
AutoMacTC: Automated Mac Forensic Triage Collector

Purpose

This is a modular forensic triage collection framework designed to access various forensic artifacts on macOS, parse them, and present them in formats viable for analysis. The output may provide valuable insights for incident response in a macOS environment. Automactc can be run against a live system or dead disk (as a mounted volume.)

Requirements

- Python 3.9 or earlier (backwards compatible with Python 2.7)
- MacOS target systems, for live collection (successfully tested on macOS major releases 10.11 through 11.2.3 as well the as M1 processor)
- MacOS analysis systems, for triage against a mounted disk image (disk images from macOS 10.11 through 10.15 systems are supported)

Basic usage

At its simplest, you can run automactc with the following invocation.

```
1 sudo /usr/bin/python automactc.py -m all
```

This will run all modules (-m) with default settings, i.e.

```
1 - default input directory will be /, or the root of the current volume
2 - default output directory will be ./, or the working directory from
   which automactc is run (NOT the location of the script)
3 - default prefix for output filenames will be automactc-output
4 - default behavior is to populate a runtime.log for debugging and info
5 - default format for individual artifacts output files is CSV
6 - default CPU priority is set to low
7 - default behavior on completion is to compress all output files to tar
   .gz
```

using the version of python that is pre-installed on the macOS device.

In order to list all available modules and do nothing else, simply run:

```
1 automactc.py -l
```

The inputdir and outputdir can be specified with the -i and -o flags, respectively.

```
1 automactc.py -i / -o /automactc_output -m all
```

For macOS 10.15+ systems, the `-is` flag is used to specify the input system drive if using mounted drive from 10.15+ system (e.g. “Macintosh HD”).

Modules can be specified for inclusion or exclusion on a per-module basis. In other words, you can INCLUDE specific modules, such as `pslist`, `bash`, and `profiler`:

```
1 automactc.py -m pslist bash profiler
```

Or, you can exclude specific modules, to run all EXCEPT those specified, such as `dirlist` and `autoruns`:

```
1 automactc.py -x dirlist autoruns
```

Output Control

For every module, `automactc` will generate an output file and populate it with data. The output file format defaults to CSV, but can be toggled to JSON with the `-fmt` flag. It is not currently possible to specify output format on a per-module basis.

```
1 automactc.py -m all -fmt json
```

Upon successfully populating the output file with data, the file is rolled into a `.tar` archive that is generated when `automactc` completes its first module. Upon completion of the last module, `automactc` will GZIP the `.tar` archive to `.tar.gz`.

The name of the tar archive follows the following naming convention:

```
1 prefix,hostname,ip,automactc_runtime.tar
```

The first field, `prefix`, can be specified at runtime with `-p`. If unspecified, the prefix is set to `automactc-output`. The other fields are populated from data gathered at runtime. This is useful when running `automactc` on several systems for a single incident.

```
1 automactc.py -m all -p granny-smith
```

While the default behavior is to generate a tarball, use of the `-nt` flag will prevent the creation of a tar archive and will leave the output files as-is in the output directory.

```
1 automactc.py -m all -p granny-smith -nt
```

Current Modules

-
- ```
1 - pslist (current process list at time of automactc run)
2 - lsof (current file handles open at time of automactc run)
3 - netstat (current network connections at time of automactc run)
4 - unifiedlogs (collect Unified Logging events from a live system based
 on specified predicates)
5 - asl (parsed Apple System Log (.asl) files)
6 - auditlog (parsing audit log files from private/var/audit/)
7 - autoruns (parsing of various persistence locations and plists)
8 - bash (parsing bash/.*_history files for all users)
9 - chrome (parsing chrome visit history and download history)
10 - cookies (parsing the cookies database for each user for chrome and
 firefox)
11 - coreanalytics (parsing program execution evidence produced by Apple
 diagnostics)
12 - dirlist (list hof files and directories across the disk)
13 - eventtaps (parsing event tap items)
14 - firefox (parsing firefox visit history and download history)
15 - installhistory (parsing program installation history)
16 - mru (parsing SFL and MRU plist files)
17 - netconfig (parsing airport and network interface settings)
18 - quarantines (parsing QuarantineEventsV2 database)
19 - quicklook (parsing Quicklooks database)
20 - safari (parsing safari visit history and download history)
21 - spotlight (parsing user spotlight top searches)
22 - ssh (parsing known_hosts and authorized_keys files for each user)
23 - syslog (parsing system.log files)
24 - systeminfo (basic system identification, such as current IP address,
 serial no, hostname)
25 - terminalstate (decode and parse savedState files for the Terminal
 application for each user)
26 - users (listing present and deleted users on the system)
27 - utmpx (listing user sessions on terminals)
```

