Beyond the good ol' LaunchAgents - 1 - shell startup files

theevilbit.github.io/beyond/beyond_0001

March 14, 2021

> This is part 1 in the series of "Beyond the good ol' LaunchAgents", where I try to collect various persistence techniques for macOS. For more background check the introduction.

Shell startup files are executed when our shell environment like `zsh` or `bash` is starting up. macOS defaults to `/bin/zsh` these days, and whenever we open `Terminal` or SSH into the device, this is the shell environment we are placed into. `bash` and `sh` are still available, however they have to be specifically started.

When we start `zsh` it checks number of files, and if there is an environment variable or command inside, it will be set or executed. Probably the most common is `.zshrc`, which is used to set the shell environment and execute commands. It can be found in the user's home directory. `bash` has a similar one, called `.bashrc`.

The man page of zsh, which we can read with `man zsh` has a long description of the startup files.

```
STARTUP/SHUTDOWN FILES
       Commands are first read from /etc/zshenv; this cannot be overridden.
Subsequent behaviour is modified by the RCS and GLOBAL_RCS options; the former
       affects  all  startup  files,  while  the second only affects global startup
files (those shown here with an path starting with a /).  If one of the
       options is unset at any point, any subsequent startup file(s) of the
corresponding type will not be read.  It is also possible for a file in  $ZDOT-
       DIR to re-enable GLOBAL_RCS. Both RCS and GLOBAL_RCS are set by default.

       Commands  are  then  read  from  $ZDOTDIR/.zshenv.  If the shell is a login
shell, commands are read from /etc/zprofile and then $ZDOTDIR/.zprofile.
       Then, if the shell is interactive, commands are read from /etc/zshrc and then
$ZDOTDIR/.zshrc.  Finally, if the shell is a login shell,  /etc/zlogin
       and $ZDOTDIR/.zlogin are read.

       When  a  login  shell  exits, the files $ZDOTDIR/.zlogout and then
/etc/zlogout are read.  This happens with either an explicit exit via the exit or
       logout commands, or an implicit exit by reading end-of-file from the terminal.
However, if the shell terminates due to  exec'ing  another  process,
       the logout files are not read.  These are also affected by the RCS and
GLOBAL_RCS options.  Note also that the RCS option affects the saving of his-
       tory files, i.e. if RCS is unset when the shell exits, no history file will be
saved.

       If ZDOTDIR is unset, HOME is used instead.  Files listed above as being in
/etc may be in another directory, depending on the installation.

       As /etc/zshenv is run for all instances of zsh, it is important that it be
kept as small as possible.  In particular, it is a good idea to put  code
       that  does  not need to be run for every single shell behind a test of the
form `if [[ -o rcs ]]; then ...' so that it will not be executed when zsh
       is invoked with the `-f' option.

       Any of these files may be pre-compiled with the zcompile builtin command (see
zshbuiltins(1)).  If a compiled file exists (named  for  the  original
       file plus the .zwc extension) and it is newer than the original file, the
compiled file will be used instead.
```

