

**Emulating a Windows Operating System  
on an Android Mobile Device**

**Stephen Budd  
George Mason University  
November 29, 2018**

## Concept Overview

---

The processing power and capabilities of today's mobile devices are more powerful than ever. Anecdotally, it seems mobile devices are becoming the device of choice for the average user's primary computing needs, significantly more so than the traditional desktop computers. So with all this computing capability at our fingertips, what *unconventional* things can we do with it? It is this very question that fueled my curiosity for this research project. My hypothesis is as follows:

*Emulating a different operating system, specifically Windows XP, on an Android device will likely yield artifacts of interest.*

## Testing

---

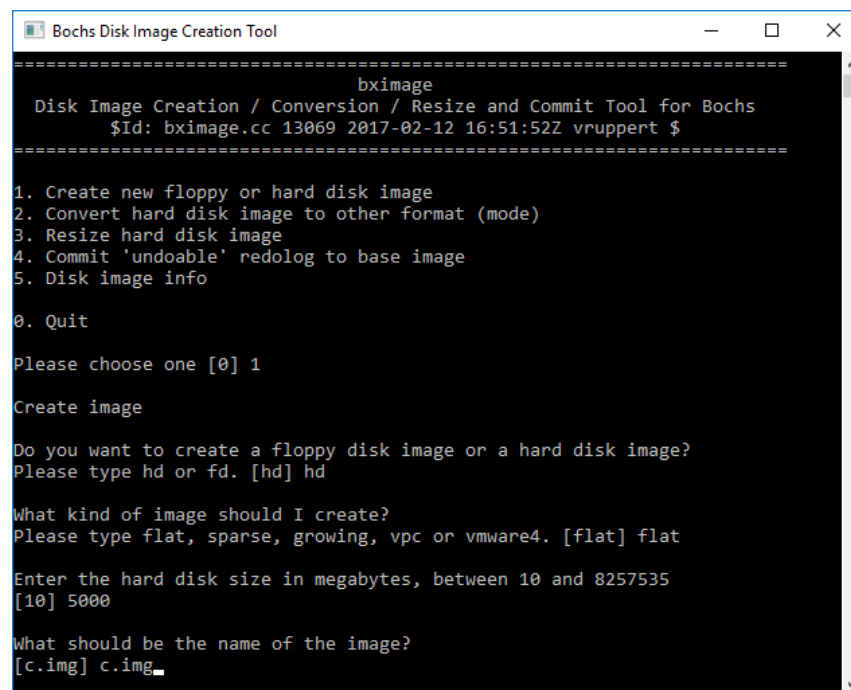
The following narrative details the steps I performed as part of this research project.

1. **Create a Blank Image File:** Before an emulation application was selected, I first needed to create a blank image file (i.e. \*.img) as a landing point for the Windows XP installation.

*Note: Windows XP was selected because its Minimum Hardware Requirements were less demanding than any newer version of Windows, and all with full acknowledgment that XP is no longer supported by Microsoft.*

To accomplish this, I used a free open source tool called Bochs Disk Image Creation Tool. Figure 1 below shows this command line interface (CLI) and the associated options which I selected for the blank image file. Note the filename as *c.img* as this will be referenced throughout this write-up. The default directory for the newly created image file is:

C:\Users\%Username%\AppData\Local\VirtualStore\Program Files (x86)\Bochs-2.6.8.

A screenshot of a Windows command prompt window titled "Bochs Disk Image Creation Tool". The window contains the following text:

```
=====
                        bximage
Disk Image Creation / Conversion / Resize and Commit Tool for Bochs
$Id: bximage.cc 13069 2017-02-12 16:51:52Z vruppert $
=====

1. Create new floppy or hard disk image
2. Convert hard disk image to other format (mode)
3. Resize hard disk image
4. Commit 'undoable' redolog to base image
5. Disk image info

0. Quit

Please choose one [0] 1

Create image

Do you want to create a floppy disk image or a hard disk image?
Please type hd or fd. [hd] hd

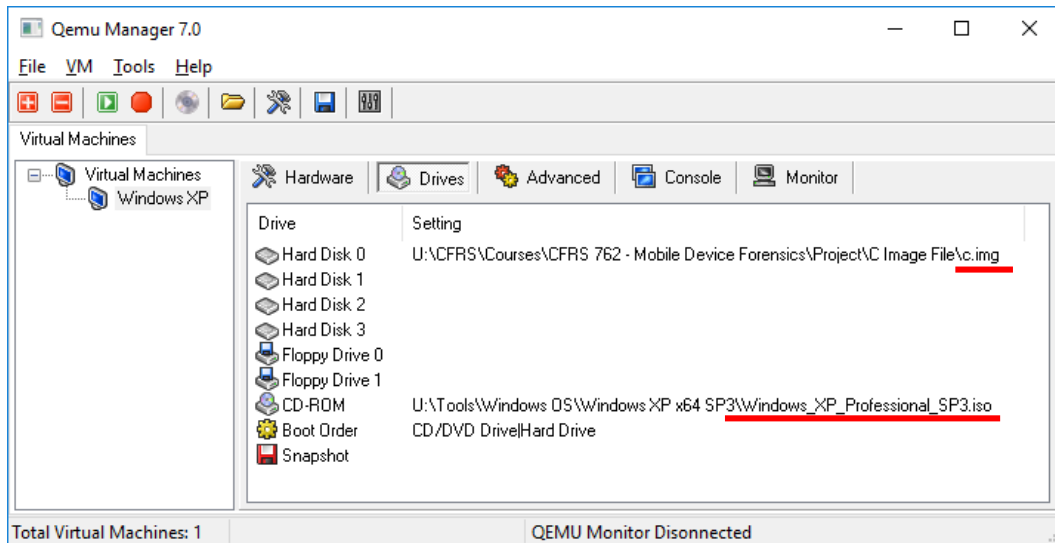
What kind of image should I create?
Please type flat, sparse, growing, vpc or vmware4. [flat] flat

Enter the hard disk size in megabytes, between 10 and 8257535
[10] 5000

What should be the name of the image?
[c.img] c.img_
```

