33RD USENIX SECURITY SYMPOSIUM

Rise of Inspectron

Automated Black-box Auditing of Cross-platform Electron Apps

Mir Masood Ali, Mohammad Ghasemisharif, Chris Kanich, and Jason Polakis

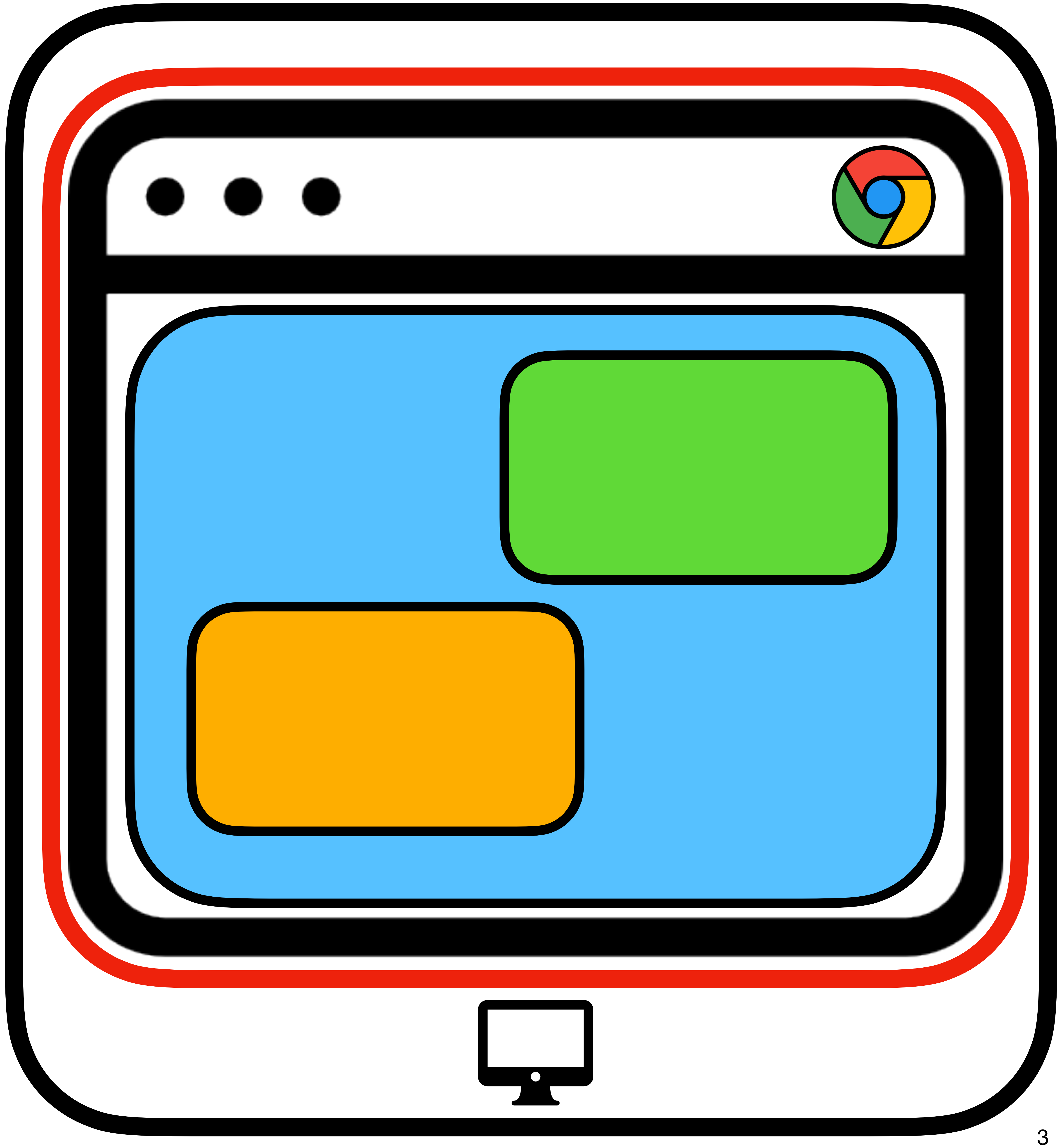
UIC

Paper in a Slide

- **Web browsers** have come a long way in adapting to new threats and ensuring user security
- **Website developers** are increasingly adopting cross-platform frameworks to provide native features
- **Electron** helps developers port their websites to cross-platform desktop apps — a complex and error-prone process
- **Inspectron** is an automated, dynamic analysis framework that audits Electron apps for security and privacy vulnerabilities
- **Our analysis** resulted in 106 reports, including to Postman and WordPress, and resulted in the resolution of a high-severity CVE

