Security of Electron-based desktop applications

By Alanna Titterington

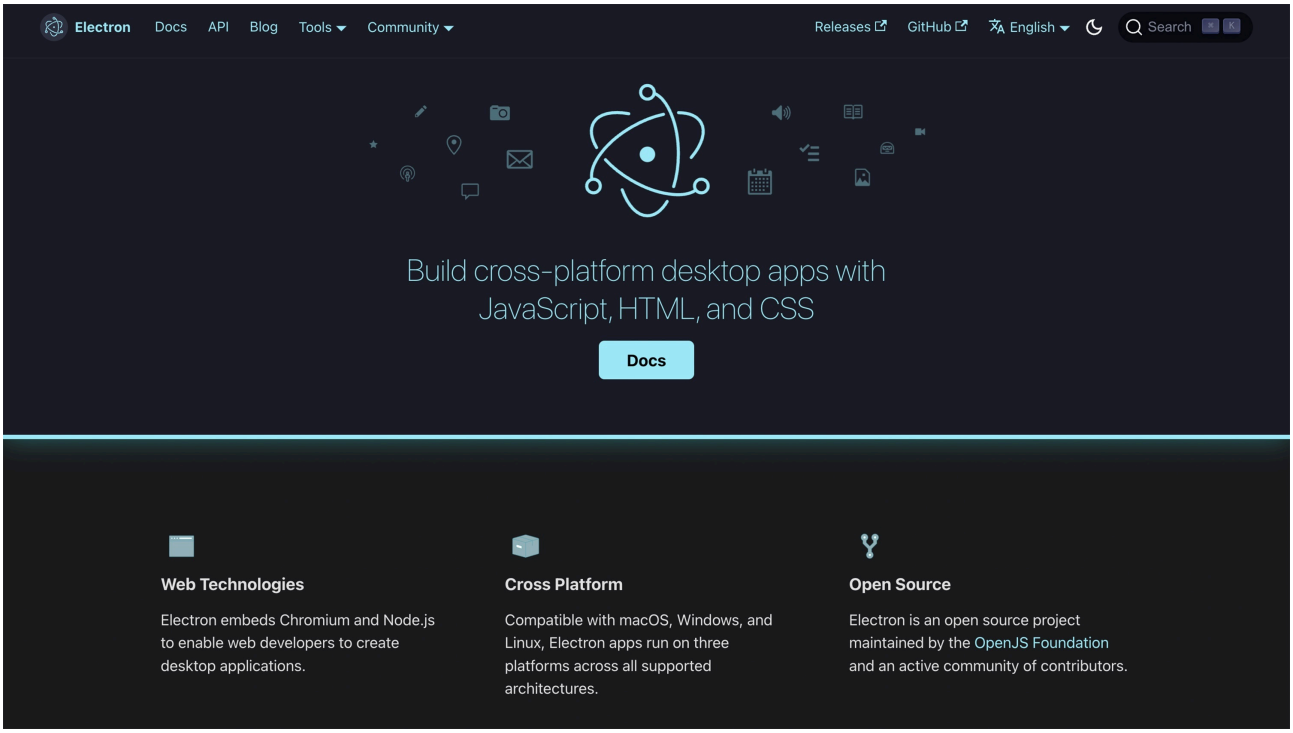
Published: 2023-09-14 · Archived: 2026-04-05 15:25:04 UTC

Early this year I gave you five reasons to avoid desktop versions of messengers. The fact that many such applications use the Electron framework is one of them. This means that such a messenger works as an additional browser in your system, and its updates are quite difficult to control.

But, as I wrote in that post, it has become clear the problem is much more widespread — affecting not only messengers but hundreds of other apps as well. Chances are, because of Electron-based apps, you have a many more browsers than you think in your system this very minute…

What is Electron, and why do application developers want to use it?