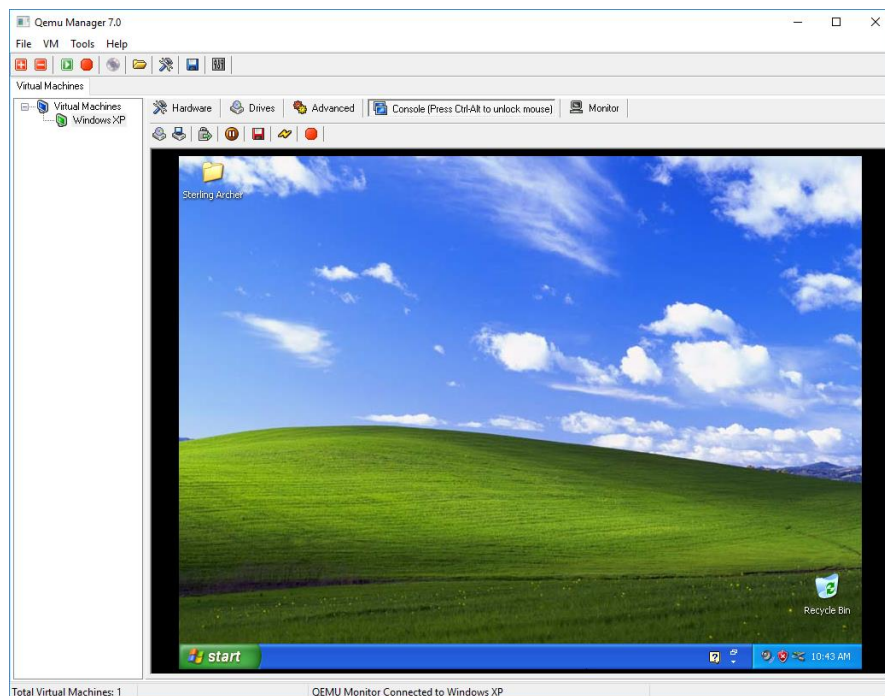
**Figure 1** - Bochs Disk Image Creation Tool

2. **Windows XP Installation:** A free Windows-based application called Qemu Manager v7.0 was used to install Windows XP on the blank *c.img* image file. Qemu Manager is a generic and open source machine emulator and virtualizer. Figure 2 below shows the drive configuration for this installation. Note the *c.img* (blank disk) image file and the Windows XP ISO files similar to creating a virtual machine using Virtual Box or VMWare.



**Figure 2** - Qemu Manager drive configuration for Windows XP installation

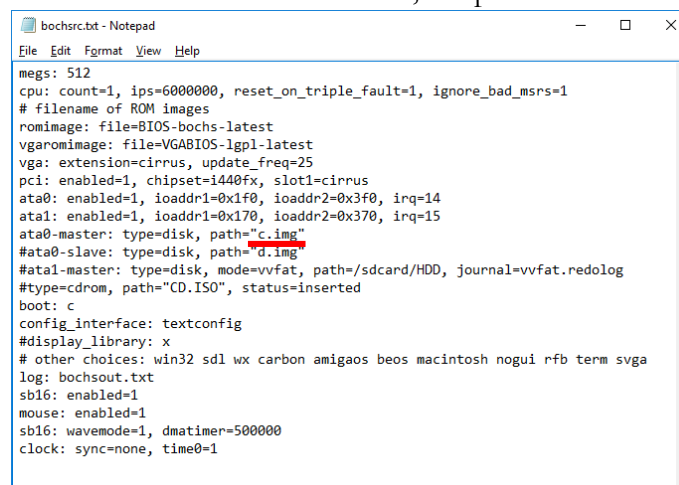
3. **Test Windows XP Installation:** Next, the XP operating system was tested in Qemu Manager to validate a successful installation. The username *Sterling Archer* was set up with Administrator privileges. Figure 3 below is a screenshot of XP running in Qemu Manager.



