JavaScript Restrictor

Webextension that improves your privacy and security

User guide

Libor Polčák

TARZAN project VI20172020062 document
Faculty of Information Technology, Brno University of Technology

Last change: June 19, 2020

JavaScript Restrictor

A JS-enabled web page can access any of the APIs that a web browser provides. The user has only a limited control and some APIs cannot be restricted by the user easily. JavaScript Restrictor aims to improve the user control of the web browser. Similarly to a firewall that controls the network traffic, JavaScript Restrictor controls the APIs provided by the browser. The goal is to improve the privacy and security of the user running the extension.

JavaScript Restrictor (JSR) is a browser extension with support for multiple browsers: Firefox, Google Chrome, and Opera. The extension also works with Brave, Microsoft Edge, and most likely any Chromium-based browser.

Various websites collect information about users without their awareness. The collected information is used to track users. Malicious websites can fingerprint user browsers or computers. JavaScript Restrictor protects the user by restricting or modifying several web browser APIs used to create side-channels and identify the user, the browser or the computer. JavaScript Restrictor can block access to JavaScript objects, functions and properties or provide a less precise implementation of their functionality, for example, by modifying or spoofing values returned by the JS calls. The goal is to mislead websites by providing false data or no data at all.

Another goal of the extension is not to break the visited websites. As the deployment of JavaScript only websites rise, it is necessary to fine-tune the API available to the websites to prevent unsolicited tracking and protect against data thefts.

JavaScript Restrictor currently supports modifying and restricting the following APIs (for more details visit levels of protection page):

- **Network boundary shield** (NBS) prevents web pages to use the browser as a proxy between local network and the public Internet. See the Force Point report for an example of the attack. The protection encapsulates the WebRequest API, so it captures all outgoing requests including all elements created by JavaScript.
- **window.Date object** and **window.performance functions** provide high-resolution timestamps that can be used to idenfity the user or can be used for microarchitectural attacks and timing attacks.
- **HTMLCanvasElement**: return white image data by modifiing canvas.toDataURL(), canvas.toBlob() and CanvasRenderingContext2D.getImageData() functions that can be used to fingerprint user’s device. Canvas element provides access to HW acceleration which may reveal the card and consequently be used as a fingerprinting source.
- **navigator.deviceMemory** or **navigator.hardwareConcurrency** can reveal hardware specification of the device.
- **XMLHttpRequest (XHR)** performs requests to the server after the page is displayed and gathered information available through other APIs. Such information might carry

identification data or results of other attacks.
- **ArrayBuffer** can be exploited for microarchitectural attacks.
    - Encapsulates window.DataView, window.Uint8Array, window.Int8Array, window.Uint8ClampedArray, window.Int16Array, window.Uint16Array, window.Int32Array, window.Uint32Array, window.Float32Array, window.Float64Array
- **SharedArrayBuffer (window.SharedArrayBuffer)** can be exploited for timing attacks.
- **WebWorker (window.Worker)** can be exploited for timing attacks.

JavaScript Restrictor provides four in-built levels of protection:

- 0 - the functionality of the extension is turned off. All web pages are displayed as intended without any interaction from JavaScript Restrictor.
- 1 - the minimal level of protection. Only changes that should not break most pages are enabled. Note that timestamps are rounded so pages relying on precise time may be broken.
- 2 - intended to be used as a default level of protection, this level should not break any site while maintaining strong protection.
- 3 - maximal level of protection: enable all functionality.

The default level of protection can be set by a popup (clicking on JSR icon) or through options of the extension. Specific level of protection for specific domains can be set in options by adding them to the list of websites with specific level of protection. This can be done also by a popup during a visit of the website.

Permissions

Javascript Restrictor requires these permissions:

- **storage –** *used for storing extension configuration and user options*
- **tabs –** *used for updating icon badge of the extension on tab change*
- **webRequest, webRequestBlocking, and all_urls –** *needed for modyfing JavaScript objects and APIs on all pages and also used for capturing and blocking malicious HTTP requests*
- **dns –** *used for resoluting DNS queries in Firefox version of HTTP request shield*
- **notifications–** *used for notifying user on blocked HTTP requests/hosts*

JavaScript Resctictor does **not** collect any data about users.

Network Boundary Shield (NBS)

NBS is active independently on the levels defined below. If necessary, you can whitelist websites for which the NBS should be turned off. Generally, you want NBS to be active, however, some pages can be broken, because they require interaction between public Internet and local network, for example, some Intranet information systems might be broken by the NBS.

Levels controlling JS-object wrapping

Level 0

- *All functionality is disabled*

Level 1

- **Manipulate the time precision provided by Date and performance: –** *ON*
  - Round time to: *hundredths of a second (1.230 – Date, 1230 – performance)*
- **Protect against canvas fingerprinting –** *OFF*
- **Spoof hardware information to the most popular HW –** *ON*
  - JS navigator.deviceMemory: *4* (not applied if the browser does not support the property, e.g. Firefox)
  - JS navigator.hardwareConcurrency: *2*
- **Filter XMLHttpRequest requests –** *OFF*
- **Protect against ArrayBuffer exploitation –** *OFF*
- **Protect against SharedArrayBuffer exploitation –** *OFF*
- **Protect against WebWorker exploitation –** *OFF*
- **Disable Battery status API –** *ON*

Level 2

- **Manipulate the time precision provided by Date and performance –** *ON*
  - Round time to: *tenths of a second (1.200 – Date, 1200 – performance)*
- **Protect against canvas fingerprinting: –** *ON*
  - Reading from canvas returns white image.
- **Spoof hardware information to the most popular HW –** *ON*
  - JS navigator.deviceMemory: *4* (not applied if the browser does not support the property)
  - JS navigator.hardwareConcurrency: *2*
- **Filter XMLHttpRequest requests –** *OFF*
- **Protect against ArrayBuffer exploitation –** *OFF*
- **Protect against SharedArrayBuffer exploitation –** *OFF*
- **Protect against WebWorker exploitation –** *OFF*
- **Disable Battery status API –** *ON*

Level 3

- **Manipulate the time precision provided by Date and performance –** *ON*
  - Round time to: *full seconds (1.000 – Date, 1000 – performance)*
    - *Randomize time*
- **Protect against canvas fingerprinting: –** *ON*
  - Reading from canvas returns white image.
- **Spoof hardware information to the most popular HW –** *ON*
  - JS navigator.deviceMemory: *4* (not applied if the browser does not support the property)
  - JS navigator.hardwareConcurrency: *2*
- **Filter XMLHttpRequest requests: –** *confirm requests but do not block*
- **Protect against ArrayBuffer exploitation –** *ON*
  - *Use random mapping of array indexing to memory*
- **Protect against SharedArrayBuffer exploitation –** *ON*
  - *Block SharedArrayBuffer* – SharedArrayBuffer provided by the browser is not available to page scripts at all.
- **Protect against WebWorker exploitation –** *ON*
  - *Remove real parallelism* – Use Worker polyfill instead of the native Worker.

- **Disable Battery status API –** *ON*