Electron is a cross-platform desktop application development framework that employs web technologies — mostly HTML, CSS, and JavaScript. It was originally created by GitHub for its source code editor Atom (hence its original name — Atom Shell). Later on the framework was renamed Electron, ultimately evolving into an extremely popular tool used to create desktop applications for various operating systems, including Windows, macOS, and Linux.



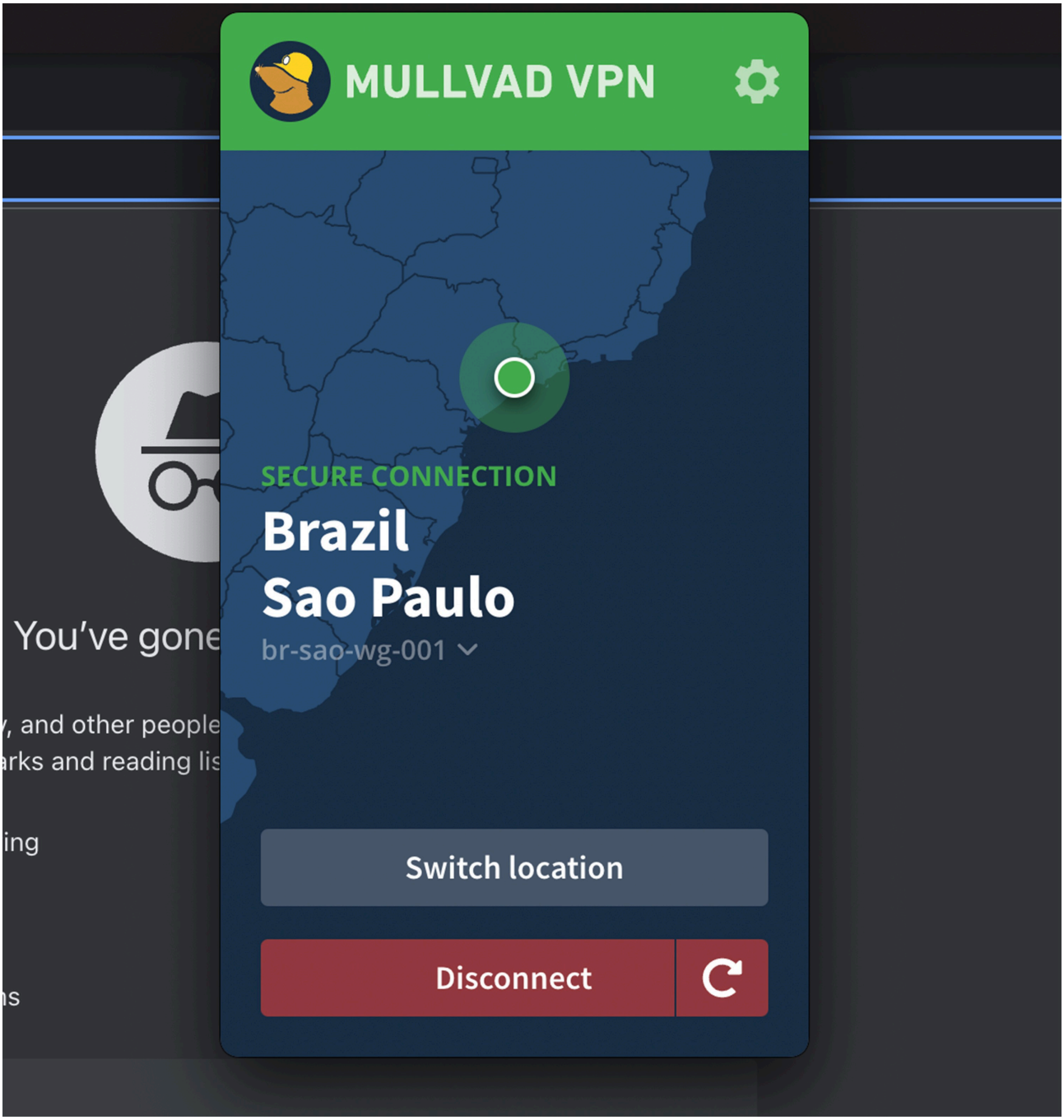
Main page of the Electron framework official site. Source

Electron itself is based on the Chromium browser engine, which is responsible for displaying web content within a desktop application. So any Electron application is effectively a single website opened in the Chromium browser.