Web Browsers

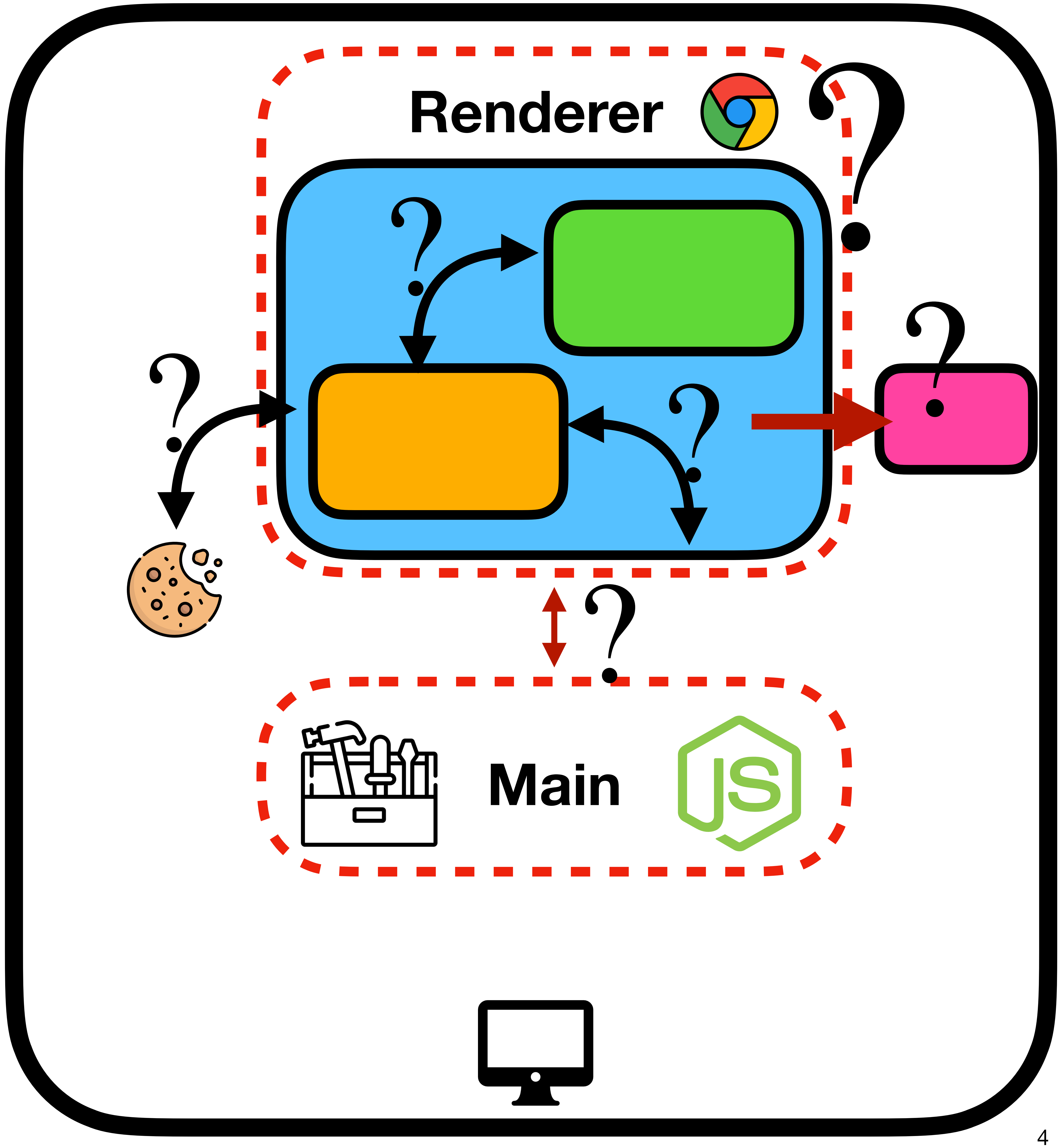
- Users visit websites to access services
- Websites load external content
- Web browsers handle rendering and manage loaded content
- Web browsers add safeguards between loaded content and the underlying device



