MALICIOUS STREAMS

Mac OS X Startup

The world according to launchd

The need to persist tools and infections between reboots is critical for the cyber criminal. In the Microsoft Windows world, we have an established body of knowledge and tools for determining programs set to launch at startup. This same level of maturity does not exist for the Mac OS X platform. Even though details of OS X's startup systems have been widely published, there is a lack of dissemination of this information within the Forensics community. Further, there exists a gap in open source tools to aid in the compilation of OS X startup items. The intent of this article is to explore the startup mechanisms of OS X and to introduce a basic tool to help with the examination of Mac OS X systems.

Examining OS X startup in detail reveals a tale of two worlds. The first shows the *NIX roots of the OS that makes use of traditional startup files such as those found in Table 1. These files could be leveraged to start nearly any process, but in practice is mainly used to set up the system and user's environment variables and shell configuration.

/etc/rc.*	~/.bashrc
/etc/profile	~/.profile
/etc/bashrc	~/.login

Table 1: Traditional *NIX Startup Files

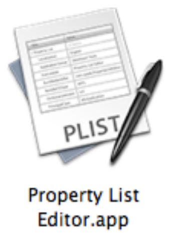
The second world brings the OS X specific touch with the implementation of the *launchd* daemon (*/sbin/launchd - OSX 10.4+*). The *launchd* daemon is responsible for starting and stopping most of the OS X specific processes. The list of processes to be started by this daemon, along with their initial configuration, is scattered across hundreds of XML based Property Lists (plist) files.

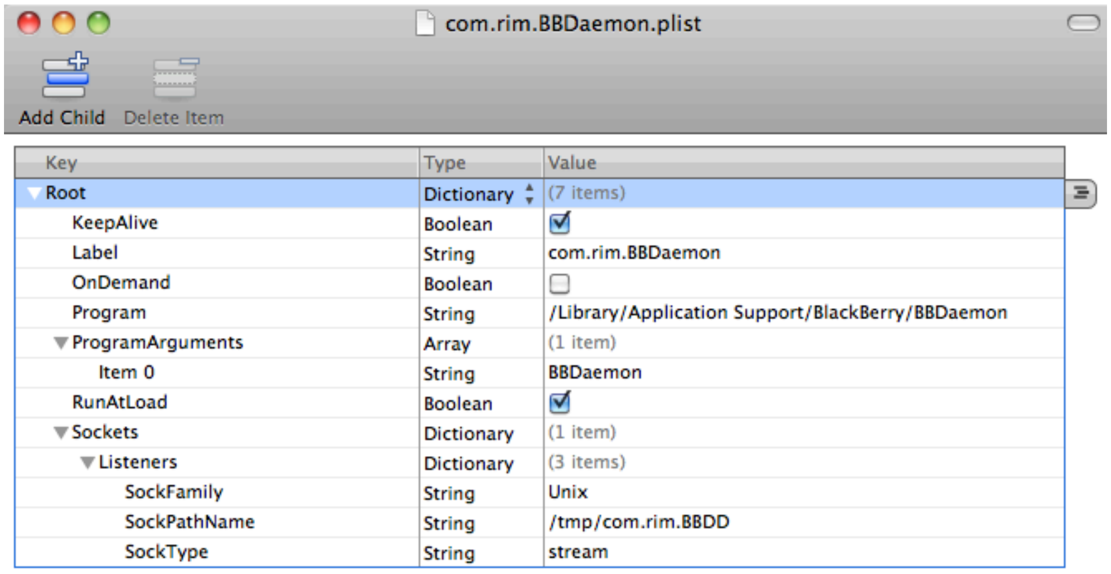
These files are housed in multiple system and user directories (Table 2).

/System/Library/LaunchDaemons/*
/System/Library/LaunchAgents/*
/Library/LaunchDaemons/*
/Library/LaunchAgents/*
/Library/StartupItems/*
/Library/Preferences/com.apple.loginwindow.plist
~/Library/LaunchAgents/*
~/Library/Preferences/loginwindow.plist

Table 2: Directories & Paths Used by the Launchd Daemon

These plists utilize a hierarchal (XML[1]) structure to house named-valued pairs to store configuration data. OS X uses plist files much like MS Windows uses the Registry to store configuration data. The primary difference is the configuration data is not centrally stored in hives but rather distributed across many text (or less commonly in binary) files located in various directories throughout the filesystem. Locating the correct plist used for a particular configuration can be a challenge but luckily those used by *launchd* are corralled into a small subset of system and user directories, Table 2. Many tools exist to view these files with one popular option being Apple's Property List Editor. This tool provides an easy to use interface, Figure 1, for viewing & modifying both text based and binary plist files. Property List Editor is delivered as part of Apple's free Xcode Developer suite. Alternate tools for viewing plists include XML editors, WWW browsers, and text editors.