## Advanced usage

One can utilize the **-rtr** flag to reduce verbosity of some modules to display nicely on CrowdStrike RTR console. Specifically the real time updates of the dirlist module are reduced in order to not overflow the console window.

```
1 automactc.py -m all --rtr
```

AutoMacTC can be deployed and executed with the provided sample bash wrapper **deploy.sh**. The provided wrapper will - execute AutoMacTC with the version python installed at **/usr/bin/python** - use the **-rtr** flag to reduce verbosity when running remotely via a terminal - use the **-prefix** 'automactc-output' - output in **json** format - exclude the 10.14/10.15+ unsupported live modules **quicklooks**, **coreanalytics**, and **safari**

---

To use the wrapper script: - Compress the automactc folder into a tar.gz archive - Copy the archive and the wrapper script to the host system into their own folder (or a location such as /private/tmp or /tmp) - Run the wrapper with sudo

```
1 sudo bash deploy.sh
```

By default, automactc populates verbose debug logging into a file named `prefix,hostname,ip, runtime.log`. You can disable the generation of this log with:

```
1 automactc.py -m all -nl
```

By default, automactc will print the INFO and ERROR log messages to the console. To run automactc in quiet mode and write NO messages to the console, use -q. INFO messages include program startup messages, one message per module start, and completion/cleanup messages.

```
1 automactc.py -m all -q
```

To print DEBUG messages to the console along with INFO and ERROR messages, use the -d flag.

```
1 automactc.py -m all -d
```

Automactc runs with the lowest CPU priority (niceness) possible by default. It is possible to disable niceness and run at a normal priority with the -r flag.

```
1 automactc.py -m all -r
```

Automactc can also be run against a dead disk, if the disk is mounted as a volume on the analysis system. Once mounted, run automactc with the appropriate inputdir (pointing to the Volume mount point) and -f to toggle forensic mode ON.

NOTE: for a live system, if you wish to collect dirlist on mounted peripheral devices, you can use -f with -i /, else dirlist will not recurse further into mounted /Volumes.

```
1 automactc.py -i /Volumes/mounted_IMAGE/ -o /path/to/output -f -m all
```

## Dirlist Arguments

### Directory Inclusion/Exclusion

It is possible to limit dirlist recursion to specific directories with the -K flag. By default, dirlist will attempt to recurse from the root of the inputdir volume unless otherwise specified with this flag. Multiple directories can be specified in a space separated list.

```
1 automactc.py -m dirlist -K /Users/ /Applications/ /tmp
```

---

It is also possible to exclude specific directories from dirlist recursion with the -E flag.

```
1 automactc.py -m dirlist -E /path/to/KnownDevDirectory
```

By default, the following directories and file are excluded on live systems:

```
1 /.fsevents (to reduce output verbosity)
2 /.DocumentRevisions-V100 (to reduce output verbosity)
3 /.Spotlight-V100 (to reduce output verbosity)
4 /Users/*/Pictures (to avoid permissions errors)
5 /Users/*/Library/Application Support/AddressBook (to avoid permissions
 errors)
6 /Users/*/Calendar (to avoid permissions errors)
7 /Users/*/Library/Calendars (to avoid permissions errors)
8 /Users/*/Library/Preferences/com.apple.AddressBook.plist (to avoid
 permissions errors)
9 /System/Volumes/Data/private/var/folders/kb/* (to reduce output
 verbosity)
10 /System/Volumes/Data/private/var/folders/zz/* (to reduce output
 verbosity)
```

