TBRT Plist Navigator Research Report

Project Contributors: Todd Chu, Blair Wolfinger, Ravi Jagannathan, Tom Pari

***Digital Forensics/NYU Poly  
CS6963***

***Instructor Dr. Darren R. Hayes.***

***Spring 2012***

# Introduction

We heard through Prof. Hayes from Ryan Kubasiak of appleexaminaer.com that there is a dire need for a Windows application that can read the content of the binary plist file as it is in the encoded form and quite challenging to interpret for humans. And the commercially available software is expensive.

Hence, Prof. Hayes working with New York State Police department identified the need for open source windows plist reader application as one of the options for the course project. We signed up for this project.

Our team took this seed proposal and combined it with the Digital Forensic concepts we learned in the course.

This research paper provides a report of our team’s research in implementing this project.

# Importance of Plist to Forensic Investigators

Operating Systems i.e. Linux, Windows, iOS, Mac OS X keep configuration, events and log the activities in the system in their own ways. Apple computers have become very popular both at the desktop and mobile level (Ex: iPod, Mac books laptops, Apple OS X desktops, iPhones). Apple devices keep such information in files called Property List (i.e. plist) files, in a distributed manner. Property lists organize data into named values and lists of values using several object types. These types give one the means to produce data that is meaningfully structured, transportable, storable, and accessible, but still as efficient as possible. Plists can be considered the equivalent of Windows “registry” for OS X and iOS. For this reason, it is important for forensic examiners to understand where all they can find plist files, and what information they can gleam from them. ‘plist’ files contain a treasure trove of information that can make or break a case.

Initially plist files appeared in ASCII format and later evolved into XML and binary format. Binary plists can be processed faster and stored in a compact manner. But the binary format plist files are encoded and is challenging for a human to gleam information out of it.

To illustrate the importance of content of plist file to Forensic investigators, we enlist the information found in plist files that are of interest to Forensic investigators:

* Plist files logs various system and application activities
* Devices that have been synced to the computer
* The WiFi devices the devices were connected to, includes a WiFi’s SSID, IP address used, sub netmask etc.
* CD, DVD, iPod iPhone devices connected to Mac desktop, including the media’s details such as serial number, use count, software version running on the device etc.
* Internet Browser history, cookies etc.
* Internet net chat logs
* Text and Multimedia messaging content and logs
* Contact lists and call logs on mobile device such as iPhone
* Email details

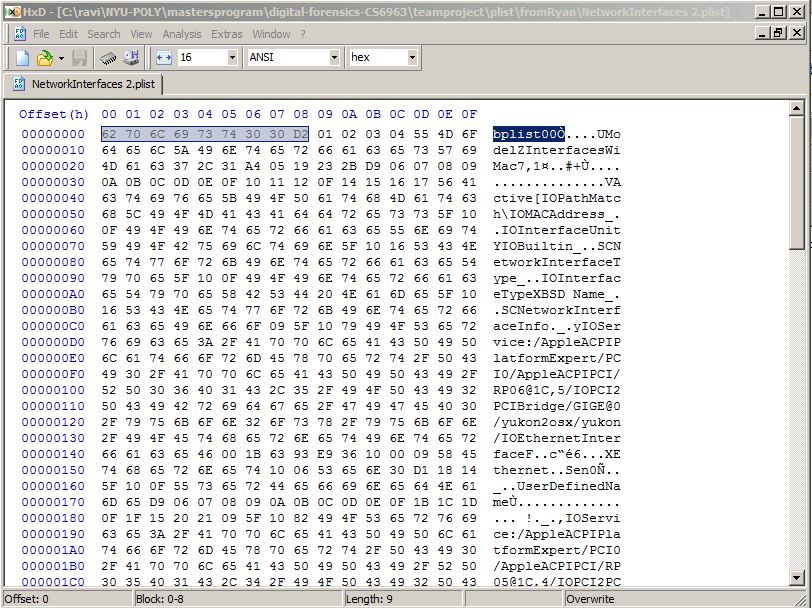
Following is the some of the plist file examples one may find on a Mac device depending on the software on the system and the device:

- /user/Library/Preferences/com.apple.loginitems.plist  
- /user/Library/Preferences/com.apple.recentitems.plist  
- /hd/Library/Preferences/SystemConfiguration/com.apple.airport.preferences.plist  
- /hd/Library/Preferences/SystemConfiguration/com.apple.network.indentification.plist  
- /user/Library/Preferences/com.apple.finder.plist  
- /user/Library/Preferences/com.apple.iPod.plist  
- History.plist  
- Downloads.plist

