Detecting and Understanding JavaScript Global Identifier Conflicts on the Web

Paper review

Max Trunnikov 2023-08-22

Meta Data

- **Authors:** Mingxue Zhang, Wei Meng
- **Conference:** ESEC/FSE
- **Year:** 2020
- **Citations:** 4
- **References:** 36
- **Downloads:** 264
- **Pages:** 11

What is this study about?

1. Many web-applications include JavaScript code from different hosts
2. All scripts are loaded in the same single global namespace
3. Result: scripts may override global variables and functions from other ones
4. All this leads to unexpected behaviour and runtime errors

Table of Contents

Problem statement 👍

- **Three types of conflicts:**
  - Value conflicts
  - Function definition conflicts
  - Type conflicts
- **Scope of research:** problems caused by global identifier conflicts in scripts from different organizations:
  - third-party overwrites first-party
  - first-party overwrites third-party
  - third-party overwrites other third-party
- **Research challenges:**
  - Type inference
  - Object support
  - Alias analysis

Innovation 👈

- The article is not about innovations
- The article is about research and analysis

Contribution 👍

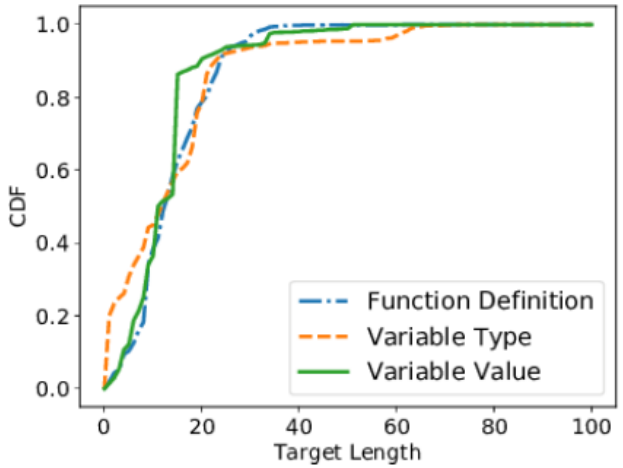
- **JSObserver** - browser based dynamic analysis framework for detecting JavaScript global identifier conflict at run time (https://github.com/cuhk-seclab/JSObserver)
- Analysed data from the main pages of **Alexa top 100K websites** in 2019 (https://zenodo.org/record/3874944)
- **Result:** 145K+ conflicts detected on 31K+ popular websites
- **Conclusion:** JavaScript global identifier conflict is an emerging threat to both the web users and the integrity of web applications. Need to isolate JavaScript code from different organizations.

Logical correctness 👍
Proof of statements 👍

- ✅ Design and methodology
- ✅ Implementation
- ✅ Evaluation
- ✅ Case studies
- ✅ Performance of JSObserver
- ✅ Related work
- ✅ Discussion

Readability 👍

- Very nice research article
- Clear and logical narration
- Visual content:
  - Graphics
  - Code
  - Statistics tables




```
1  function addHTML() {
2    var html = "...<img ...src=\"https://dre.pt/.../logo-portal.
         png\" ...> ";
3    if(isMobile.Android()){
4      html += "<div ...><a href=\"...\" ...></a>Aceder</div>";
5    }
6    else if(isMobile.iOS()) {
7      html += "<div ...><a href=\"...\" ...></a>Aceder</div>";
8    }
9    ... ...
10   document.body.innerHTML += html;
11 }
```

Listing 3: First-party definition of addHTML() on https://dre.pt/.

```
1  function addHTML() {
2    var html = "...<img ...src=\"https://dkq729jo4daj5.cloudfront
         .net/.../logo-portal.png\" ...> ";
3    html += "<div ...> <a href=\"\" ...></a> Aceder </div>";
4    ... ...
5    document.body.innerHTML += html;
6  }
```

Listing 4: Third-party redefinition of addHTML().

Table 2: Categorization of global identifier conflicts.

Category	#Websites	#Conflicts	%Conflicts
Function Definition	9,566	36,813	25.23
third -> first	715	1,510	1.03
third -> diff_third	311	543	0.37
first -> third	349	704	0.48
third -> same_third	1,283	7,086	4.86
first -> first	6,829	25,580	17.53
unknown	891	1,390	0.95
Variable Type	3,501	27,893	19.12
third -> first	338	556	0.38
third -> diff_third	156	206	0.14
first -> third	288	434	0.30
third -> same_third	434	820	0.56
first -> first	1,881	22,882	15.68
unknown	643	2,995	2.05
Variable Value	27,199	81,212	55.66
third -> first	7,128	8,582	5.88
third -> diff_third	5,302	7,476	5.12
first -> third	2,021	2,493	1.71
third -> same_third	4,270	9,302	6.37
first -> first	11,986	40,248	27.58
unknown	7,980	13,111	8.99

Design and Narrative Structure 👍

References and figures order ☹️

all the possible paths and revealed malicious behaviors. Other analysis includes data race detection [14, 22, 29], determinacy analysis [32], JavaScript performance profiling [9], concurrency error detection [13] and crash path computation [18]. These techniques are orthogonal to JSObserver, which focuses on JavaScript global identifier conflicts.

Table 9: Slowdown on Page Loading Time.

Round	Average (X)	Max (X)	#Incomplete Loading
1	10.84	192.48	2
2	11.58	194.78	6
3	10.45	213.62	4

...

4.6 Performance of JSObserver

We measure the slowdown on page loading time to evaluate the performance overhead incurred by JSObserver. Specifically, we used a Vanilla Chromium browser and the prototype of JSObserver to visit the Alexa top 100 websites separately, waited for at most 5 minutes before closing the browser, and calculated the average page loading time and the average slowdown in three rounds. The experiment results are shown in Table 9. As shown, JSObserver incurs an av-

Summary and links to the subsections 😊

2 PROBLEM STATEMENT

In this section, we first formally define the three types of conflicts that we study, then demonstrate the scope of our research, and finally discuss our research challenges.

3 DESIGN AND METHODOLOGY

In this section, we present JSObserver, a browser-based dynamic analysis framework for detecting JavaScript global identifier conflicts at run time. We record each function definition in the V8 parser to detect function definition conflicts (§3.1). We perform just-in-time instrumentation of all JavaScript code that is executed to cover all writes to a memory location (§3.2). The records allow us to detect conflicting writes by different scripts to the same global memory locations (§3.3).

An exemplary article