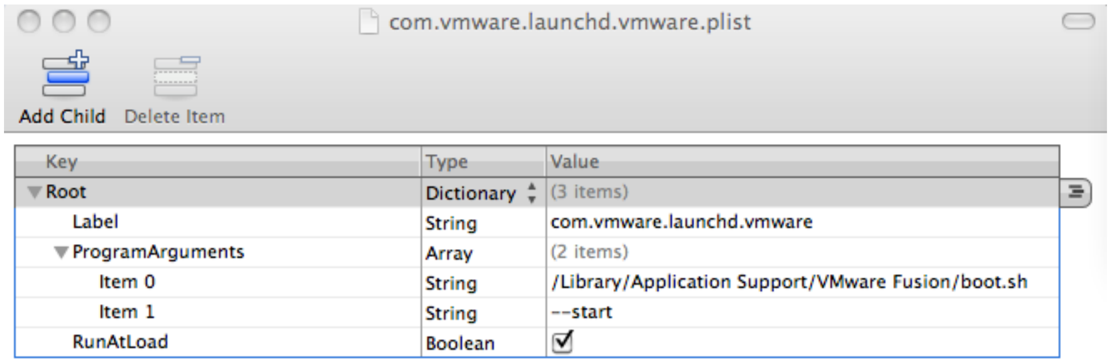


Key	Type	Value
Root	Dictionary	(7 items)
KeepAlive	Boolean	✓
Label	String	com.rim.BBDaemon
OnDemand	Boolean	☐
Program	String	/Library/Application Support/BlackBerry/BBDaemon
ProgramArguments	Array	(1 item)
Item 0	String	BBDaemon
RunAtLoad	Boolean	✓
Sockets	Dictionary	(1 item)
Listeners	Dictionary	(3 items)
SockFamily	String	Unix
SockPathName	String	/tmp/com.rim.BBDD
SockType	String	stream

Figure 1: Property List Editor Modifying a Startup Plist

The majority of *launchd* plists utilize a structure that specifies one startup executable per plist file. The specification of the startup executable is accomplished by a combination of the *Program* and *ProgramArguments* fields within the plist. As shown in Figure 1, the *Program* field specifies the full path to the executable and *ProgramArguments* specifies the command line arguments passed to the executable. An alternate approach employed by some *launchd* plists is to exclude the *Program* field and specify the fullpath to the executable as argument 0 of the *ProgramArguments* field, Figure 2.

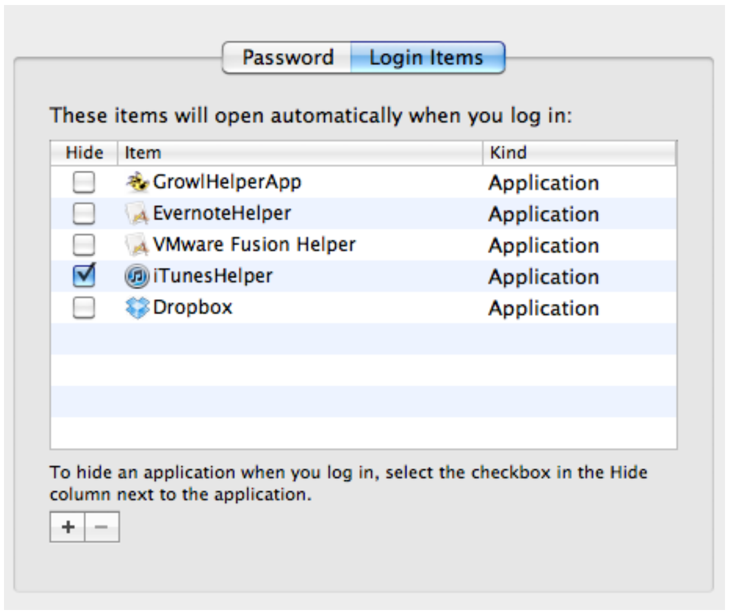
[1] Some older plist files use a raw text format rather than the XML standard



Key	Type	Value
Root	Dictionary	(3 items)
Label	String	com.vmware.launchd.vmware
ProgramArguments	Array	(2 items)
Item 0	String	/Library/Application Support/VMware Fusion/boot.sh
Item 1	String	--start
RunAtLoad	Boolean	✓

Figure 2: Alternate plist format - Executable Path Passed as *ProgramArguement : Item 0*

Finally, a specialty plist named loginwindow.plist (see Table 2 for full path) specifies items that are set to launch at user login. This plist serves as the backend data store for the graphical *Login Items* tab included in the *Accounts* section of the *Systems Preferences*, Figure 3.