Electron

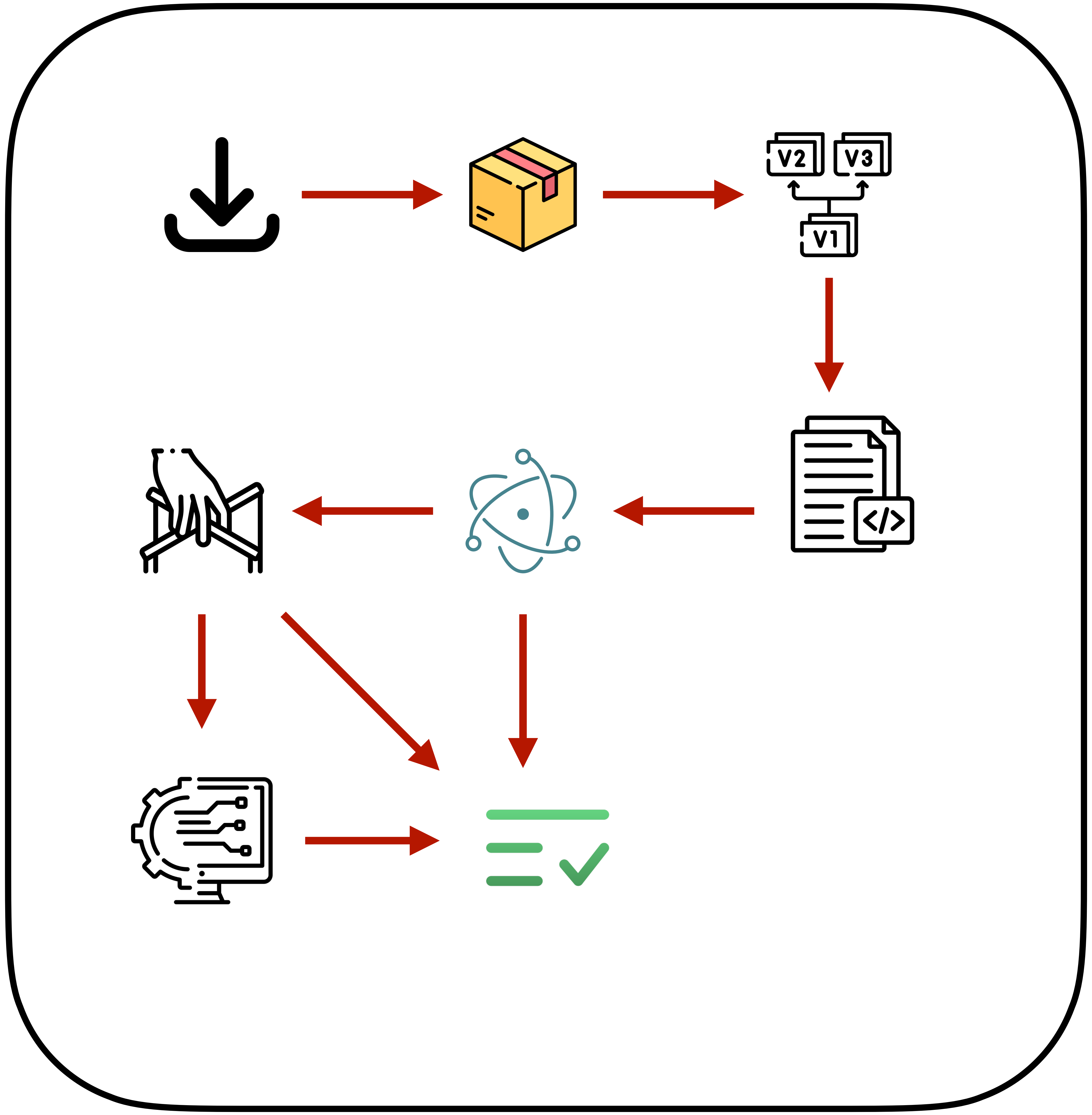
Complex. Tricky. Error-prone

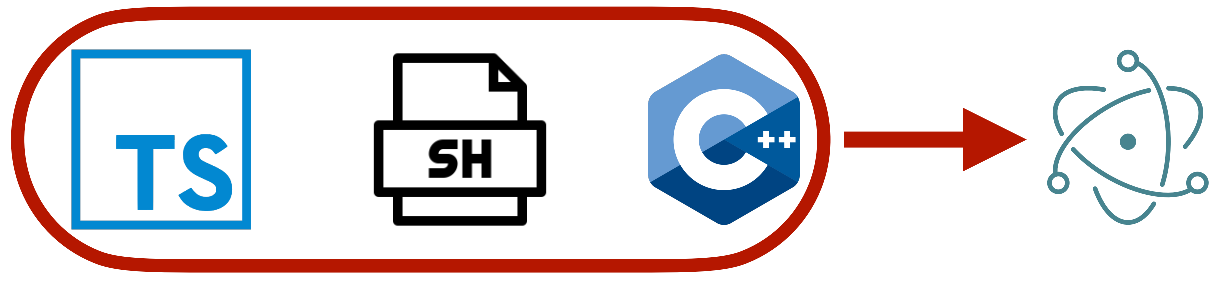
- Rendered content re-uses web browser engine
- Developers configure native features within a privileged process
- Developers answer:
  - How do we isolate rendered content?
  - How do the two processes talk to each other?
  - How do we retain the Same Origin Policy (SoP)?
  - How do we restrict navigation?
  - How do we secure cookies and sensitive tokens?



Inspectron: Evaluating Apps

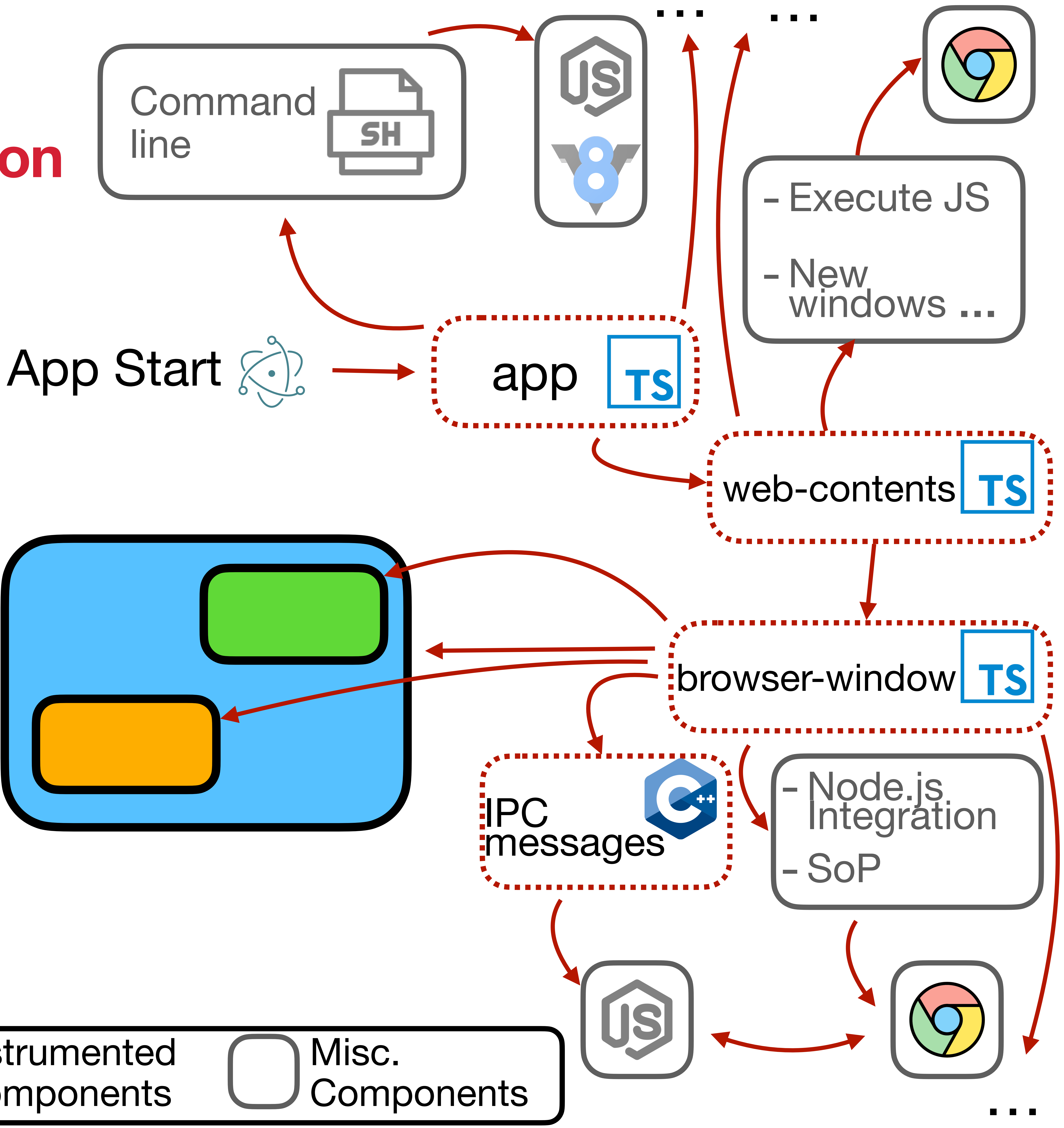
- We downloaded each app and extracted packed files
- We evaluated the app to determine its underlying library versions
- We extracted the source code and executed the app within our instrumented Electron
- We evaluated the app's loaded content using Puppeteer
- We also performed additional checks outside of the two automated libraries
- We combined results from multiple sources into an extensive report