By default, the following directories are excluded when running forensic mode against a mounted image:

```
1 /.fsevents (to reduce output verbosity)
2 /.DocumentRevisions-V100 (to reduce output verbosity)
3 /.Spotlight-V100 (to reduce output verbosity)
```

Any additional directories to exclude will be appended to this default list, unless you provide the -E no-defaults argument first, in which case only your specified directories will be excluded.

```
1 automactc.py -m dirlist -E no-defaults /path/to/KnownDevDirectory
```

## Hashing

*The hashing arguments below can be used for BOTH dirlist and the autoruns modules.*

By default, the dirlist module will hash files only with the sha256 algorithm. If you wish to use both the SHA256 and MD5 algorithms, use -H sha256 md5. If you wish to use only md5, use -H md5. If you wish to use neither, use -H none. NOTE: If you run the dirlist module against a dead disk with hashing enabled, this currently takes a LONG time to run.

```
1 automactc.py -m dirlist -H sha256 md5
```

By default, the dirlist module will only hash files with sizes under 10mb. To override this setting and hash files under a different size threshold, the threshold can be changed with the -S flag in number of

---

megabytes. NOTE: increasing the size threshold will likely increase the amount of time it takes to run the dirlist module. For example, to hash files up to 15MB:

```
1 automactc.py -m dirlist -S 15
```

## Bundles, Signatures, Multithreading

By default, the dirlist module will NOT recurse into bundle directories, including the following:

```
1 '.app', '.framework', '.lproj', '.plugin', '.kext', '.osax', '.bundle', '.driver', '.wdgt', '.Office', '.blacklight'
```

To override this setting, use the -R flag. NOTE: this produces a far higher volume of output and takes significantly more time. These bundle directories will be configurable in a future update.

By default, the dirlist module will check codesignatures for all .app, .kext, and .osax files found. To prevent the dirlist module from checking any code signatures, use the -NC flag. *This argument can be used for BOTH dirlist and the autoruns modules.*

```
1 automactc.py -m dirlist -NC
```

By default, the dirlist module has been multithreaded to increase processing speed. Multithreading can be disabled with the -NM flag.

```
1 automactc.py -m dirlist -NM
```

## Unified Logs Live module

By default, to reduce verbosity and time taken, only a subset of the total available sample predicates are enabled. You can optionally enable additional predicates by removing the comment character from existing predicates or by adding your own custom predicates.

The sample set of predicates was obtained from <https://www.crowdstrike.com/blog/how-to-leverage-apple-unified-log-for-incident-response/>.

## Help Menu

```
1 usage: automactc.py [-m INCLUDE_MODULES [INCLUDE_MODULES ...] | -x
2 EXCLUDE_MODULES [EXCLUDE_MODULES ...] | -l] [-h] [-v]
3 [-i INPUTDIR] [-is INPUTSYSDIR] [-o OUTPUTDIR] [-p
4 PREFIX]
5 [-f] [-nt] [-nl] [-fmt {csv,json}] [-np] [-b] [-O]
```

---

```

5 [-q | -r | -d]
6 [-K DIR_INCLUDE_DIRS [DIR_INCLUDE_DIRS ...]]
7 [-E DIR_EXCLUDE_DIRS [DIR_EXCLUDE_DIRS ...]]
8 [-H DIR_HASH_ALG [DIR_HASH_ALG ...]]
9 [-S DIR_HASH_SIZE_LIMIT] [-R] [-NC] [-NM]

```

AutoMacTC: an Automated macOS forensic triage collection framework.

module filter:

```

1 -m INCLUDE_MODULES [INCLUDE_MODULES ...], --include_modules
 INCLUDE_MODULES [INCLUDE_MODULES ...]
2 module(s) to use, use "all" to run all modules,
 space
3 separated list only
4 -x EXCLUDE_MODULES [EXCLUDE_MODULES ...], --exclude_modules
 EXCLUDE_MODULES [EXCLUDE_MODULES ...]
5 assumes you want to run all modules EXCEPT
 those
6 specified here, space separated list only
7 -l, --list_modules if flag is provided, will list available modules
 and
8 exit.