Users usually have no idea at all how the thing works. From their point of view, an Electron application is just another program you install, run in the usual way, give access to some files, occasionally update to the newest version, and so on.

Why has Electron grown so popular with developers? The idea is mainly this: no matter what digital service one might want to create, a web version is still needed. And the Electron framework allows you to develop just the web version and, based on it, produce full-fledged apps for all the desktop operating systems out there.

Electron's other convenience features include making installation packages, their diagnostics, publication to app stores, and automatic updates.



Et tu autem, Brute! You can find Electron in apps you least expect to

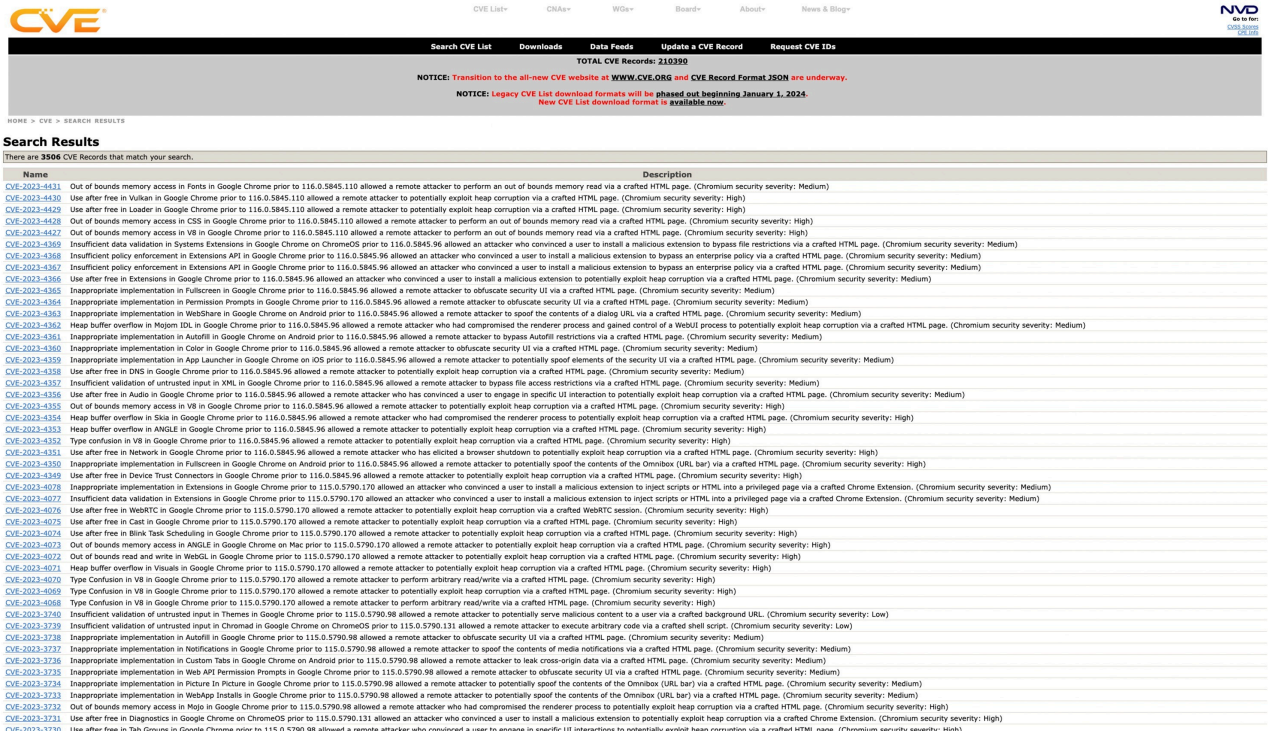
Summing up, the Electron framework is popular among developers — most particularly as it allows to greatly accelerate and simplify the application development process for all desktop operating systems in one go.