Inspectron: Instrumenting Electron

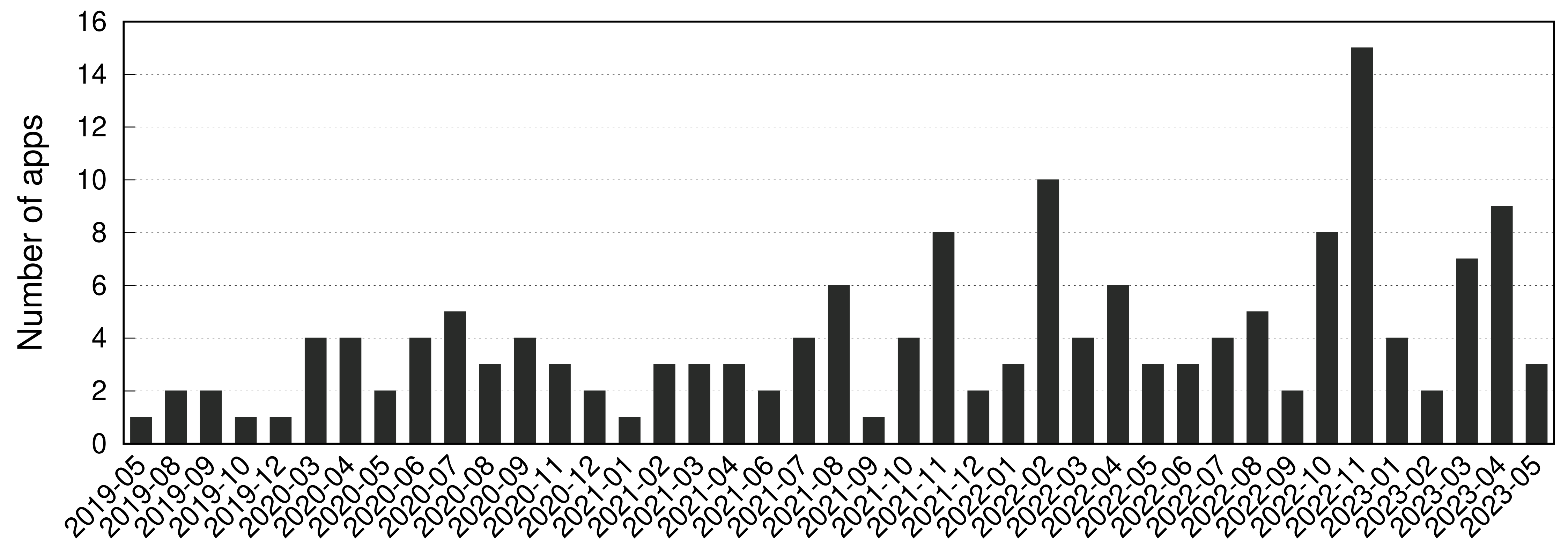
- Electron's source is spread across multiple languages and directory paths
- When an app starts, Electron reads and passes options to underlying libraries
- New Window
  - Should the window be allowed to open new windows?
  - Should the window have access to Node.js libraries?
  - Should the window be allowed to send messages to other app processes?
- Repeat the process for each window



Evaluation: Versions

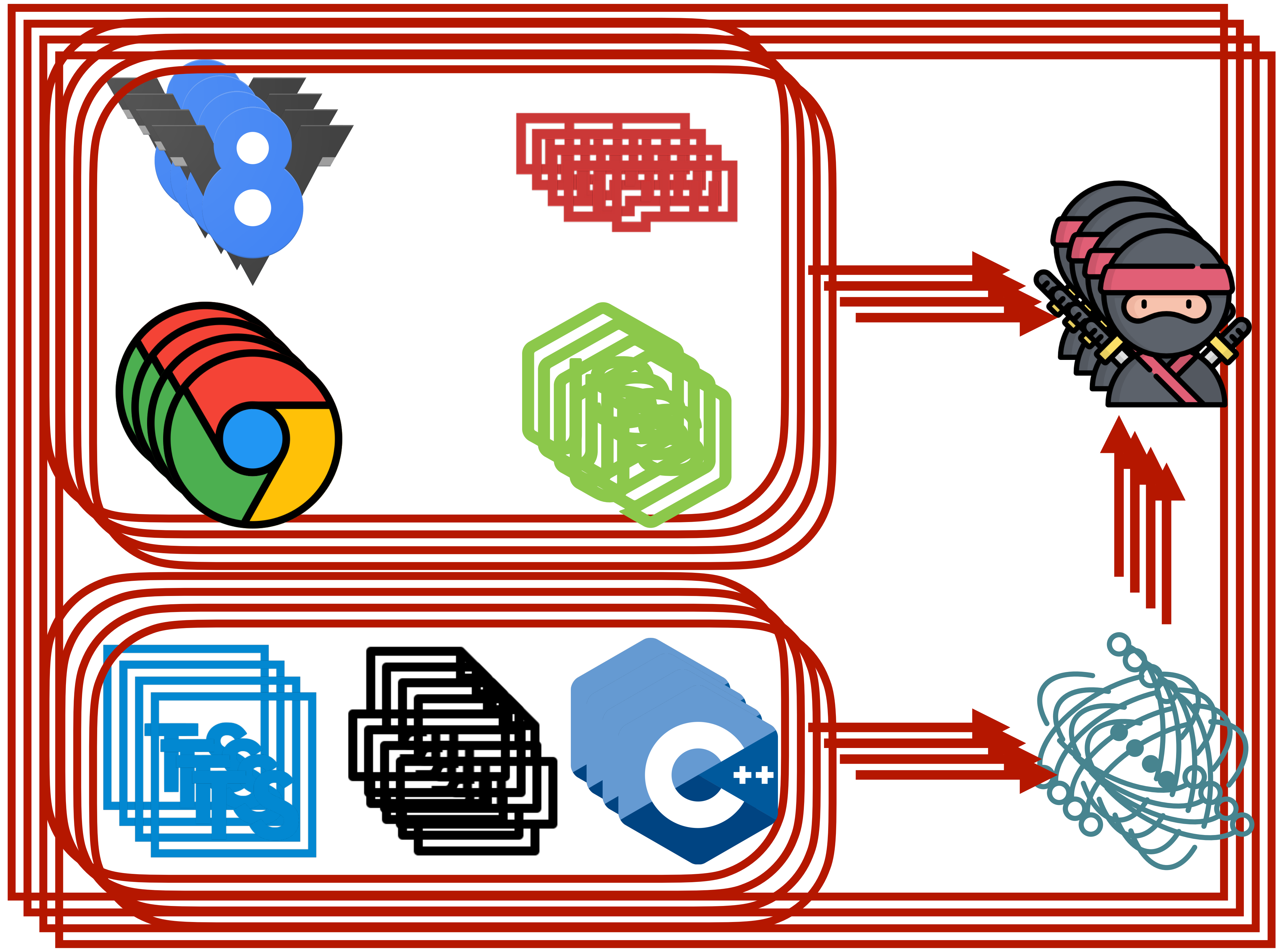
- We evaluated 109 apps with their latest versions as of May 2023

- Electron Versions go back **4** years to May 2019!



