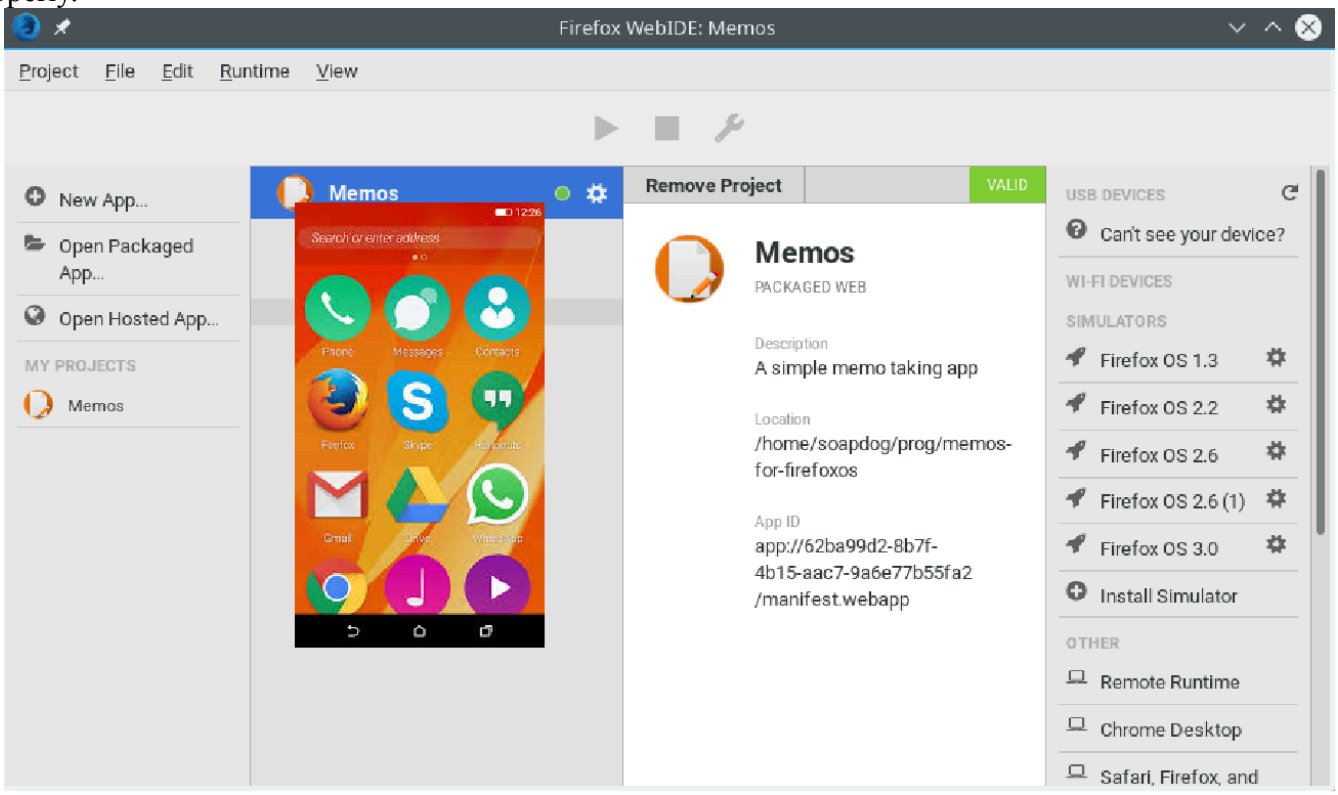



**WebAppStore.SQLite and WebAppStore.sqlite-wal**

There can be many artifacts of CALLS, SMS, CONTACTS and Internet History in these two database files. The information is stored as plain text in HTML format. I believe these files are populated by recent user activity. For example when a user looks at an SMS messages on their phone. The SMS message displayed by the the OS is actually in the form of a webpage. The OS renders the webpage from the data of the specific SQLite storage container. That webpage is cached in WebAppStore.wal. The remnants of recent user activities can be viewed in HTML formatted text in the WebAppStore.SQLite and WebAppStore.WAL using tools like ENCASE. When I tried to open WEBAPPSTORE.WAL using a regular SQLite Browser it rested an encryption code.

**SIMULATOR:**

KaiOs was built from the Firefox OS. There is actually a programming environment that is pretty easy to install from Firefox version 37.0 . It is a simulator for Pre-KaiOs . I was never able to take a SQLite databases I extracted from a phone and put it into the browser directory of the simulator (even though they had the same SQLite db names). The JavaScript for the simulator was just a little different from the real phone. I was able to get the simulator to "accept" a KaiOs JavaScript Package it just would not work properly.