```

general arguments:

```

1 -h, --help show this help message and exit
2 -v, --verbose enable verbose logging
3 -i INPUTDIR, --inputdir INPUTDIR
4 input directory; mount dmg with mountdmg.sh
 script and
5 use -f to analyze mounted HFS or APFS Volume,
 use
6 volume appended with "Data" (e.g. "Macintosh HD
 -
 Data") for 10.15+ systems
7 -is INPUTSYSDIR, --inputsysdir INPUTSYSDIR
8 input system drive if using mounted drive from
9 10.15+
10 system (e.g. "Macintosh HD")
11 -o OUTPUTDIR, --outputdir OUTPUTDIR
12 output directory
13 -p PREFIX, --prefix PREFIX
14 prefix to append to tarball and/or output files
15 -f, --forensic_mode if flag is provided, will analyze mounted volume
16 provided as inputdir
17 -nt, --no_tarball if flag is provided, will NOT package output
18 files
19 into tarball
19 -nl, --no_logfile if flag is provided, will NOT generate logfile on
 disk

```

---

```

20 -fmt {csv,json}, --output_format {csv,json}
21 toggle between csv and json output, defaults to
22 csv
23 -np, --no_low_priority if flag is provided, will NOT run automactc
24 with
25 highest niceness (lowest CPU priority). high
26 niceness
27 is default
28 -b, --multiprocessing if flag is provided, WILL multiprocess modules
29 [WARNING: Experimental!]
30 -O, --override_mount if flag is provided, WILL bypass error where
 inputdir
 does not contain expected subdirs

```

console logging verbosity:

```

1 -q, --quiet if flag is provided, will NOT output to console
 at all
2 -r, --rtr reduce verbosity to display nicely on RTR console
3 -d, --debug enable debug logging to console

```

specific module arguments:

```

1 -K DIR_INCLUDE_DIRS [DIR_INCLUDE_DIRS ...], --dir_include_dirs
 DIR_INCLUDE_DIRS [DIR_INCLUDE_DIRS ...]
2 directory inclusion filter for dirlist module,
3 defaults to volume root, space separated list
 only
4 -E DIR_EXCLUDE_DIRS [DIR_EXCLUDE_DIRS ...], --dir_exclude_dirs
 DIR_EXCLUDE_DIRS [DIR_EXCLUDE_DIRS ...]
5 directory and file exclusion filter for dirlist
6 module. defaults are specified in README. space
7 separated list only. put 'no-defaults' as first
 item
8 to overwrite default exclusions and then
9 provide your
 own exclusions
10 -H DIR_HASH_ALG [DIR_HASH_ALG ...], --dir_hash_alg DIR_HASH_ALG [
 DIR_HASH_ALG ...]
11 either sha256 or md5 or both or none, at least
 one is
12 recommended, defaults to sha256. also applies
 to
13 autoruns module
14 -S DIR_HASH_SIZE_LIMIT, --dir_hash_size_limit DIR_HASH_SIZE_LIMIT
15 file size filter for which files to hash, in
16 megabytes, defaults to 10MB. also applies to
 autoruns
17 module

```



---

```
18 -R, --dir_recurse_bundles
19 will fully recurse app bundles if flag is
20 provided.
21 this takes much more time and space
22 -NC, --dir_no_code_signatures
23 if flag is provided, will NOT check code
24 signatures
25 for app and kext files. also applies to
26 autoruns
27 module
28 -NM, --dir_no_multithreading
29 if flag is provided, will NOT multithread the
30 dirlist
31 module
```