CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

TESTABLE

# It's (DOM) Clobbering Time:

## Attack Techniques, Prevalence, and Defenses

**Soheil Khodayari and Giancarlo Pellegrino**

CISPA Helmholtz Center for Information Security

44th IEEE Symposium on Security and Privacy
May 22-25, 2023

SCAN ME

soheil.khodayari@cispa.de

# DOM Clobbering

Code-less markup injection attack

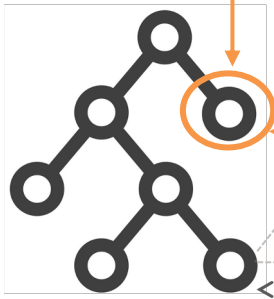
Markup id/name collides with sensitive variables or APIs, and overwrites them

```
<a id="globalConfig" href="malicious.js">
```

1 Inject HTML markup

*https://example.com*

DOM Tree



<script>

Vuln. Script

2

```
var s = document.createElement('script');
let config = window.globalConfig || {href: 'script.js'};
s.src = config.href;
document.body.appendChild(s);
// [...]
```

→ **Arbitrary Code Execution**

# Problem Statement

- Limited knowledge about DOM Clobbering
  - First instance in 2010 affecting frame-busting code

**Application code**

```
top.location = self.location
```

**Attack markup (injection)**

```
<iframe name=self src="evil.com">
```

***Q:*** *What other attack markups will work?*

Many combinations of tags, attributes, code features, and browser behaviours unexplored

No automatic detection technique or tool, and prevalence is unknown

Recent DOM Clobbering vulnerabilities in popular sites[1] question the efficacy of defenses

**RQ1:** Clobbering Markups and Browser Behaviours

**RQ2:** Vulnerability Detection and Prevalence

**RQ3:** Defenses and their Effectiveness

[1]***Source:*** *https://research.securitum.com/xss-in-amp4email-dom-clobbering/*

# RQ1: Clobbering Markups

**Goal:** automatically generate DOM Clobbering markups

Clobbering target: variable ***x.y***

**Example:** Known HTMLCollection: any two tags with the same id

```
<a id=x><a id=x name=y>
```


**24M** test cases, **19** browsers (mobile and desktop), cover all tags, attributes, relations and targets

- Test clobbering of variable *X*, object property *X.Y*, and *built-in* APIs

**Results**

Uncovered 31,432 distinct clobbering markups across five different techniques

Only 481 previously known

**Example:** New HTMLCollection: object tags with the same name

```
<object name=x><object name=x id=y>
```

# RQ1: Clobbering Markups

**Goal:** automatically generate DOM Clobbering markups

Clobbering target: variable ***x.y***

**Example:** Known HTMLCollection: any two tags with the same id

```
<a id=x><a id=x name=y>
```

1 Review HTML Specifications → 2 Derive Markup Generation Rules → 3 Test Markups in Browsers

24M test cases, 19 browsers (mobile and desktop), cover all tags, attributes, relatio d targets

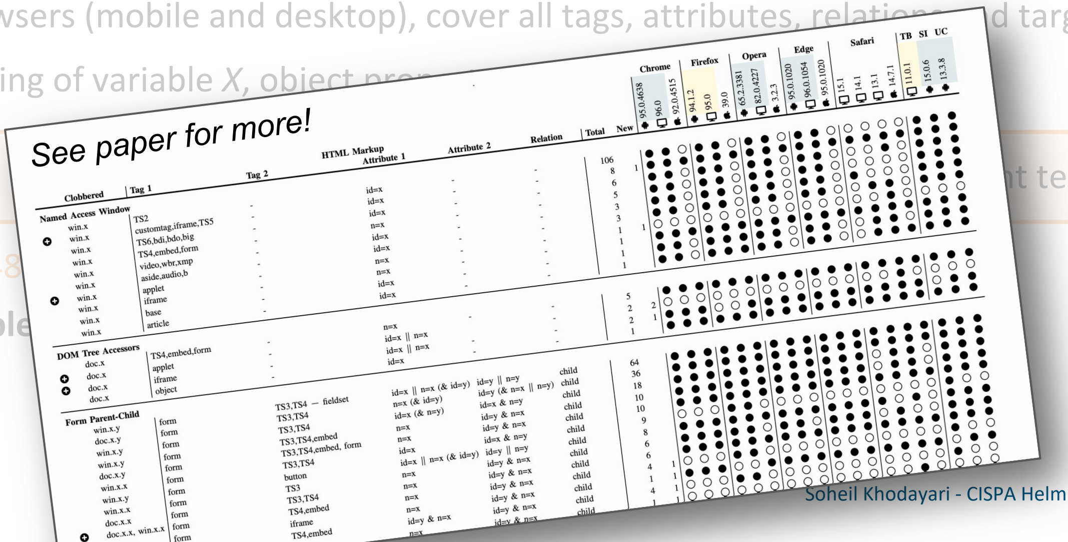
- Test clobbering of variable *X*, object prop