**Figure 3** – XP running in Qemu Manager

4. **Exemplary Devices:** Since I had to personally finance this effort, my “Research & Development” budget was rather limited. Therefore, top of line phones with the highest processing power were not an option. However, with some deliberate comparison of specification sheets and reading dozens of product reviews I ended up selecting a BLU Vivo XI+ running Android Oreo (8.1.0) and a rooted Google Pixel (1<sup>st</sup> generation) also running Android Oreo (8.1.0) as my exemplary devices.
5. **Simple Direct-Media Layer (SDL):** According to the SDL website “Simple Direct-Media Layer (SDL for short) is a cross-platform development library designed to provide low-level access to audio, keyboard, mouse, joystick, and graphics hardware via OpenGL and Direct3D. Developed by Sam Lantinga, SDL is used by video playback software, emulators, and popular games.”<sup>1</sup> When paired with the Bochs Cross Platform Emulator Android application the result is a simple yet powerful combination for running non-standard operating systems on Android devices.
6. **Bochs Cross Platform IA-32 Emulator Android App:** According to the Bochs website and the Google Play Store, the Bochs application, originally developed by Kevin Lawton, “is a highly portable open source x86 PC emulator written in C++, that runs on most popular platforms. It includes emulation of the Intel x86 CPU, common I/O devices, and a custom BIOS. Bochs can be compiled to emulate many different x86 CPUs, from early 386 to the most recent x86-64 Intel and AMD processors. Bochs is capable of running most Operating Systems inside the emulation including Linux, DOS or Microsoft Windows. The typical use of Bochs is to provide complete x86 PC emulation, including the x86 processor, hardware devices, and memory thereby allowing users to run operating systems and software within the emulator on their workstation [or device].”<sup>2,3</sup>

There are three files that must accompany the Bochs Android application to ensure successful emulation. The files are *BIOS-bochs-latest*, *bochsrc.txt*, and *VGABIOS-lgpl-latest*. Additionally, the stock *bochsrc.txt* required a slight modification prior to implementation.<sup>4</sup> Specifically the *ata0 master path* (reference Figure 4 below) needed to match the name of the image file used to install Windows XP. In this case, the pathname was changed to *c.img*.

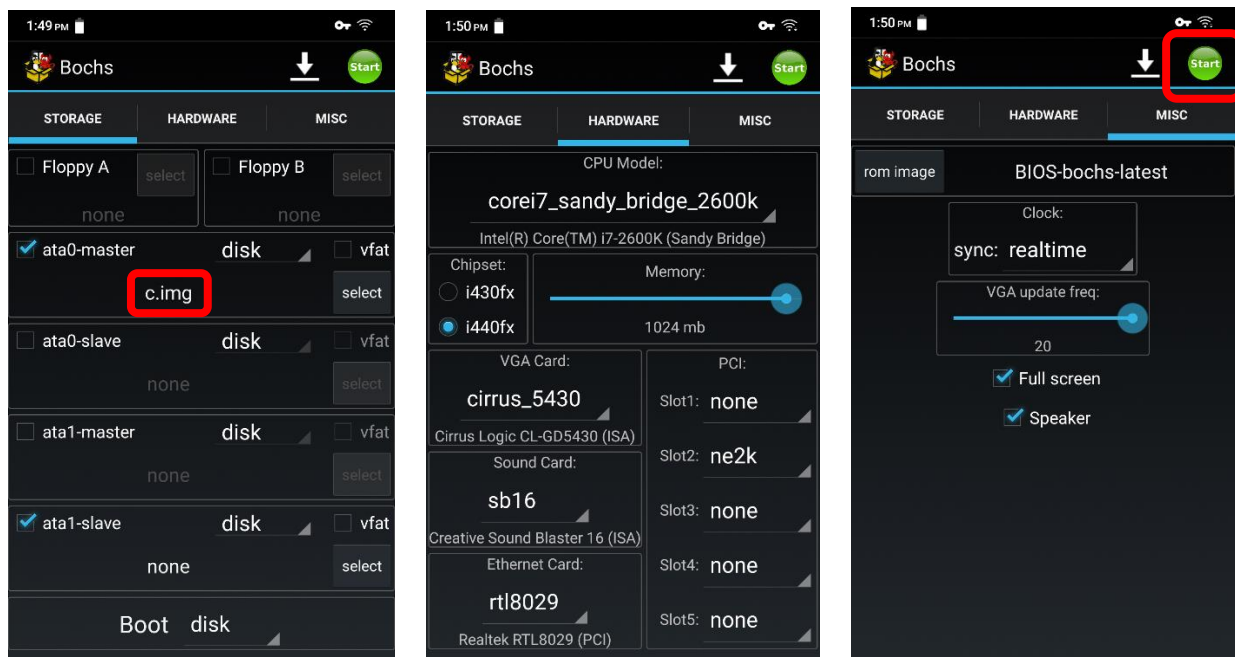


```
bochsrc.txt - Notepad
File Edit Format View Help

megs: 512
cpu: count=1, ips=6000000, reset_on_triple_fault=1, ignore_bad_msrs=1
# filename of ROM images
romimage: file=BIOS-bochs-latest
vgaromimage: file=VGABIOS-lgpl-latest
vga: extension=cirrus, update_freq=25
pci: enabled=1, chipset=i440fx, slot1=cirrus
ata0: enabled=1, ioaddr1=0x1f0, ioaddr2=0x3f0, irq=14
ata1: enabled=1, ioaddr1=0x170, ioaddr2=0x370, irq=15
ata0-master: type=disk, path="c.img"
#ata0-slave: type=disk, path="d.img"
#ata1-master: type=disk, mode=vfat, path=/sdcard/HDD, journal=vfat.redolog
#type=cdrom, path="CD.ISO", status=inserted
boot: c
config_interface: textconfig
#display_library: x
# other choices: win32 sdl wx carbon amigaos beos macintosh nogui rfb term svga
log: bochsout.txt
sb16: enabled=1
mouse: enabled=1
sb16: wavemode=1, dmatimer=500000
clock: sync=none, time0=1
```