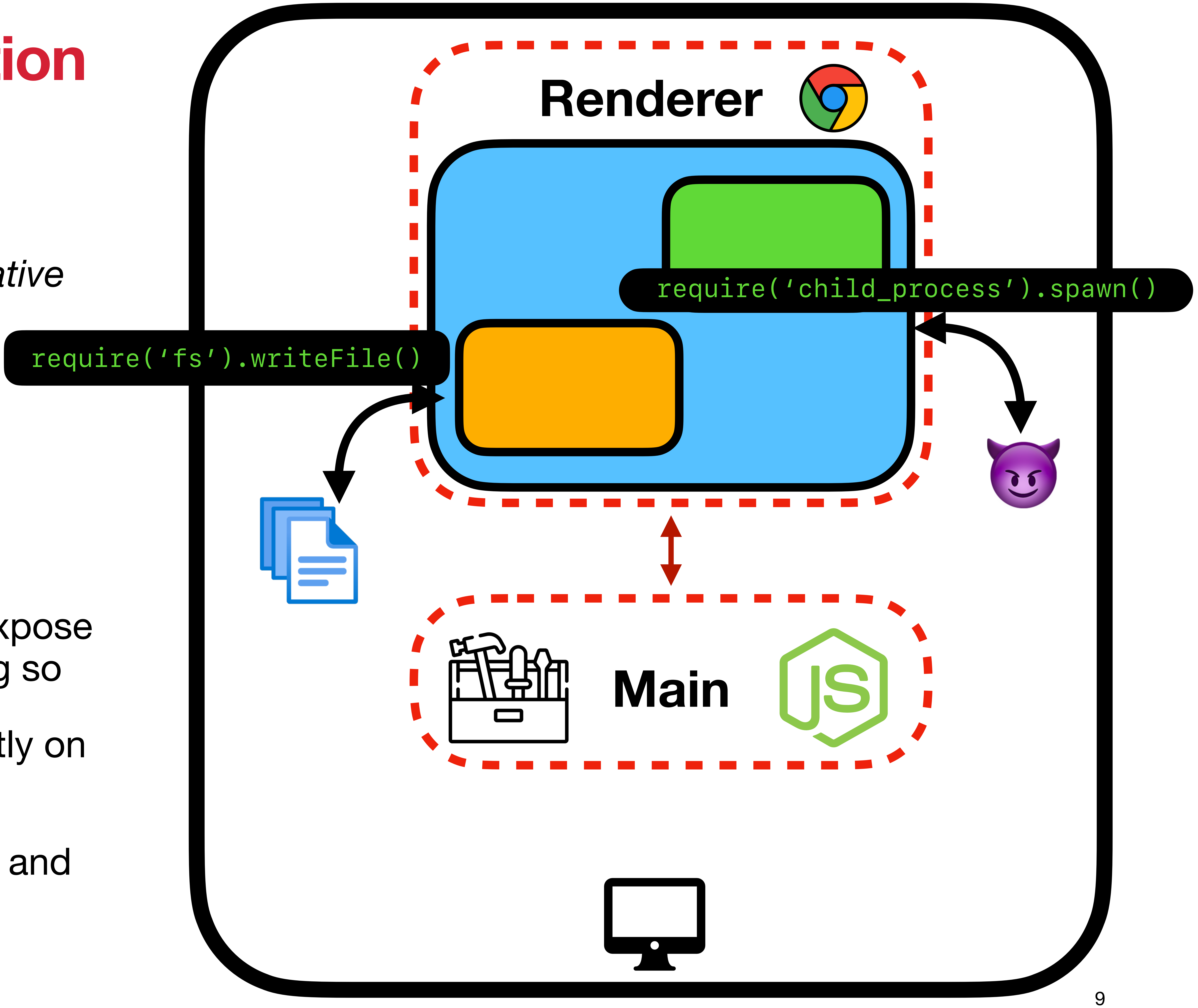
Evaluation: Versions (continued)

- Electron re-packages underlying libraries, including NPM and Chrome
- Apps built on a particular version of Electron are tied to that specific version
- ***Our Solution:*** Build an instrumented Electron for each framework version
- We instrumented 24 versions of Electron across MacOS and Linux builds