Issues with Electron-based applications

Electron-based applications have a number of drawbacks. The most obvious from the users' perspective is their sluggishness. Electron-based software is usually resource-intensive and suffers from excessive file size. No wonder: each such app carries ~~its whole home on its back like a snail~~ a full-blown Chromium browser. In effect, it operates through that browser — serving as a sort of intermedium.

Next issue: web browsers are a favorite target of cybercriminals. It's worth repeating: inside *every* Electron-based app there's a **separate instance of the Chromium** web browser. This means your system may have a dozen additional browsers installed, all of which present a tempting target for criminals.

New, serious vulnerabilities pop up almost weekly in a popular browser like Chrome/Chromium: so far this year more than 70 high, and three critical severity-level vulnerabilities have been found in Chromium as of the time of writing. Worse yet, exploits for the world's most popular browser's vulnerabilities appear really quick. This means that a good part of Chrome/Chromium holes are not just abstract bugs you treat as a matter of routine — they're vulnerabilities that can be used for attacks by cybercriminals out in the wild.



CVE CVE List CNAs WGs Board About News & Blog NVD

Search CVE List | Downloads | Data Feeds | Update a CVE Record | Request CVE IDs

TOTAL CVE Records: 210390

NOTICE: Transition to the all-new CVE website at WWW.CVE.ORG and CVE Record Format JSON are underway.

NOTICE: Legacy CVE List download formats will be phased out beginning January 1, 2024.
New CVE List download format is available now.

HOME > CVE > SEARCH RESULTS

Search Results

There are **3506** CVE Records that match your search.