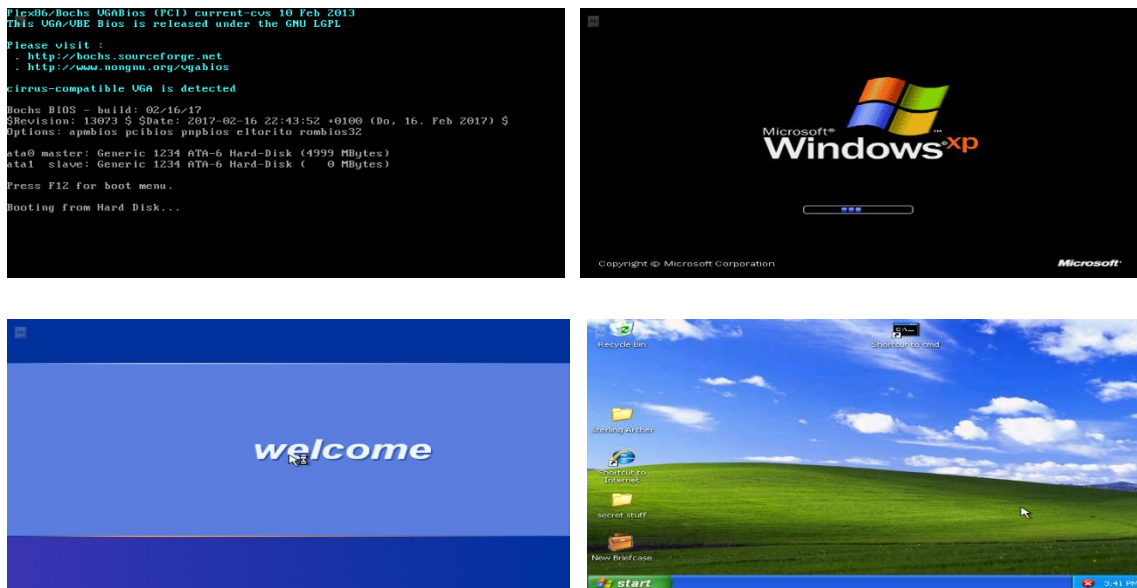
**Figure 4** – modified *bochsrc.txt*

7. **Transfer Files to Exemplary Devices:** Hine sight, I should have created the *c.img* file as a 4GB blank disk instead of 5GB so that a FAT32 formatted USB flash drive could be used to quickly transfer the three Bochs files and the *c.img* file to both exemplary devices. Apparently, neither exemplary device supports exFAT formatted USB drives even with an On-The-Go (OTG) USB Type C / Type A cable adapter. As a result, the files were simply emailed to an address specifically created for this research project, and then subsequently downloaded onto each device. All four files were then saved to the root directory of each exemplar's internal storage, specifically in a folder titled *SDL*.
8. **Emulation:** With the Bochs emulator Android app installed along with the necessary files (*BIOS-bochs-latest*, *bochsrc.txt*, and *VGABIOS-lgpl-latest*) in place on both exemplary devices it was time to configure the app to enable a successful emulation. Figure 5 below shows the settings which were used to configure the emulation. Note the *c.img* as the *ata0-master*.



**Figure 5** – Bochs Android App Settings

Tapping the *Start* button initiates the boot sequence. Figure 6, on the next page, shows a series of screenshots during the Windows XP bootup process on the BLU Vivo XI+ and are followed by Figure 7 which is a photo of the same exemplary device with XP up and running. There was an audible sigh of relief when the operating system finally loaded, despite being the outdated Windows XP.