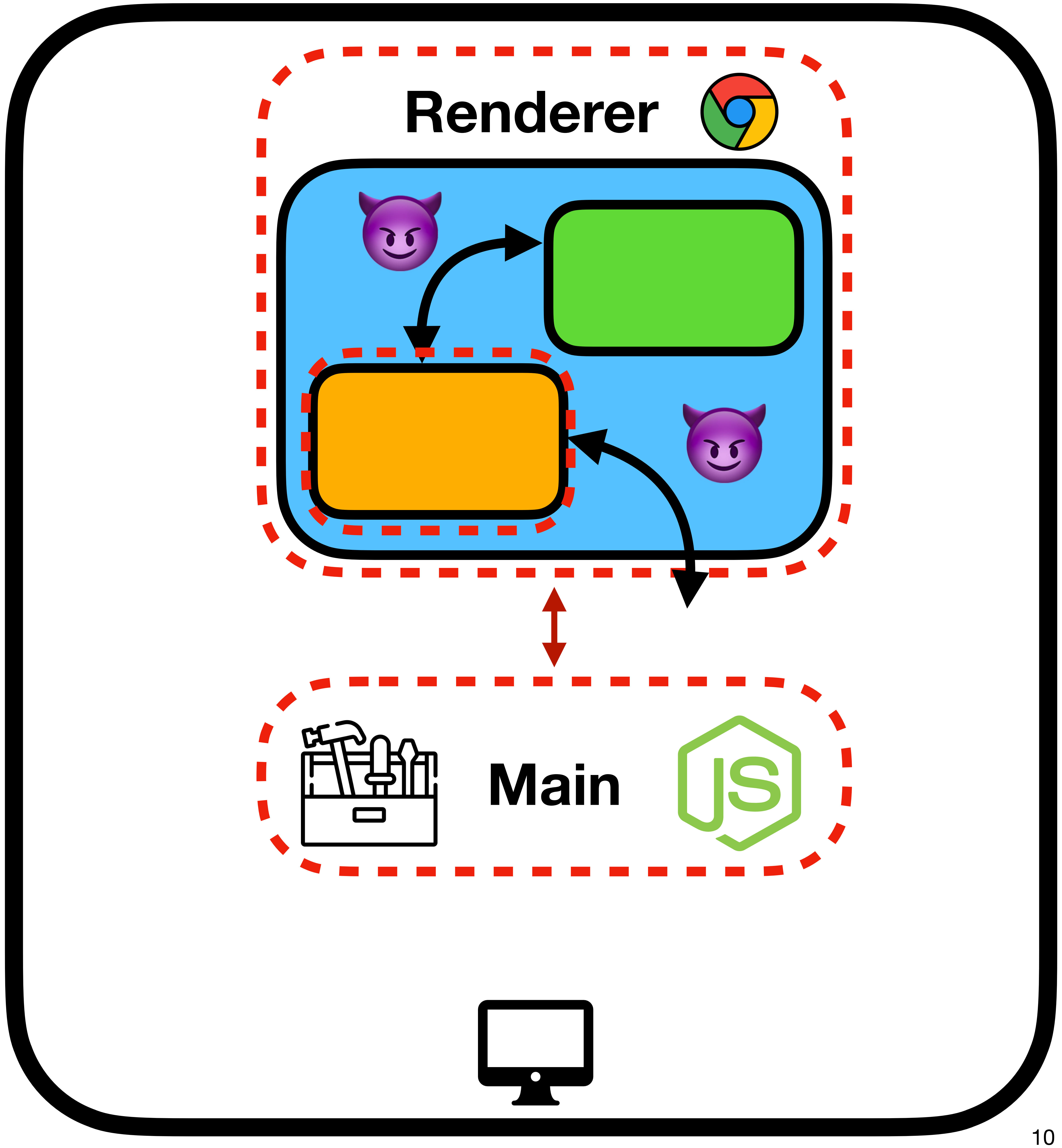
Evaluation: Node Integration

- Electron allows developers to provide *native* features
- They advise developers to use the main process primarily
- Developers can selectively expose functionality to the renderer
- Some developers may find it easier to expose all of Node.js — we found **49** apps doing so
- Loaded content can execute code directly on the system
- External, third-party content can access and edit the filesystem



Evaluation: Web Security

- The Same-origin Policy is a critical security mechanism on the web
- Electron allows developers to relax the isolation between origins
- External, third-party components can misuse this relaxation to bypass restrictions
- We found **8** apps that completely disabled web security
- The feature removes restrictions across third-party origins
- Even with other restrictions in place, third-party scripts can can execute in privileged contexts

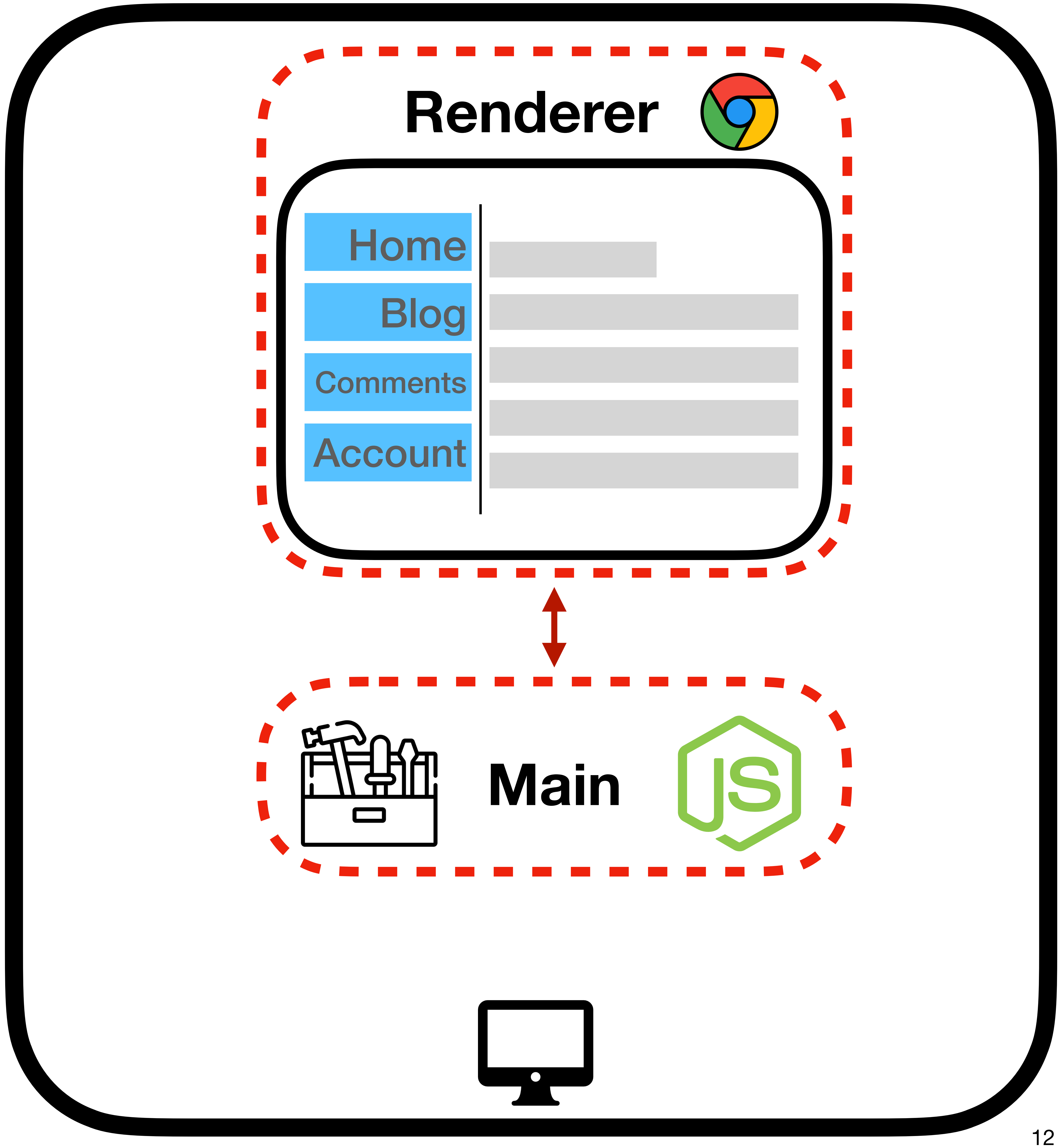


Evaluation: Overview

- Node Integration and Web Security make two of **12** *web preferences* — one of the classes of misconfigurations
- We used Inspectron to evaluate apps across **16** classes of misconfigurations
- Once built, Inspectron can automatically identify misconfigurations within apps
- We evaluated reports gathered from **109** apps