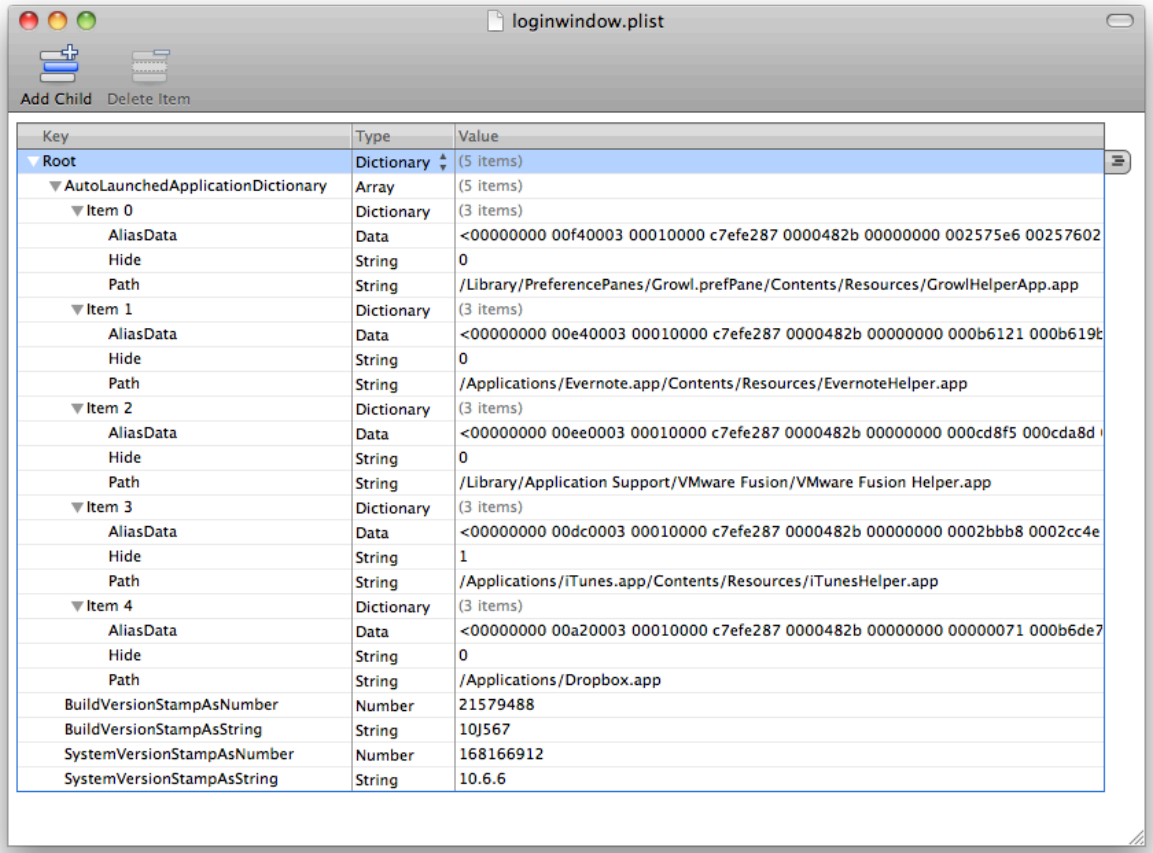
These items will open automatically when you log in:

Hide	Item	Kind
☐	GrowlHelperApp	Application
☐	EvernoteHelper	Application
☐	VMware Fusion Helper	Application
✓	iTunesHelper	Application
☐	Dropbox	Application

To hide an application when you log in, select the checkbox in the Hide column next to the application.

Figure 3: Login Items Section of System Preferences: Accounts

The format of *loginwindow.plist* deviates from the plist files described earlier. This file uses a root node field called *AutoLaunchedApplicationDictionary* with a number of *Item* # child nodes. Each child node specifies the full path of an executable and the visibility of the running application. Figure 4 shows the layout of a loginwindow.plist that supports the configuration shown in figure 3.