**Figure 6** – Screenshots of Windows XP booting up on the BLU Vivo XI+ exemplary device



**Figure 7** – Photograph of Windows XP running on the BLU Vivo Xi+ exemplary device

9. **Create User Footprint:** At this stage, it is worth noting that “using” XP on both exemplary devices was painfully slow which was disappointing but not overly surprising. Moreover, navigating the GUI with the touchscreen mouse and virtual keyboard was very cumbersome. Additionally, the installation was unable to connect to either exemplar’s Wireless Network Interface Card which kept me from generating any internet artifacts.
10. **Acquisition:** First off, thanks go out to Professor Jessica Hyde for providing a trial license of Magnet Forensics AXIOM tool suite. Full disclosure, I have no previous training (on-the-job, instructor-led, or otherwise) with AXIOM. Therefore, the issues encountered during the acquisition and analysis portions of this research project are attributable to operator error vice a capability shortfall with the tool. That being said, AXIOM made short



order of a quick/logical acquisition of the BLU Vivo XI+ exemplary. A second full/physical was attempted on the same exemplary, but was unsuccessful due to the device not being rooted. Booting the Google Pixel into the Team Win Recovery Project custom recovery enabled AXIOM to obtain a full/physical acquisition. In an effort to streamline the analysis portion of this research project the National Software Reference Library (NSRL) Android Reference Data Set (RDS) from the National Institute of Standards and Technology's website was incorporated for both acquisitions.<sup>5</sup>

## Targeted Analysis

### ➤ Target #1 - Analyzing the *c.img* file:

In no particular order, the following descriptions represent general findings from analyzing the *c.img* file. Every tool employed for this portion of the analysis is, as of this writing, open source. Additionally, all parsing and analysis were performed on a forensic workstation. This section is far from an exhaustive analysis for the simple fact that analyzing *.img* files is synonymous with performing traditional computer forensics on other image file formats (e.g. .E01, .dd, .raw), and therefore outside the scope of this research project. As such, only strings, registry, and prefetch data parsing has been detailed below.

1. **Strings Parsing:** A command line utility developed by Eric Zimmerman named *bstrings.exe*<sup>6</sup>, was run against the *c.img* file using the following syntax:

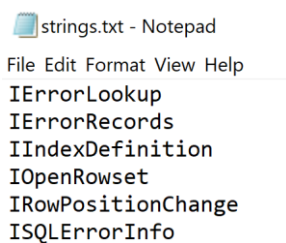
*bstrings.exe -m 8 -b 1024 -o C:\strings.txt -s -f <path to c.img file>*

Figure 8 below provides a brief description of each switch employed in the aforementioned command line.

<i>-m 8</i>	Sets the minimum string length to eight characters. The default is three. Eight was used simply to speed up the string parsing process.
<i>-b 1024</i>	Chunks <i>c.img</i> in blocks of 1024MB which resulted in five chunks for the 5GB <i>c.img</i> file
<i>-o</i>	Writes output of the search to the defined <i>strings.txt</i> file.
<i>-s</i>	Really Quiet Mode (Does not display hits to console. Speeds up processing when using the <i>-o</i> switch)
<i>-f</i>	Specifies the file to search

**Figure 8** - Description of each switch employed

Figure 9 below is a very small and arbitrary screenshot from the *strings.txt* output file.



```

strings.txt - Notepad
File Edit Format View Help
IErrorLookup
IErrorRecords
IIndexDefinition
IOpenRowset
IRowPositionChange
ISQLErrorInfo

```

**Figure 9** – Screenshot of the *strings.txt* output file

2. **Registry Parsing:** To parse the Windows XP registry hives I used RegRipper, an open source tool developed by Harlan Carvey. Figure 10 below shows two screenshots from the report generated after parsing the NTUSER registry hive. Note the *Sterling Archer* fictitious username.

```
-----
logonusername v.20080324
(NTUSER.DAT) Get user's Logon User Name value

Logon User Name
Software\Microsoft\Windows\CurrentVersion\Explorer
LastWrite Time [Sat Oct 6 04:58:58 2018 (UTC)]
Logon User Name = Sterling Archer
-----

shellfolders v.20090115
(NTUSER.DAT) Retrieve user Shell Folders values

Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
LastWrite Time Sat Oct 6 04:57:50 2018 (UTC)
Personal          C:\Documents and Settings\Sterling Archer\My Documents
Desktop           C:\Documents and Settings\Sterling Archer\Desktop
AppData           C:\Documents and Settings\Sterling Archer\Application Data
Cookies           C:\Documents and Settings\Sterling Archer\Cookies
Favorites         C:\Documents and Settings\Sterling Archer\Favorites
NetHood           C:\Documents and Settings\Sterling Archer\NetHood
PrintHood         C:\Documents and Settings\Sterling Archer\PrintHood
Recent            C:\Documents and Settings\Sterling Archer\Recent
SendTo            C:\Documents and Settings\Sterling Archer\SendTo
Start Menu        C:\Documents and Settings\Sterling Archer\Start Menu
Templates         C:\Documents and Settings\Sterling Archer\Templates
Programs          C:\Documents and Settings\Sterling Archer\Start Menu\Programs
Startup           C:\Documents and Settings\Sterling Archer\Start Menu\Programs\Startup
Local Settings    C:\Documents and Settings\Sterling Archer\Local Settings
Local AppData     C:\Documents and Settings\Sterling Archer\Local Settings\Application Data
Cache             C:\Documents and Settings\Sterling Archer\Local Settings\Temporary Internet Files
History           C:\Documents and Settings\Sterling Archer\Local Settings\History
My Pictures        C:\Documents and Settings\Sterling Archer\My Documents\My Pictures
Fonts             C:\WINDOWS\Fonts
My Music          C:\Documents and Settings\Sterling Archer\My Documents\My Music
My Video
Administrative Tools
CD Burning        C:\Documents and Settings\Sterling Archer\Local Settings\Application Data\Microsoft\CD Burning
-----
```