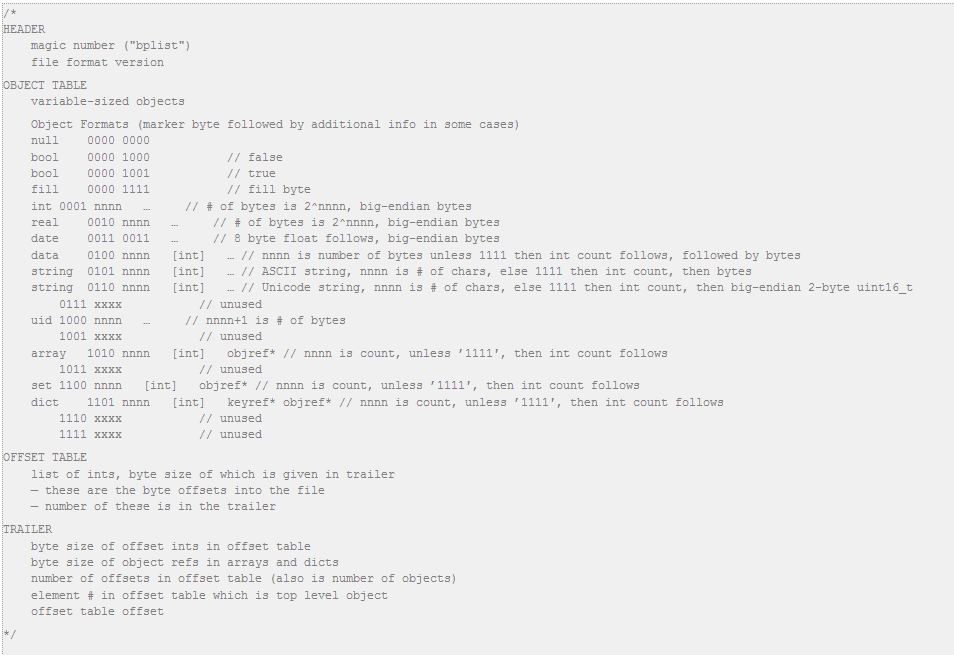
A few software is available to process the plist, particularly which is in binary format, on Apple devices; as per Ryan Kubasiak of appleexaminaer.com, only Oxygen software is available on Windows to process binary plists. But they are expensive and put the law enforcement in the bind. To alleviate Forensic Investigator from this tight corner, as per Prof. Hayes’s advice and encouragement we chose this project.

# Plist Research

For Forensic Investigators the plist file is important as it holds vital information for both desktop, laptop and mobile forensics as plist is used by OS X, iOS and neXTSTEP. Applications store configuration and run-time information in plist. To understand the internal details of the binary plist file, we examined the content of plist in hexadecimal format using “HxD Hexaeditor”. Below is the picture of “NetworkInterfaces 2.plist” file provided to us as an example plist:



As highlighted above, with web search, we recognized plist files start with the magic string “bplist” and it is coded using pre-defined encoding scheme:



Combining the information above with the corresponding XML format of the “NetworkInterfaces 2.plist” provided to us, we could identify the Key names, Key types and their values by hand. Quickly we recognized this requires development of an algorithm. We searched on the web and encountered the link <http://opensource.apple.com/source/CF/CF-476.15/CFPropertyList.c> from Apple computers the Objective C code published as open source. But this code was meant to be used on Apple machines, not for Windows. So, we faced the challenge of finding a different solution.

# Quest to find a platform independent solution

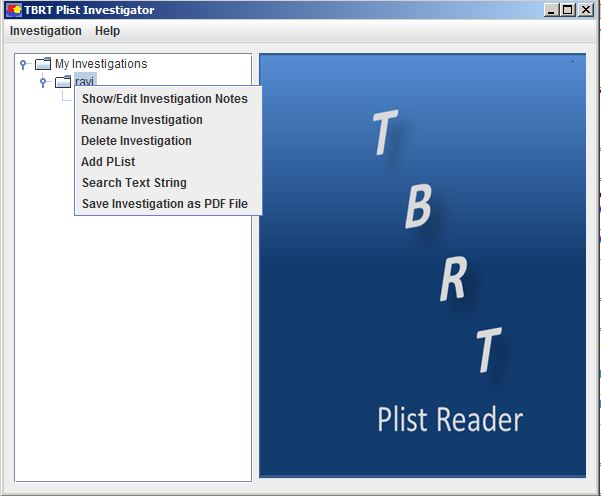
At this point we decided the best way to proceed is to base our project on Java as it is highly portable across many Operating Systems. Since plist has been in existence for a while, we thought somebody must have produced an open source parser for plist. We searched for it and found the weblink <http://code.google.com/p/plist/> had the Java code and the binary library (JAR files) needed. We quickly tested with a couple of example plist files provided to us as part of the project and found the package actually works.

# Development environment

Eclipse IDE is the most popular IDE (Integrated Development Environment) for Java and from the start we knew that’s the IDE application we would use for development. We debated whether to host our source code control in house or as a Google project. As we ran into some difficulties attempting to host it internally for a geographically dispersed team, we ended up choosing Google-code based development, which comes with SVN source code control system. Also, Eclipse has SVN plugins to integrate the source code control with the code development in a convenient manner. We tested it and found out it worked well for our purpose. On top of it, this way we can share the fruit of our labor with others. We setup the Google-code environment as read/write for four of us and as read-only for others.