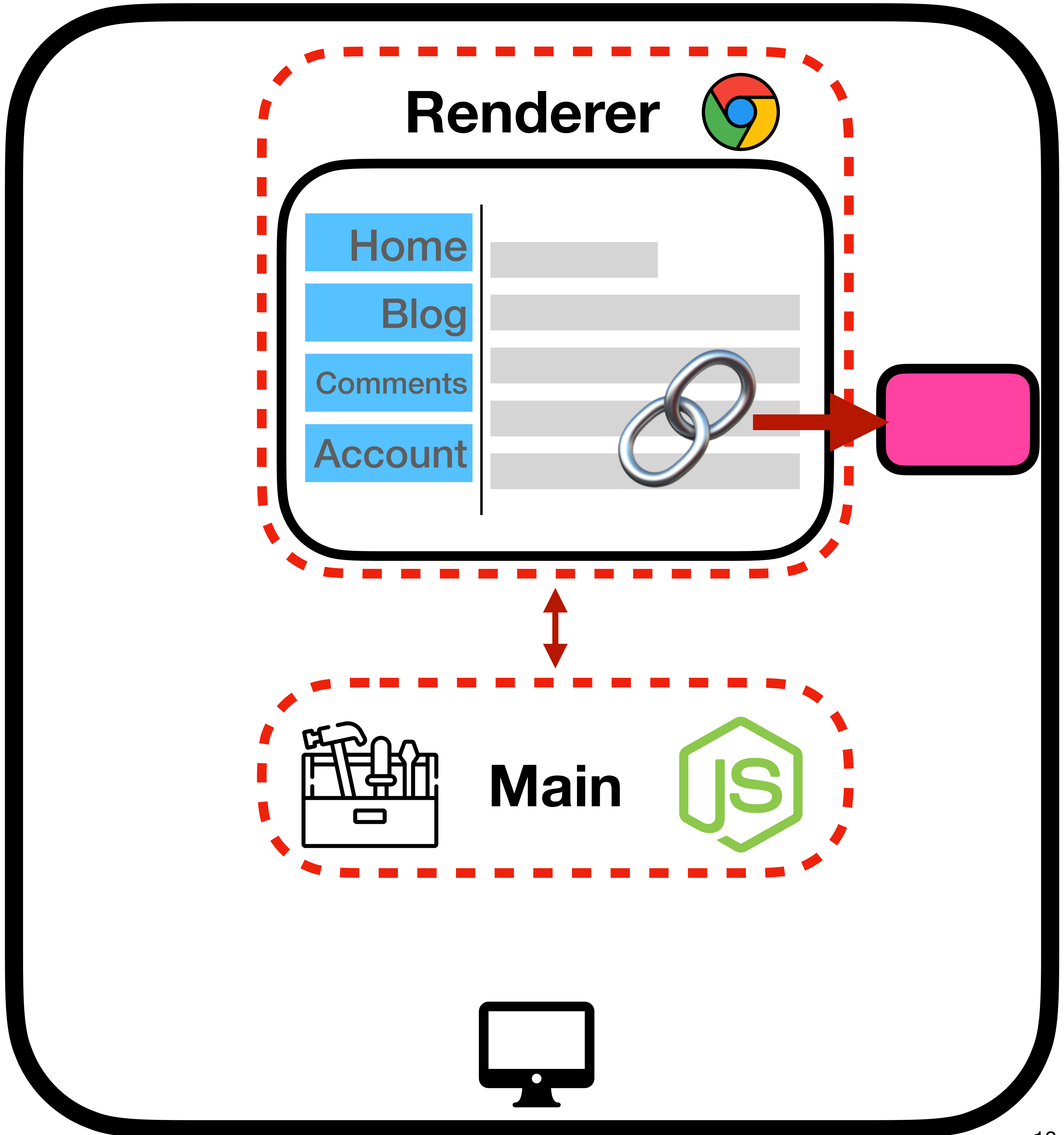
Evaluation: WordPress

- WordPress offers users a desktop app to access its services
- Users build and manage websites, write blog posts, engage with comments
- WordPress developers reuse website code within their Electron app



Evaluation: WordPress (continued)

- We noticed two important misconfigurations
- ***Chrome Version****:* As of May 2023, WordPress was using Electron v12, with Chrome v89 — **>3** years old
- ***Navigation:*** WordPress did not block loading third-party links within the app. Inspectron checks for restrictions.