**Figure 10** – Screenshot *NTUSER.dat* RegRipper report

3. **Prefetch Parsing:** To parse the *c.img's* *C:\Windows\Prefetch* directory I used WinPrefetchView v1.35 developed by Nir Soft. The screenshot below shows the results of the prefetch parsing which were filtered from highest to lowest Run Count.

Filename	Created Time	Modified Time	File Size	Process EXE	Process Path	Run Counter	Last Run Time	Missing Process
WUAUCLT.EXE-399A8E72.pf	10/6/2018 12:52:36 AM	10/6/2018 12:57:58 AM	17,860	WUAUCLT.EXE	\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\WUAUCLT.EXE	10	10/6/2018 12:57:48 AM	No
VERCLSID.EXE-3667BD89.pf	10/6/2018 12:55:31 AM	10/6/2018 12:58:08 AM	21,924	VERCLSID.EXE	\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\VERCLSID.EXE	7	10/6/2018 12:58:06 AM	No
CMD.EXE-087B4001.pf	10/6/2018 12:55:20 AM	10/6/2018 12:55:27 AM	12,716	CMD.EXE	\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\CMD.EXE	5	10/6/2018 12:55:25 AM	No
DEVCON.EXE-3189D4D7.pf	10/6/2018 12:55:23 AM	10/6/2018 12:55:26 AM	7,940	DEVCON.EXE	\DEVICE\HARDDISKVOLUME1\DEVCON.EXE	4	10/6/2018 12:55:26 AM	No
REGSVR32.EXE-25EFE2F2.pf	10/6/2018 12:56:12 AM	10/6/2018 12:57:25 AM	35,502	REGSVR32.EXE	\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\REGSVR32.EXE	3	10/6/2018 12:57:15 AM	No
SHMGRATE.EXE-1BA69E68.pf	10/6/2018 12:56:21 AM	10/6/2018 12:57:15 AM	22,450	SHMGRATE.EXE	\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\SHMGRATE.EXE	3	10/6/2018 12:57:13 AM	No
SVCHOST.EXE-3530F672.pf	10/6/2018 12:52:22 AM	10/6/2018 12:52:30 AM	20,560	SVCHOST.EXE	\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\SVCHOST.EXE	3	10/6/2018 12:52:20 AM	No
IE4UINIT.EXE-169A5A39.pf	10/6/2018 12:56:03 AM	10/6/2018 12:57:07 AM	38,930	IE4UINIT.EXE	\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\IE4UINIT.EXE	2	10/6/2018 12:57:05 AM	No
LOGONUI.EXE-0AF22957.pf	10/6/2018 12:55:15 AM	10/6/2018 12:59:05 AM	19,108	LOGONUI.EXE	\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\LOGONUI.EXE	2	10/6/2018 12:58:55 AM	No
SETUP50.EXE-362FF7C9.pf	10/6/2018 12:56:16 AM	10/6/2018 12:56:28 AM	57,692	SETUP50.EXE	\DEVICE\HARDDISKVOLUME1\PROGRAM FILES\OUTLOOK EXPRESS\SETUP50.EXE	2	10/6/2018 12:56:25 AM	No
WMIADAP.EXE-2DF42582.pf	10/6/2018 12:54:14 AM	10/6/2018 12:56:47 AM	21,936	WMIADAP.EXE	\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\WBEM\WMIADAP.EXE	2	10/6/2018 12:56:33 AM	No
WMIPRVSE.EXE-28F301A9.pf	10/6/2018 12:53:33 AM	10/6/2018 12:53:38 AM	22,554	WMIPRVSE.EXE	\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\WBEM\WMIPRVSE.EXE	2	10/6/2018 12:53:28 AM	No
WSCNTFY.EXE-1824F5EB.pf	10/6/2018 12:55:25 AM	10/6/2018 12:58:58 AM	9,488	WSCNTFY.EXE	\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\WSCNTFY.EXE	2	10/6/2018 12:58:58 AM	No
AGENTSVR.EXE-002E45A8.pf	10/6/2018 12:52:57 AM	10/6/2018 12:52:57 AM	5,628	AGENTSVR.EXE	\DEVICE\HARDDISKVOLUME1\WINDOWS\MSAGENT\AGENTSVR.EXE	1	10/6/2018 12:52:47 AM	No
ALG.EXE-0F138680.pf	10/6/2018 12:53:32 AM	10/6/2018 12:53:32 AM	17,406	ALG.EXE	\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\ALG.EXE	1	10/6/2018 12:53:22 AM	No
CSRSS.EXE-12863473.pf	10/6/2018 12:55:31 AM	10/6/2018 12:55:31 AM	1,890			1	10/6/2018 12:55:11 AM	No
DPSFNHSHR.EXE-1C8143D1.pf	10/6/2018 12:55:30 AM	10/6/2018 12:55:30 AM	18,544	DPSFNHSHR.EXE	\DEVICE\HARDDISKVOLUME1\DPSFNHSHR.EXE	1	10/6/2018 12:55:20 AM	No
EXPLORER.EXE-082F38A9.pf	10/6/2018 12:55:25 AM	10/6/2018 12:55:25 AM	29,172	EXPLORER.EXE	\DEVICE\HARDDISKVOLUME1\WINDOWS\EXPLORER.EXE	1	10/6/2018 12:55:15 AM	No