# How to make it useful to a Digital Forensic Investigator?

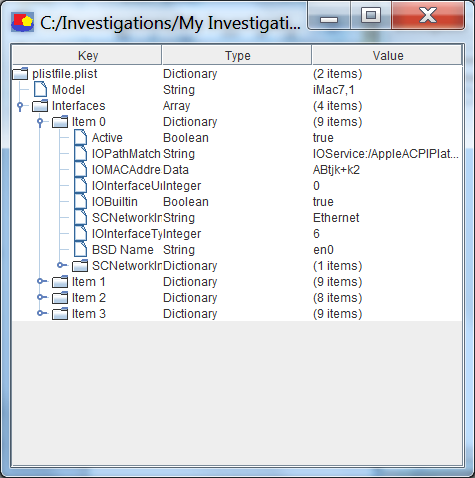
Often people develop application(s) for a class exercise that may not be suitable for using it in practice. To make our efforts useful to the Forensic community, we looked into how we can help an investigator in organizing plist. We decided to put in practice what we learned in the class. We came up with the concept of treating each plist file as evidence; and the concept of investigation which may contain one or multiple plist evidences. Since keeping meticulous notes is important in handling evidence, we also added the facility to add notes at investigation level and at each plist (i.e. evidence) level. As a group, we created a simple write-up of the project, and then agreed upon a common understanding of the final goals.  We were then able to start implementing in parallel through dividing up the tasks. The above concepts are captured in a single image of our product below:



# Deciding data structures

After having consensus among us on the outline of how the application is going to work, we decided to leverage the powerful built-in classes available in Java SDK: JTree, JTable, JMenuItem, JPopupMenu etc.

The hierarchical structure of the data that comprises a Plist file brought forth the challenge of defining the simple means of displaying its structure and the data contained. This challenge was resolved using a combination of two Java Swing components: the JTree and the JTable. The JTree was used to provide a simple yet effective means of navigating the hierarchy of the Plist and JTable was used to display the data in tabular form.



# Output file creation

The ‘dd-plist” open source Java plist parser (<http://code.google.com/p/plist/>) provided a convenient way to store plist files in XML and binary format. But it didn’t have the facility to store the plist in PDF format. We communicated with the author of the dd-plist package and he replied stating that he has no idea of how to help us on that.

We did research and found two potential candidates: iTEXT and Apache PdfBox. Upon further research we found that the iTEXT package, although would be useful, was not free. Hence we decided to use Apache PdfBox. While using that package, in the Java method of converting a text file to PDF file, we found a bug which hindered in creating a better formatted output. We filed a bug and sent email to the mailing list and never received a reply. So, we formatted the PDF output to the best of our ability.

# Adding Search Capability

Providing search capability in an investigation or a plist (i.e. evidence) was our stretch goal. Once we had the package working with PDF file creation, we decided to put in more time and develop the search feature. Using the search feature an investigator can search for fixed strings in plist(s) (i.e. evidence) or in an investigation, including the notes in each element.

# System Requirements

TBRT Plist Navigator requires JRE1.6 or JRE1.7 running on Windows XP or Windows 7 environment. Since Java VM consumes considerably memory, we recommend a X86 system with at least 2GB of RAM, 1.86 GHZ CPU and a minimum of 4 Gb of hard-disk storage.

# Plist Tool Comparison

There are a few plist reader tools available in the market. But they are expensive and some are not user friendly. A comparison of Plist tools against our TBRT Plist Navigator is tabulated below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name of the tool** | **CellDEK** | **Oxygen** | **CelleBrite UFED** | **.XRY** | **Paraben Device Seizure** | **TBRT plist reader** |
| Plist Viewer | No | Yes | No | No | No | Yes |
| Price (US Dollars) | $14457 | $1970 | $3943 | $6572 | $1181 | Open Source and FREE! |
| Plist processing on Windows Support | No | Yes | No | No | No | Yes |
| Individual  Binary Plist read on Windows | No | No | No | No | No | Yes |
| Save plist as PDF on Windows | No | Yes | No | No | No | Yes |
| Search for content in plist and in investigation containing plist | No | No | No | No | No | Yes |
| Save individual plist as XML | Yes | Yes | No | No | No | Yes |

From the above comparison chart, one can see we have produced a product i.e. “TBRT Plist Navigator” that is quite valuable to the Forensic community using the Open Source community, using what we learned in our course and our creativity.

# Licensing

As we are not legal experts in software licensing issues. We did the due diligence on licensing to the best of our knowledge. We noticed the Java plist property library <http://code.google.com/p/plist/> is of MIT license (<http://www.opensource.org/licenses/mit-license.php>) , and Apache PdfBox is of Apache license (<http://www.apache.org/licenses/LICENSE-2.0>). The code we published itself is open source and free. Beyond that we leave the usage licensing responsibility to the end user as we are not legal experts in software licensing.

# Conclusion

In the beginning of the project we had too many unknowns. We had no Mac expertise and 75% of the group had no Java expertise. We had to learn about plist, Java, and apply the concepts we learned in the class to this project. Above all, we wanted to do our best so that our product is used in daily practice. We did all we could and at the end of our coding phase, we think the product is in a shape that a Forensic Investigator can use on multiple platforms in working with plist.

We have provided an Open Source plist reader tool, which is an alternative as New York State Police department desired, to expensive plist Forensic products available.