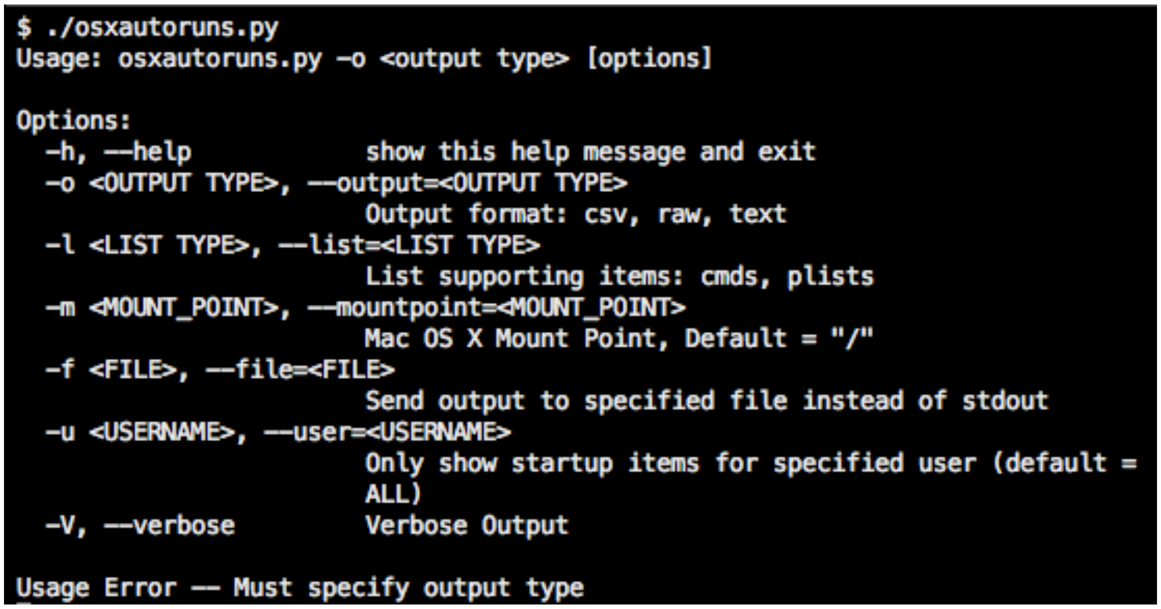


Key	Type	Value
Root	Dictionary	(5 items)
AutoLaunchedApplicationDictionary	Array	(5 items)
Item 0	Dictionary	(3 items)
AliasData	Data	<00000000 00f40003 00010000 c7efe287 0000482b 00000000 002575e6 00257602
Hide	String	0
Path	String	/Library/PreferencePanes/Growl.prefPane/Contents/Resources/GrowlHelperApp.app
Item 1	Dictionary	(3 items)
AliasData	Data	<00000000 00e40003 00010000 c7efe287 0000482b 00000000 000b6121 000b619b
Hide	String	0
Path	String	/Applications/Evernote.app/Contents/Resources/EvernoteHelper.app
Item 2	Dictionary	(3 items)
AliasData	Data	<00000000 00ee0003 00010000 c7efe287 0000482b 00000000 000cd8f5 000cda8d
Hide	String	0
Path	String	/Library/Application Support/VMware Fusion/VMware Fusion Helper.app
Item 3	Dictionary	(3 items)
AliasData	Data	<00000000 00dc0003 00010000 c7efe287 0000482b 00000000 0002bbb8 0002cc4e
Hide	String	1
Path	String	/Applications/iTunes.app/Contents/Resources/iTunesHelper.app
Item 4	Dictionary	(3 items)
AliasData	Data	<00000000 00a20003 00010000 c7efe287 0000482b 00000000 00000071 000b6de7
Hide	String	0
Path	String	/Applications/Dropbox.app
BuildVersionStampAsNumber	Number	21579488
BuildVersionStampAsString	String	10J567
SystemVersionStampAsNumber	Number	168166912
SystemVersionStampAsString	String	10.6.6

Figure 4: loginwindow.plist structure

AUTOMATED ANALYSIS

While examining each plist file manually is an achievable task, it can become very time consuming when multiplied by the hundreds of files involved in OS X startup. *osxautoruns.py* is a new open source, python-based tool that automates this task. *osxautoruns.py* locates all plist files specified by the filestructure outlined in Table 2 and extracts relevant startup information. The tool also supports various output formats and the ability to analyze the current system (live analysis) or a mounted image. Figure 5 shows basic syntax and Figure 6 provides a snippet of sample output. Additional details and options are documented in the README.txt file included within application download bundle.