**Figure 11** – Screenshot of the *c.img's* *C:\Windows\Prefetch* directory



**Figure 13** – Unreadable file structure for `/data/data` from the Google Pixel

2. **XML Files of Interest:** Using Magnet Forensics AXIOM, four XML files of interest were found in the *net.sourceforge.bochs/sp* directory and are shown in Figure 14 below.

#### EVIDENCE (4)

	Name	Type	File ext...	Size (bytes)	Created
	admob.xml	File	.xml	807	
	net.sourceforge.bochs_preferences.xml	File	.xml	136	
	MainActivity.xml	File	.xml	133	
	WebViewChromiumPrefs.xml	File	.xml	127	

**Figure 14** – XML Files of Interest from the *net.sourceforge.bochs/sp* directory

- a. **admob.xml:** A review of the *admob.xml* file resulted in three artifacts of interest which have been marked by the red arrows and boxes in Figure 15 below.

```

1 <?xml version='1.0' encoding='utf-8' standalone='yes' ?>
2 <map>
3   <string name="app_settings_json">{"status":1,"app_id":
   &quot;ca-app-pub-4572241703388342~5758790772&quot;,"auto_collect_location
   &quot;:true,"ad_unit_id_settings":{"ad_unit_id":
   &quot;ca-app-pub-4572241703388342/3371412534&quot;,"format":
   &quot;banner&quot;},"persistable_banner_ad_unit_ids":["
   &quot;ca-app-pub-4572241703388342/3371412534&quot;"]}
4   <long name="app_settings_last_update_ms" value="1538805957909" />
5   <int name="request_in_session_count" value="-1" />
6   <long name="first_ad_req_time_ms" value="1541361732452" />
7   <boolean name="auto_collect_location" value="true" />
8   <long name="app_last_background_time_ms" value="1541364595064" />
9 </map>

```

**Figure 15** – *admob.xml* from the *net.sourceforge.bochs/sp* directory

The value strings encircled above appeared to be epoch timestamps. Testing this theory, I used an online epoch time converter<sup>7</sup> which resulted in the following human-readable dates:

Last Update: 1538805957909	Friday, November 16, 2018 3:12:16 AM GMT-05:00
First Ad Req Time: 1541361732452	Friday, November 16, 2018 6:17:17 PM GMT-05:00
App Last Background Time: 1541364595064	Friday, November 16, 2018 6:32:52 PM GMT-05:00

- b. **net.sourceforge.bochs\_preferences.xml:** Figure 16 shows the XML code for the *net.sourceforge.bochs\_preferences.xml* file.

```

1 <?xml version='1.0' encoding='utf-8' standalone='yes' ?>
2 <map>
3   <boolean name="variations_seed_native_stored" value="true" />
4 </map>

```

**Figure 16** – *net.sourceforge.bochs\_preferences.xml* from the *net.sourceforge.bochs/sp* directory

At the time of this writing, I am unable to determine the level of interest or purpose of the *variations\_seed\_native\_stored* XML code.

c. **MainActivity.xml**: Figure 17 shows the XML code for the *MainActivity.xml* file

```
1 <?xml version='1.0' encoding='utf-8' standalone='yes' ?>
2 <map>
3   <string name="saved_path">/storage/emulated/0/SDL</string>
4 </map>
```

**Figure 17** – *MainActivity.xml* from the *net.sourceforge.bochs/sp* directory

The */storage/emulated/0/SDL* represents the Android backup location for the SDL files used by the Boch Android application.

d. **WebViewChromiumPrefs.xml**: Figure 18 shows the XML code for the *WebViewChromiumPrefs.xml* file.

```
1 <?xml version='1.0' encoding='utf-8' standalone='yes' ?>
2 <map>
3   <int name="lastVersionCodeUsed" value="353808052" />
4 </map>
```

**Figure 18** – *WebViewChromiumPrefs.xml* from the *net.sourceforge.bochs/sp* directory

The *lastVersionCodeUsed* represents the current version of the Bochs Android application used on the device. The Google Play store confirms 2.6.9 is the current version, as of this writing, for the Bochs Android application installed which was installed on both exemplary devices. Unfortunately, at the time of this writing, I am unable to correlate “2.6.9” with “353808052” number string from the XML code above.

