OSXAuditor

edj

Table of Contents

Contribution Guide

This page describes how to contribute changes to OSXAuditor.

Please do **not** create a pull request without reading this guide first. Failure to do so may result in the **rejection** of the pull request.

Philosophy

Here are some core ideas to keep in mind for contributing to OSXAuditor

- At it's core OSXAuditor must be as handy as possible for the Responder.
- OSXAuditor is most effective and helpful if it can run on a stock system without installing any additional tools.
- Must work properly on either live or dead os x system (ie. hard drive copy).
- All output must go through the printandlog() function so that people can add new output format easily.
- All the hases go into the global hash db.

Submitting to OSXAuditor

Submitting Issues

Issues are helpful if you're experiencing issues or want to suggest a feature (code is even better though). Please make sure you provide as much detail as possible including stack traces, screenshots, and code samples.

Submitting Code

If your changes need to be modified due to some reviews, it is less clutter to tweak an isolated feature branch and push it again.

We welcome pull requests. Here's a quick guide:

1. Fork the repo.
2. Create a feature branch.
3. Build.
4. Push to your fork and submit a pull request.

At this point you're waiting on us. We like to at least comment on, if not accept, pull requests within three business days (and, typically, one business day). We may suggest some changes or improvements or alternatives.

Some things that will increase the chance that your pull request is accepted,:

- Use Python idioms and helpers
- Update the documentation, the surrounding one, examples elsewhere, guides, whatever is affected by your contribution

Syntax

- Four spaces, no tabs.
- No trailing whitespace.
- Prefer &&/|| over and/or.
- a = b and not a=b.
- Follow the conventions you see used in the source already.
- When in doubt default to PEP 8 -- Style Guide for Python Code.

Data-Driven Documents

D3.js is a JavaScript library for manipulating documents based on data. **D3** helps you bring data to life using HTML, SVG and CSS. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

Want to learn more? See the wiki.

For examples, see the gallery and mbostock's bl.ocks.

OS X Auditor

OS X Auditor is a free Mac OS X computer forensics tool.

OS X Auditor parses and hashes the following artifacts on the running system or a copy of a system you want to analyze: * the kernel extensions * the system agents and daemons * the third party's agents and daemons * the old and deprecated system and third party's startup items * the users' agents * the users' downloaded files * the installed applications

It extracts: * the users' quarantined files * the users' Safari history, downloads, topsites, LastSession, HTML5 databases and localstore * the users' Firefox cookies, downloads, formhistory, permissions, places and signons * the users' Chrome history and archives history, cookies, login data, top sites, web data, HTML5 databases and local storage * the users' social and email accounts * the WiFi access points the audited system has been connected to (and tries to geolocate them)

It also looks for suspicious keywords in the .plist themselves.

It can verify the reputation of each file on: * Team Cymru's MHR * VirusTotal * Malware.lu * your own local database

It can aggregate all logs from the following directories into a zipball: * /var/log (-> /private/var/log) * /Library/logs * the user's ~/Library/logs

Finally, the results can be: * rendered as a simple txt log file (so you can cat-pipe-grep in them… or just grep) * rendered as a HTML log file * sent to a Syslog server

Author

Jean-Philippe Teissier - @Jipe_

How to install

Just copy all files from github

Dependencies

If you plan to run OS X Auditor on a Mac, you will get a full plist parsing support with the OS X Foundation through pyobjc: * pip install pyobjc

If you can't install pyobjc or if you plan to run OS X Auditor on another OS than Mac OS X, you may experience some troubles with the plist parsing: * pip install biplist * pip install plist

These dependencies will be removed when a working native plist module will be available in python

How to run

- OS X Auditor runs well with python >= 2.7.2 (2.7.5 is OK). It does not run with a different version of python yet (due to the plist nightmare)
- OS X Auditor is written to work on Moutain Lion. It will do his best on older OS X versions.
- You must run it as root (or via sudo) if you want to use is on a running system, otherwise it won't be able to access some system and other users' files

Type osxauditor.py -h to get all the available options, then run it with the selected options

eg. [sudo] python osxauditor.py -a -m -l localhashes.db -H log.html

Changelog

0.4.1

- CHANGE: Search for generic backdoors in LaunchAgentPlist

0.4

- NEW: extracts events (boot/shutdown, hibernation in/out, sudo commands, usb devices, ttys opened/closed, from the system logs and create a (not readable yet) timeline (-e/eventlogs)
- NEW: extracts users' LoginItems
- NEW: extracts users' RecentItems
- NEW: extracts the LastSession from Safari artifacts
- NEW: extract system groups and users details
- FIX: wrong os.path.join() calls
- FIX: bug in the recursive ParsePackagesDir()

0.3.1

- NEW: provides with the system name, version and build of the audited system
- NEW: ability to analyze installed Applications (-i/--installedapps)
- NEW: extracts the Archived History from Google Chrome artifacts
- NEW: a human readable HTML log report :)
- FIX: HTMLLog() and SYSLOGLog() now handle exceptions
- FIX: ParsePackagesDir() is now recursive and only tries to parse apps or kernel extensions. Some DEBUG output added as well
- FIX: HUGE UTF-8/UNICODE improvement
- FIX: .DS_Store and .localized files are ignored in ParsePackagesDir()