Name	Description
CVE-2023-4431	Out of bounds memory access in Fonts in Google Chrome prior to 116.0.5845.110 allowed a remote attacker to perform an out of bounds memory read via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-4430	Use after free in Vulkan in Google Chrome prior to 116.0.5845.110 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4429	Use after free in Loader in Google Chrome prior to 116.0.5845.110 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4428	Out of bounds memory access in CSS in Google Chrome prior to 116.0.5845.110 allowed a remote attacker to perform an out of bounds memory read via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4427	Out of bounds memory access in V8 in Google Chrome prior to 116.0.5845.110 allowed a remote attacker to perform an out of bounds memory read via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4369	Insufficient data validation in Systems Extensions in Google Chrome on ChromeOS prior to 116.0.5845.96 allowed an attacker who convinced a user to install a malicious extension to bypass file restrictions via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-4368	Insufficient policy enforcement in Extensions API in Google Chrome prior to 116.0.5845.96 allowed an attacker who convinced a user to install a malicious extension to bypass an enterprise policy via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-4367	Insufficient policy enforcement in Extensions API in Google Chrome prior to 116.0.5845.96 allowed an attacker who convinced a user to install a malicious extension to bypass an enterprise policy via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-4366	Use after free in Extensions in Google Chrome prior to 116.0.5845.96 allowed an attacker who convinced a user to install a malicious extension to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-4365	Inappropriate implementation in Fullscreen in Google Chrome prior to 116.0.5845.96 allowed a remote attacker to obfuscate security UI via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-4364	Inappropriate implementation in Permission Prompts in Google Chrome prior to 116.0.5845.96 allowed a remote attacker to obfuscate security UI via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-4363	Inappropriate implementation in WebShare in Google Chrome on Android prior to 116.0.5845.96 allowed a remote attacker to spoof the contents of a dialog URL via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-4362	Heap buffer overflow in Mojom IDL in Google Chrome prior to 116.0.5845.96 allowed a remote attacker who had compromised the renderer process and gained control of a WebUI process to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-4361	Inappropriate implementation in Autofill in Google Chrome on Android prior to 116.0.5845.96 allowed a remote attacker to bypass Autofill restrictions via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-4360	Inappropriate implementation in Color in Google Chrome prior to 116.0.5845.96 allowed a remote attacker to obfuscate security UI via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-4359	Inappropriate implementation in App Launcher in Google Chrome on iOS prior to 116.0.5845.96 allowed a remote attacker to potentially spoof elements of the security UI via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-4358	Use after free in DNS in Google Chrome prior to 116.0.5845.96 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-4357	Insufficient validation of untrusted input in XML in Google Chrome prior to 116.0.5845.96 allowed a remote attacker to bypass file access restrictions via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-4356	Use after free in Audio in Google Chrome prior to 116.0.5845.96 allowed a remote attacker who has convinced a user to engage in specific UI interaction to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-4355	Out of bounds memory access in V8 in Google Chrome prior to 116.0.5845.96 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4354	Heap buffer overflow in Skia in Google Chrome prior to 116.0.5845.96 allowed a remote attacker who had compromised the renderer process to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4353	Heap buffer overflow in ANGLE in Google Chrome prior to 116.0.5845.96 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4352	Type confusion in V8 in Google Chrome prior to 116.0.5845.96 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4351	Use after free in Network in Google Chrome prior to 116.0.5845.96 allowed a remote attacker who has elicited a browser shutdown to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4350	Inappropriate implementation in Fullscreen in Google Chrome on Android prior to 116.0.5845.96 allowed a remote attacker to potentially spoof the contents of the Omnibox (URL bar) via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4349	Use after free in Device Trust Connectors in Google Chrome prior to 116.0.5845.96 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4078	Inappropriate implementation in Extensions in Google Chrome prior to 115.0.5790.170 allowed an attacker who convinced a user to install a malicious extension to inject scripts or HTML into a privileged page via a crafted Chrome Extension. (Chromium security severity: Medium)