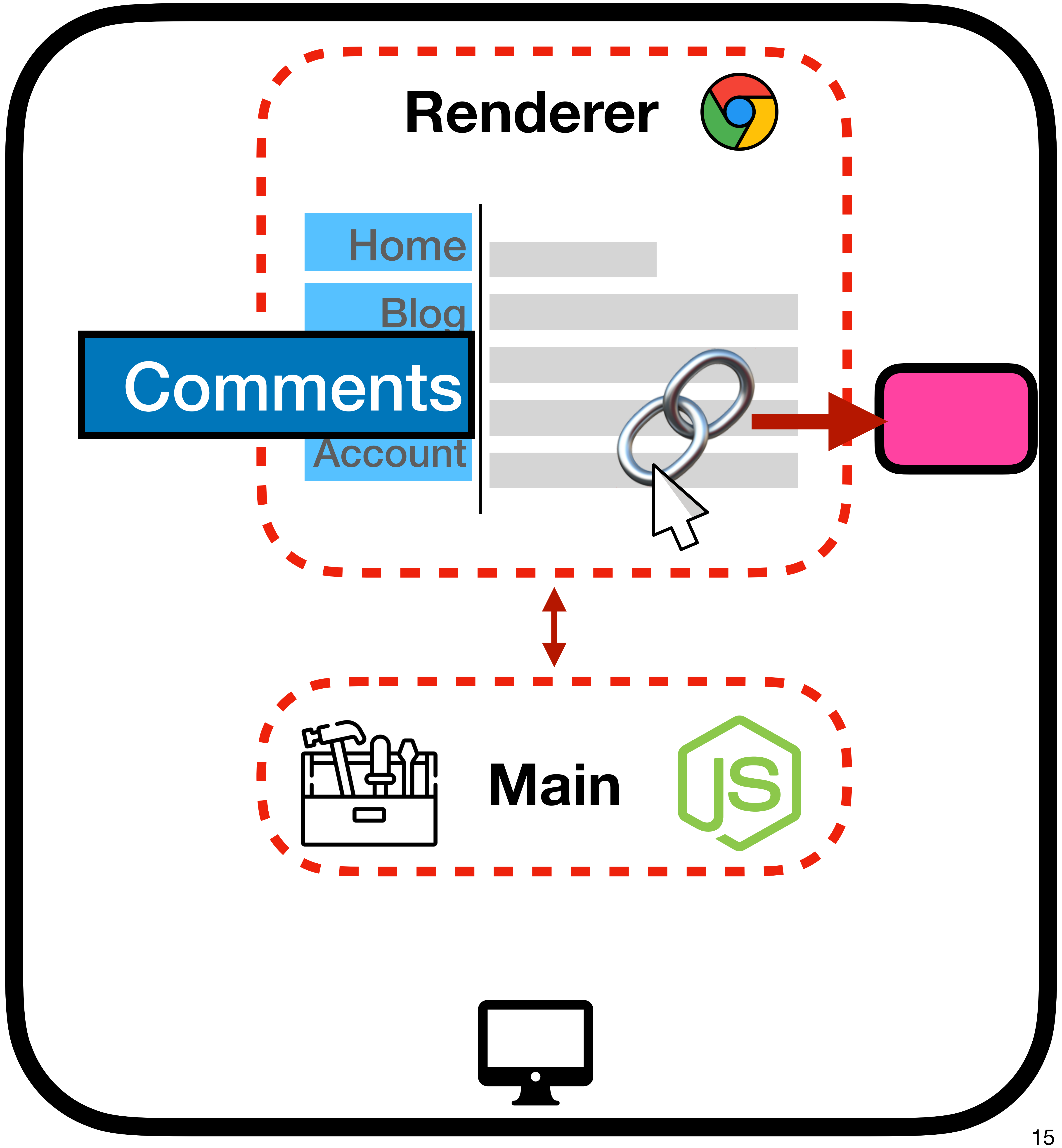


Evaluation: WordPress (continued)

- Elsewhere, an attacker visits the victim's blog
- The attacker adds a comment on the victim's blog post
- The attacker embeds a link in the comment

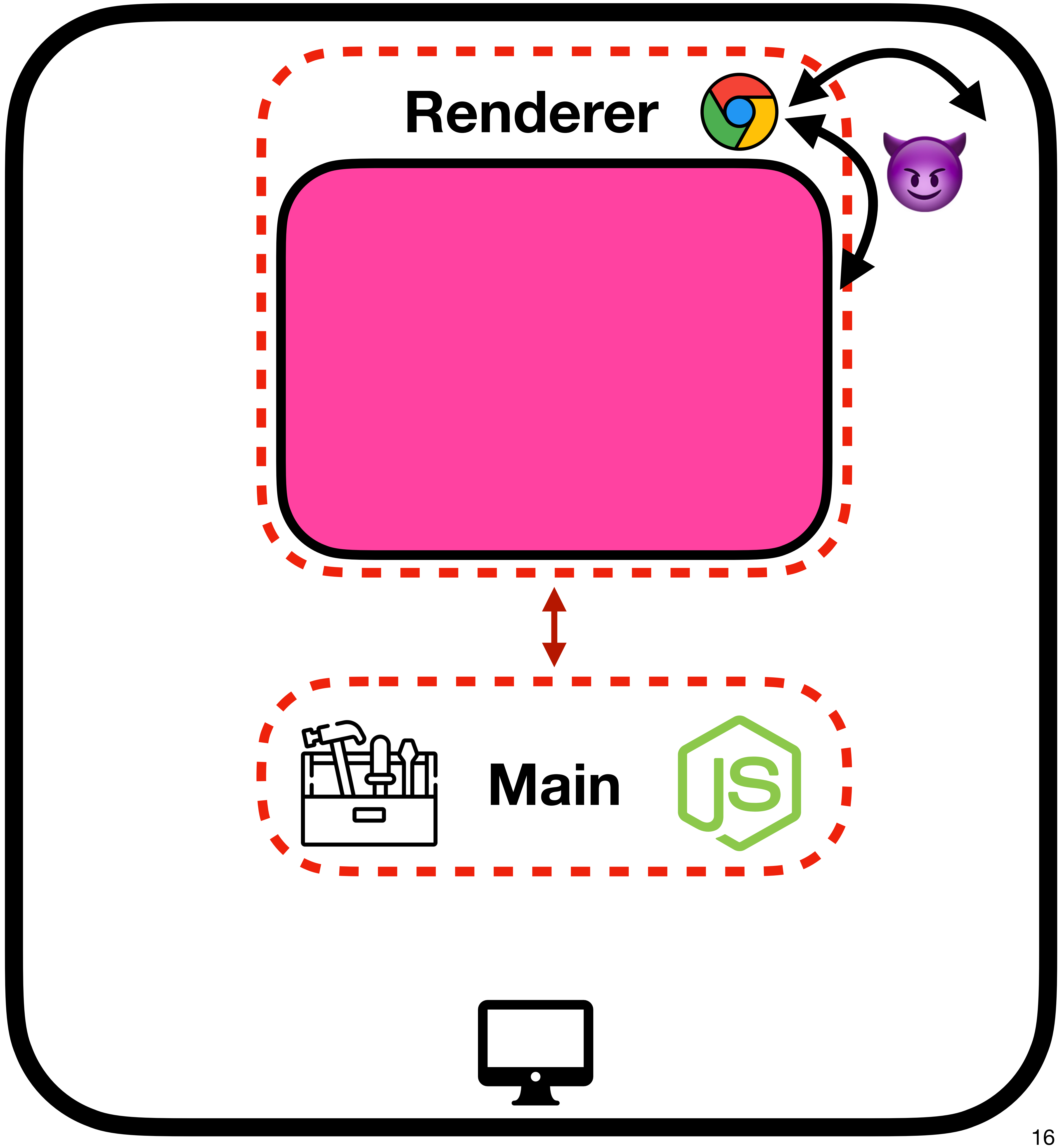
Evaluation: WordPress (continued)

- The victim checks comments on their blog
- The victim clicks on the link within the attackers comment



Evaluation: WordPress (continued)

- The attacker now runs in the window meant for the Wordpress app
- The attacker's site is loaded in an old version of Chrome
- The attacker can use existing exploits on V8 and Blink to execute malicious code, and compromise the user's system



Conclusion

- Electron provides numerous convenient features but also creates new paths for vulnerabilities
- We built Inspectron to automatically find and report on these vulnerabilities within Electron apps
- Analysing apps using Inspectron resulted in the submission of **106** reports. We also found a high-severity CVE and were rewarded by three app developers.

Artifact

mali92@uic.edu

33RD USENIX SECURITY SYMPOSIUM

Rise of Inspectron

Automated Black-box Auditing of Cross-platform Electron Apps

Mir Masood Ali, Mohammad Ghasemisharif, Chris Kanich, and Jason Polakis

UIC