0.3

- NEW: ability to parse Google Chrome artifacts (History and archives history, Cookies, Login Data, Top Sites, Web Data, HTML5 databases and local storage) with -b/--browsers
- NEW: ability to extract the Wi-Fi APs the audited system has been connected to from the Airport Preferences and tries to geolocate them using Geomena (-A/--airportprefs). You must use -g/--wifiapgeolocate to enable the geolocation (or set GEOLOCATE_WIFI_AP to True in the code).
- NEW: ability to extract users' social and email accounts (-U/--usersaccounts)
- FIX: ability to handle the locked sqlite databases especially while auditing a live system
- FIX: hashes duplicates removed
- FIX: better identify md5 in the HTML output
- CHANGE: indicates if a section (Startup items, Packages directory, Db tables, etc…) is empty to clarify the output
- CHANGE: the downloads artifacts (-d/--downloads) include the old and new Mail.app default download directories

0.2.1

- CHANGE/FIX: implement a BigFileMd5() function to hash very big files, avoid MemoryError execptions and reduce the memory footprint
- FIX: UTF-8 entries from LSQuarantineEvent in ParseQuarantines()

0.2

- NEW: ability to send the results to a remote syslogd server (-S)
- NEW: ability to create a zipball of all the log files found on the audited system (-z)
- CHANGE: the analysis of startup artifacts includes the old and deprecated StartupItems
- CHANGE: the analysis of startup artifacts includes the ScriptingAdditions
- CHANGE: the analysis of quarantined artifact includes the old QuarantineEvents for Mac OS X systems <= 10.6
- CHANGE: great improvement of plist hangling using the Python Objective-C bridge (PyObjC) and OS X Foundation
- CHANGE: some changes in the options parameters (-t, -l)
- CHANGE: license changed from CC to GPL
- CHANGE: debug levels are now more consitent in the output logs
- CHANGE: a small change with the Bootstrap CSS
- CHANGE: the VirusTotal lookup is now done in a bulk mode
- FIX: a bug in ParseLaunchAgents() on plist files containing both Program and ProgramArguments keys

0.1

- Initial Release

Design & Capabilities

Design & Capabilities

Artifacts

Users

- Library/Preferences/com.apple.LaunchServices.QuarantineEventsV2
- Library/Preferences/com.apple.LaunchServices.QuarantineEvents
- Library/Preferences/com.apple.loginitems.plist
- Library/Mail Downloads/
- Library/Containers/com.apple.mail/Data/Library/Mail Downloads
- Library/Accounts/Accounts3.sqlite
- Library/Containers/com.apple.mail/Data/Library/Mail/V2/MailData/Accounts.plist

- Library/Preferences/com.apple.recentitems.plist

- Firefox
- Library/Application Support/Firefox/Profiles/
- cookies.sqlite
- downloads.sqlite
- formhistory.sqlite
- places.sqlite
- signons.sqlite
- permissions.sqlite
- addons.sqlite
- extensions.sqlite
- content-prefs.sqlite
- healthreport.sqlite

- webappsstore.sqlite

- Safari
- Library/Safari/
- Downloads.plist
- History.plist
- TopSites.plist
- LastSession.plist
- Databases

- LocalStorage

- Chrome
- Library/Application Support/Google/Chrome/Default/
- History
- Archived History
- Cookies
- Login Data
- Top Sites
- Web Data
- databases

- Local Storage

System

- /System/Library/LaunchAgents/
- /System/Library/LaunchDaemons/
- /System/Library/ScriptingAdditions/

- /System/Library/StartupItems/Library/ScriptingAdditions/
- /System/Library/Extensions/
- /System/Library/CoreServices/SystemVersion.plist
- /Library/LaunchAgents/
- /Library/LaunchDaemons/
- /Library/StartupItems/
- /Library/Preferences/SystemConfiguration/com.apple.airport.preferences.plist
- /Library/logs
- /var/log
- /etc/localtime
- StartupParameters.plist
- /private/var/db/dslocal/nodes/Default/groups/admin.plist
- /private/var/db/dslocal/nodes/Default/users

TODO

- extract user info from /private/var/db/dslocal/nodes/Default/users

Related work

Disk Arbitrator

Disk Arbitrator is Mac OS X forensic utility designed to help the user ensure correct forensic procedures are followed during imaging of a disk device. Disk Arbitrator is essentially a user interface to the Disk Arbitration framework, which enables a program to participate in the management of block storage devices, including the automatic mounting of file systems. When enabled, Disk Arbitrator will block the mounting of file systems to avoid mounting as read-write and violating the integrity of the evidence.

https://github.com/aburgh/Disk-Arbitrator

Volafox

volafox a.k.a 'Mac OS X Memory Analysis Toolkit' is developed on python 2.x

https://code.google.com/p/volafox/

Mandiant Memoryze(tm) for the Mac

Memoryze for the Mac is free memory forensic software that helps incident responders find evil in memory… on Macs. Memoryze for the Mac can acquire and/or analyze memory images. Analysis can be performed on offline memory images or on live systems.

http://www.mandiant.com/resources/download/mac-memoryze

Volatility MacMemoryForensics

https://code.google.com/p/volatility/wiki/MacMemoryForensics

License

Table of Contents