3. **Other Files of Interest**: Figure 19 on the next page shows the contents of the *leveldb* folder. Inside this folder were three log files and three other unknown file types. Viewing the unknown file types each in a hex editor (HxD v1.7.7.0) and a database viewer (DB Browser for SQLite v3.10.1) yielded little to no information of interest which was not surprising given their file sizes (i.e. 16, 41, and 0 bytes respectively).

**EVIDENCE (6)**

Selected folder only Column view

Name	Type	File extension	Size (bytes)	Created	Accessed	Modified	MD5 hash
CURRENT	File		16			10/6/2018 6:05:57 AM	46295cac801e5d4857d09837238a6394
MANIFEST-000001	File		41			10/6/2018 6:05:57 AM	5af87dfd673ba2115e2fc5cfdb727ab
000003.log	File	.log	0			10/6/2018 6:05:57 AM	d41d8cd98f00b204e9800998ecf8427e
LOG	File		182			10/27/2018 3:04:26 AM	ac768e8bdabf20908691682f7613f3e1
LOG.old	File	.old	182			10/27/2018 2:49:48 AM	71f4e41635403b4d949473b1cf707265
LOCK	File		0			10/6/2018 6:05:57 AM	d41d8cd98f00b204e9800998ecf8427e

**Figure 19** – Files of Interest from the *net.sourceforge.bochs/r/app\_webview/Local Storage/leveldb* directory

For completeness, Figure 20 below shows the content of the *LOG.old* file primarily because when compared to the other files it held the most content, all 182 bytes in total.

LOG.old

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 32 30 31 38 2F 31 31 2F 30 34 2D 31 34 3A 34 36 2018/11/04-14:46
00000010 3A 32 35 2E 38 35 30 20 33 30 33 35 31 20 52 65 :25.850 30351 Re
00000020 75 73 69 6E 67 20 4D 41 4E 49 46 45 53 54 20 6C using MANIFEST 1
00000030 65 76 65 6C 64 62 2F 4D 41 4E 49 46 45 53 54 2D evelldb/MANIFEST-
00000040 30 30 30 30 30 31 0A 32 30 31 38 2F 31 31 2F 30 000001.2018/11/0
00000050 34 2D 31 34 3A 34 36 3A 32 35 2E 38 35 31 20 33 4-14:46:25.851 3
00000060 30 33 35 31 20 52 65 63 6F 76 65 72 69 6E 67 20 0351 Recovering
00000070 6C 6F 67 20 23 33 0A 32 30 31 38 2F 31 31 2F 30 log #3.2018/11/0
00000080 34 2D 31 34 3A 34 36 3A 32 35 2E 38 35 32 20 33 4-14:46:25.852 3
00000090 30 33 35 31 20 52 65 75 73 69 6E 67 20 6F 6C 64 0351 Reusing old
000000A0 20 6C 6F 67 20 6C 65 76 65 6C 64 62 2F 30 30 30 log leveldb/000
000000B0 30 30 33 2E 6C 6F 67 20 0A 003.log .0
```

**Figure 20** – Contents of the *LOG.old* file

## Conclusion

This research project, while not terribly enlightening, did satisfy my curiosity about running a non-traditional operating system on an Android device. Additionally, it was helpful for me, as a novice and a student, to examine the data using a commercial-grade tool while simultaneously leaning on course lecture notes and textbooks to draw my conclusions. From my standpoint, the *admob.xml* file appeared to yield the most data of interest, and I suspect this specific information would be helpful for the examiner as they develop their timeline surrounding a specific chain of events.

## References

---

1. “About SDL.” Simple DirectMedia Layer. Accessed October 13, 2018.  
<https://www.libsdl.org/>
2. “Welcome to the Boch IA-32 Emulator Project.” Bochs. Accessed October 13, 2018.  
<http://bochs.sourceforge.net/>
3. “Bochs.” Google Play. Accessed on October 13, 2018.  
<https://play.google.com/store/apps/details?id=net.sourceforge.bochs&hl=en>
4. “Bochs User Manual Chapter 4. Setup.” Bochs. Accessed October 20, 2018.  
<http://bochs.sourceforge.net/doc/docbook/user/bochsrc.html>
5. “Current RDS Hash Sets, RDS Version 2.62 – September 2018, Android RDS.” National Institute of Standards and Technology. Accessed November 10, 2018.  
<https://www.nist.gov/itl/ssd/software-quality-group/nsrl-download/current-rds-hash-sets>
6. “EricZimmerman/bstrings.” GitHub. Accessed November 11, 2018.  
<https://github.com/EricZimmerman/bstrings>
7. “Epoch & Unix Timestamp Conversion Tools.” EpochConverter. Accessed November 17, 2018. <https://www.epochconverter.com/>