CVE-2023-4077	Insufficient data validation in Extensions in Google Chrome prior to 115.0.5790.170 allowed an attacker who convinced a user to install a malicious extension to inject scripts or HTML into a privileged page via a crafted Chrome Extension. (Chromium security severity: Medium)
CVE-2023-4076	Use after free in WebRTC in Google Chrome prior to 115.0.5790.170 allowed a remote attacker to potentially exploit heap corruption via a crafted WebRTC session. (Chromium security severity: High)
CVE-2023-4075	Use after free in Cast in Google Chrome prior to 115.0.5790.170 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4074	Use after free in Blink Task Scheduling in Google Chrome prior to 115.0.5790.170 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4073	Out of bounds memory access in ANGLE in Google Chrome on Mac prior to 115.0.5790.170 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4072	Out of bounds read and write in WebGL in Google Chrome prior to 115.0.5790.170 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4071	Heap buffer overflow in Visuals in Google Chrome prior to 115.0.5790.170 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4070	Type Confusion in V8 in Google Chrome prior to 115.0.5790.170 allowed a remote attacker to perform arbitrary read/write via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4069	Type Confusion in V8 in Google Chrome prior to 115.0.5790.170 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)
CVE-2023-4068	Type Confusion in V8 in Google Chrome prior to 115.0.5790.170 allowed a remote attacker to perform arbitrary read/write via a crafted HTML page. (Chromium security severity: High)
CVE-2023-3740	Insufficient validation of untrusted input in Themes in Google Chrome prior to 115.0.5790.98 allowed a remote attacker to potentially serve malicious content to a user via a crafted background URL. (Chromium security severity: Low)
CVE-2023-3739	Insufficient validation of untrusted input in Chromad in Google Chrome on ChromeOS prior to 115.0.5790.131 allowed a remote attacker to execute arbitrary code via a crafted shell script. (Chromium security severity: Low)
CVE-2023-3738	Inappropriate implementation in Autofill in Google Chrome prior to 115.0.5790.98 allowed a remote attacker to obfuscate security UI via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-3737	Inappropriate implementation in Notifications in Google Chrome prior to 115.0.5790.98 allowed a remote attacker to spoof the contents of media notifications via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-3736	Inappropriate implementation in Custom Tabs in Google Chrome on Android prior to 115.0.5790.98 allowed a remote attacker to leak cross-origin data via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-3735	Inappropriate implementation in Web API Permission Prompts in Google Chrome prior to 115.0.5790.98 allowed a remote attacker to obfuscate security UI via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-3734	Inappropriate implementation in Picture In Picture in Google Chrome prior to 115.0.5790.98 allowed a remote attacker to potentially spoof the contents of the Omnibox (URL bar) via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-3733	Inappropriate implementation in WebApp Installs in Google Chrome prior to 115.0.5790.98 allowed a remote attacker to potentially spoof the contents of the Omnibox (URL bar) via a crafted HTML page. (Chromium security severity: Medium)
CVE-2023-3732	Out of bounds memory access in Mojo in Google Chrome prior to 115.0.5790.98 allowed a remote attacker who had compromised the renderer process to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)
CVE-2023-3731	Use after free in Diagnostics in Google Chrome on ChromeOS prior to 115.0.5790.131 allowed an attacker who convinced a user to install a malicious extension to potentially exploit heap corruption via a crafted Chrome Extension. (Chromium security severity: High)
CVE-2023-3730	Use after free in Tab Groups in Google Chrome prior to 115.0.5790.98 allowed a remote attacker who convinced a user to engage in specific UI interactions to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)