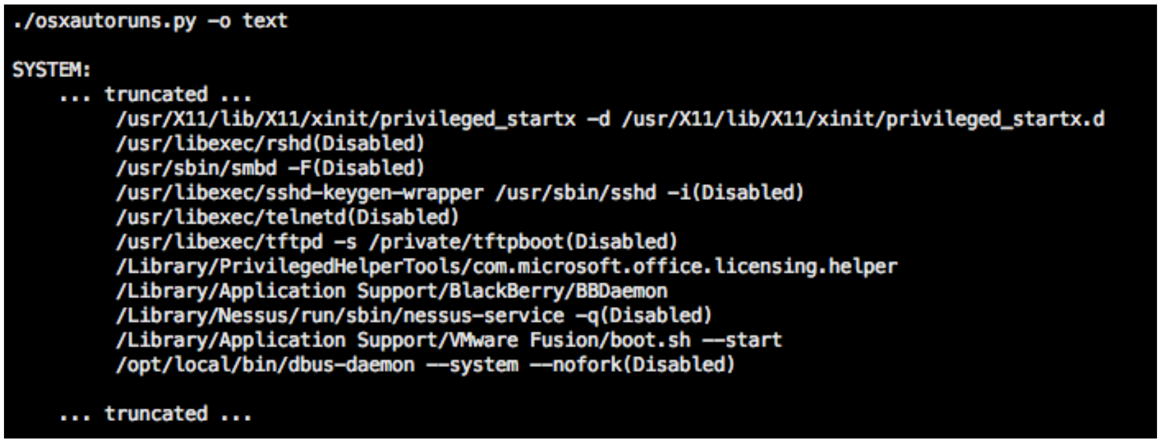


```
$ ./osxautoruns.py
Usage: osxautoruns.py -o <output type> [options]

Options:
  -h, --help            show this help message and exit
  -o <OUTPUT TYPE>, --output=<OUTPUT TYPE>
                        Output format: csv, raw, text
  -l <LIST TYPE>, --list=<LIST TYPE>
                        List supporting items: cmds, plists
  -m <MOUNT_POINT>, --mountpoint=<MOUNT_POINT>
                        Mac OS X Mount Point, Default = "/"
  -f <FILE>, --file=<FILE>
                        Send output to specified file instead of stdout
  -u <USERNAME>, --user=<USERNAME>
                        Only show startup items for specified user (default =
                        ALL)
  -V, --verbose         Verbose Output

Usage Error -- Must specify output type
```

Figure 5: osxautoruns.py Usage Information



```
./osxautoruns.py -o text

SYSTEM:
    ... truncated ...
         /usr/X11/lib/X11/xinit/privileged_startx -d /usr/X11/lib/X11/xinit/privileged_startx.d
         /usr/libexec/rshd(Disabled)
         /usr/sbin/smbd -F(Disabled)
         /usr/libexec/sshd-keygen-wrapper /usr/sbin/sshd -i(Disabled)
         /usr/libexec/telnetd(Disabled)
         /usr/libexec/tftpd -s /private/tftpboot(Disabled)
         /Library/PrivilegedHelperTools/com.microsoft.office.licensing.helper
         /Library/Application Support/BlackBerry/BBDaemon
         /Library/Nessus/run/sbin/nessus-service -q(Disabled)
         /Library/Application Support/VMware Fusion/boot.sh --start
         /opt/local/bin/dbus-daemon --system --nofork(Disabled)

    ... truncated ...
```

Figure 6: osxautoruns.py Example Output

CAUTION: *This utility does NOT examine the traditional *NIX startup files rc.*, profile, bashrc, etc. that exist on Mac OS X systems. This tool simply automates the parsing of the plists outlined above and extracts Program & ProgramArguements fields. The output of this tool should not be considered a complete listing of all potential startup items.*

ADDITIONAL INFORMATION & REFERENCES

There are many sources of additional information available. Below are a few of those sources that were used in the development of this paper.

osxautoruns.py
Malicious Streams: Downloads
http://www.malicious-streams.com/Downloads/Downloads.html

launchd
Wikipedia
http://en.wikipedia.org/wiki/Launchd - Property_list

Creating launchd Daemons and Agents
Mac OS X Reference Library, Apple Inc.
http://developer.apple.com/library/mac/ -
documentation/MacOSX/Conceptual/BPSystemStartup/Articles/LaunchOnDemandDaemons.html

Mac OS X System Startup
Mac OS X Internals, Amit Singh
http://osxbook.com/book/bonus/ancient/whatismacosx//arch_startup.html