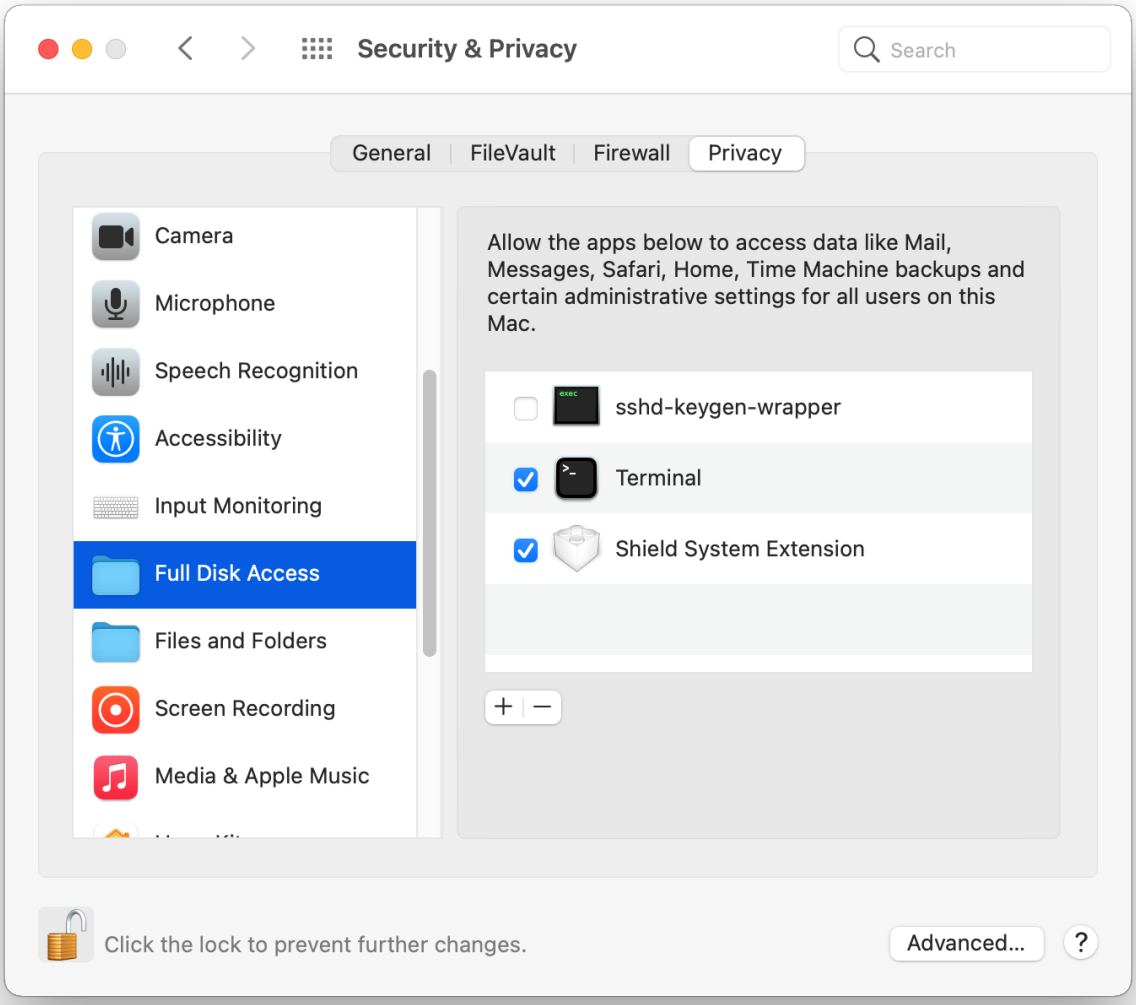
As we can see, beyond `.zshrc` we have many other files, like `.zlogin`, `.zshenv` and a few others. Normally these files don't exists on macOS.

Similar startup files exists for `bash` and `sh`. The latter supports far less such files, checking the man page it's very short.

```
$HOME/.profile

/etc/profile
```

The benefit of these scripts, especially `.zshrc` which is used as the default shell environment, is that when Terminal is started, and it starts the shell, it will inherit Terminal's privacy permissions.



Many power users will have “Full Disk Access” rights granted to Terminal (as shown above) and as such, it will be able to access many privacy sensitive locations. So if we place any script inside these files, it will have powerful rights. For example:

```
curl google.com
cp -R ~/Library/Messages /tmp/
```

This script will do a network connection and copy `Messages` to `/tmp/`.

It will result in accessing messages as shown above.

Beyond that Terminal runs outside the sandbox, thus if we can place anything inside these script files, we can escape the sandbox.

Another related command to watch for, is `chsh`, which can change the shell environment for a user.

```
chsh -s /bin/bash
```

As defenders it's a good practice to monitor for the occurrence of these files, and if they exists, then analyze their content.