Even in fine print, Chromium vulnerabilities found so far in 2023 take up several screens. Source

For the standalone Chrome browser, this isn't such a serious problem. Google is *very* quick to release patches and rather persistent in convincing users to install them and restart their browser (it even thoughtfully re-opens all their precious tabs after restarting so they don't need to fear updating).

Things are very different for the Electron-based apps. A Chromium browser built into such an app will only get patched if the app's vendor has released a new version and successfully communicated to users the need to install it.

So it appears that, with a bunch of installed Electron apps, not only do you have multiple browsers installed on your system, but also little to no control over how updated and secure those browsers are, or how many unpatched vulnerabilities they contain.

The framework's creators know full well about the problem, and strongly recommend that app developers release patches on time. Alas, users can only hope that those recommendations are followed.

And here's a fresh example: On September 11, Google fixed the CVE-2023-4863 vulnerability in Google Chrome. At that point, it was already actively exploited in the wild. It allows a remote attacker to perform an out of bounds memory write via a crafted HTML page, which can lead to the execution of arbitrary code. Of course, this bug is present in Chromium and all Electron-based applications. So, all companies using it in their applications will have to work on updates.

Which desktop applications are based on Electron?

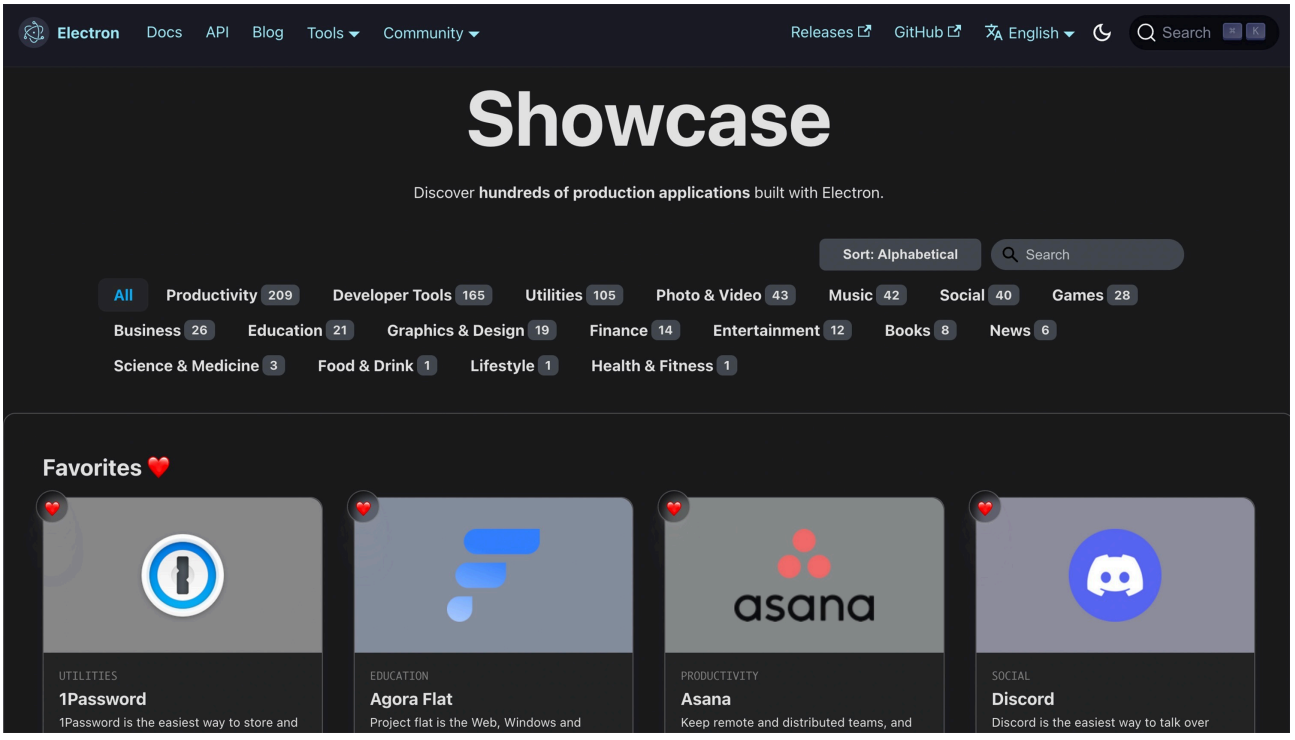
Not many folks seem to know how incredibly common Electron-based desktop applications are. I'll bet you are using more than one of them. Check them out yourself:

- 1Password
- Agora Flat
- Asana
- Discord
- Figma
- GitHub Desktop
- Hyper
- Loom
- Microsoft Teams
- Notion
- Obsidian
- Polyplane
- Postman
- Signal
- Skype
- Slack
- Splice
- Tidal
- Trello
- Twitch
- Visual Studio Code
- WhatsApp

- WordPress Desktop

I personally use around a third of the apps from the list (but, for the record, none of them as desktop applications).

That list is not exhaustive at all though, representing only the most popular Electron-based applications. In total there are several hundred such applications. A more or less complete list of them can be found on a special page on the official website of the framework (but, it seems, not all of them are listed even there).



The list of Electron-based desktop applications comprises several hundred online services, including about 20 really popular ones. Source

Security considerations

So how to avoid the threats posed by uncontrolled browsers that thoughtful developers are now unpredictably embedding into desktop apps? I have three main tips regarding this:

- Minimize the number of Electron-based apps as much as possible. It's not as difficult as it seems: the very fact of using the framework normally suggests that the service has an extremely advanced web version, which is most likely on a par with the desktop application in terms of features and convenience.
- Try to inventory all Electron-based apps used by your company's employees, and prioritize their updates. More often than not, these are collaboration applications of different forms and shades — from Microsoft Teams, Slack, and Asana, to GitHub and Figma.
- Use a reliable security solution. It will help you repel attacks in those periods when vulnerabilities are already known and being exploited but the patches haven't yet been issued. By the way, Kaspersky products have an exploit protection system: it helps our experts detect the exploitation of new, as yet unknown vulnerabilities, and warns the developers of the corresponding programs about these holes.

Source: https://www.kaspersky.com/blog/electron-framework-security-issues/49035/