**IMFORMATION for TIME/DATE STAMP RESEARCH**

I am including the following information because we as a community need to find the Time/Date Stamps and be able to convert them. I believe the key might be in the following:

```
//********* call_log.js *****  START snippet   ****************************************
//***from system\b2g\webapps\communications.gaiamobile.org\application.zip\dialer\js\call_logs.js*******

 //todo remove in release
  generateCallLog: function cl_generateCallLog() {
    var self = this;
    var number = '8910123450';   //
    for (var i = 0; i < 6; i++) {
      var d = new Date();    // Calls Current Date/Time in typically Epoch Fashion
      d.setDate(d.getDate() - i);
      var entry = {
        date: d.getTime(),
```

```
      duration: i % 2 === 0 ? 60000 : 0,
      type: i === 0 ? 'dialing' : 'incoming',
      number: number + i,
      serviceId: 0,
      emergency: false,
      voicemail: false,
      status: i % 2 === 0 ? 'connected' : null
     };
     CallLogDBManager.add(entry, function (logEntry) {
      self.appendGroupAndSaveCache(logEntry);
      NavigationMap.reset();
     });
    }
  },
//********* call_log.js *****  END snippet   **************************

//********* l10n_date.js *****  START **************************
// system\b2g\webapps\notes.gaiamobile.org\application.zip\shared\js\l10n_date.js

navigator.mozL10n.DateTimeFormat=function(){
function e(t,n){for(var o=n.match(/(%[E|O|-]?.)/g),a=0;o&&a<o.length;a++){
          var
s="";switch(o[a]){case"%a":s=i("weekday-"+t.getDay()+"-short");break;case"%A":s=i("weekday-"+t.getDay
()+"-long");break;case"%b":case"%h":s=i("month-"+t.getMonth()+"-short");break;case"%B":s=i("month-"+t.
getMonth()+"-long");break;case"%Eb":s=i("month-"+t.getMonth()+"-genitive");break;case"%-m":s=t.getMo
nth()+1;break;case"%I":s=t.getHours()%12||12;break;case"%e":s=t.getDate();break;case"%p":s=t.getHours()
<12?i("time_am"):i("time_pm");break;case"%c":case"%x":case"%X":var
r=i("dateTimeFormat_"+o[a]);r&&!/(%c|%x|%X)/.test(r)&&(s=e(t,r))}n=n.replace(o[a],s||t.toLocaleFormat(o
[a]))}return n}function t(e){e=Math.abs(e);var
t={},n=["years",31536e3,"months",2592e3,"weeks",604800,"days",86400,"hours",3600,"minutes",60];if(60>
e)return{minutes:Math.round(e/60)};for(var i=0,o=n.length;o>i;i+=2){var
a=n[i+1];e>=a&&(t[n[i]]=Math.floor(e/a),e-=t[n[i]]*a)}return t}function
n(n,o,a){switch(a=a||172800,n.constructor){case String:n=parseInt(n);break;case Date:n=n.getTime()}var
s=Date.now(),r=(s-n)/1e3;if(isNaN(r))return
i("incorrectDate");Math.abs(r)>60&&(r=r>0?Math.ceil(r):Math.floor(r));var c=new
Date;c.setHours(0,0,0,0);var d=c.getTime(),u=d-864e5;const l=new
Date(c.getFullYear().toString()).getTime(),f=navigator.mozHour12?"%I:%M %p":"%H:%M";if(l>n)return
e(new Date(n),"%B %e, %Y "+f);if(u>n)return e(new Date(n),"%B %e, "+f);if(d>n)return
i("days-ago-long",{value:1})+", "+e(new Date(n),f);if(r>14400)return i("days-ago-long",{value:0})+",
"+e(new Date(n),f);var m=o?"-short":"-long",h=t(r),p=r>=0?"-ago":"-until";for(var g in h)return
i(g+p+m,{value:h[g]})}var i=navigator.mozL10n.get;return{localeDateString:function(t){return
e(t,"%x")},localeTimeString:function(t){return e(t,"%X")},localeString:function(t){return
e(t,"%c")},localeFormat:e,fromNow:n,relativeParts:t}
};
//********* l10n_date.js *****  END **************************

MozL10n.DataTimeFormat.
```

Contact me at forensiczone@gmail.com