**Results**

nt techniques

Only 48

Example



See paper for more!

| | | | HTML Markup | | | | | Chrome | | | Firefox | | | Opera | | | Edge | | | Safari | | | | TB | SI | UC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clobbered | Tag 1 | Tag 2 | Attribute 1 | Attribute 2 | Relation | Total | New | 🤖 95.0.4638 | 🖥 96.0 | 🍎 92.0.4515 | 🤖 94.1.2 | 🖥 95.0 | 🍎 39.0 | 🤖 65.2.3381 | 🖥 82.0.4227 | 🍎 3.2.3 | 🤖 95.0.1020 | 🖥 96.0.1054 | 🍎 95.0.1020 | 🖥 15.1 | 🖥 14.1 | 🖥 13.1 | 🍎 14.7.1 | 🖥 11.0.1 | 🤖 15.0.6 | 🤖 13.3.8 |
| **Named Access Window** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| win.x | TS2 | - | id=x | - | - | 106 | | ● | ● | ○ | ● | ● | ○ | ● | ● | ○ | ● | ● | ○ | ○ | ○ | ○ | ○ | ● | ● | ● |
| ⊕ win.x | customtag,iframe,TS5 | - | id=x | - | - | 8 | 1 | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| win.x | TS6,bdi,bdo,big | - | id=x | - | - | 6 | | ● | ● | ○ | ● | ● | ○ | ● | ● | ○ | ● | ● | ○ | ○ | ○ | ● | ○ | ● | ● | ● |
| win.x | TS4,embed,form | - | n=x | - | - | 5 | | ● | ● | ○ | ● | ● | ○ | ● | ● | ○ | ● | ● | ○ | ○ | ○ | ○ | ○ | ● | ● | ● |
| win.x | video,wbr,xmp | - | id=x | - | - | 3 | | ● | ● | ○ | ● | ● | ○ | ● | ● | ○ | ● | ● | ○ | ● | ○ | ○ | ○ | ● | ● | ● |
| win.x | aside,audio,b | - | id=x | - | - | 3 | | ● | ● | ○ | ● | ● | ○ | ● | ● | ○ | ● | ● | ○ | ○ | ● | ● | ○ | ● | ● | ● |
| ⊕ win.x | applet | - | n=x | - | - | 1 | 1 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ○ | ○ | ○ | ○ |
| win.x | iframe | - | n=x | - | - | 1 | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| win.x | base | - | id=x | - | - | 1 | | ● | ● | ○ | ● | ● | ● | ● | ● | ○ | ● | ● | ○ | ○ | ● | ● | ○ | ● | ● | ● |
| win.x | article | - | id=x | - | - | 1 | | ● | ● | ○ | ● | ● | ● | ● | ● | ○ | ● | ● | ○ | ○ | ● | ● | ● | ● | ● | ● |
| **DOM Tree Accessors** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| doc.x | TS4,embed,form | - | n=x | - | - | 5 | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| ⊕ doc.x | applet | - | id=x ‖ n=x | - | - | 2 | 2 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ○ | ○ | ○ | ○ |
| ⊕ doc.x | iframe | - | id=x ‖ n=x | - | - | 2 | 1 | ● | ● | ● | ○ | ○ | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ○ | ● | ● |
| doc.x | object | - | id=x | - | - | 1 | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| **Form Parent-Child** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| win.x.y | form | TS3,TS4 — fieldset | id=x ‖ n=x (& id=y) | id=y ‖ n=y | child | 64 | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| doc.x.y | form | TS3,TS4 | n=x (& id=y) | id=y (& n=x ‖ n=y) | child | 36 | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| win.x.y | form | TS3,TS4 | id=x (& n=y) | id=x & n=y | child | 18 | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ○ | ● | ● | ○ | ● | ● | ● |
| win.x.y | form | TS3,TS4,embed | n=x | id=y & n=x | child | 10 | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ○ | ● | ● | ○ | ● | ● | ● |
| doc.x.y | form | TS3,TS4,embed, form | n=x | id=y & n=x | child | 10 | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| win.x.x | form | TS3,TS4 | id=x | id=x & n=y | child | 9 | | ○ | ○ | ○ | ● | ● | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ○ | ○ |
| win.x.y | form | button | id=x ‖ n=x (& id=y) | id=y ‖ n=y | child | 8 | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ○ | ● | ○ | ● | ● | ● |
| win.x.x | form | TS3 | n=x | id=y & n=x | child | 6 | | ● | ● | ● | ● | ● | ○ | ● | ● | ○ | ● | ● | ● | ○ | ● | ○ | ○ | ● | ● | ● |
| doc.x.x | form | TS3,TS4 | n=x | id=y & n=x | child | 6 | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| ⊕ doc.x.x, win.x.x | form | TS4,embed | n=x | id=y & n=x | child | 4 | 1 | ○ | ○ | ○ | ● | ● | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ○ | ○ |
| | form | iframe | n=x | id=y & n=x | child | 1 | 1 | ● | ● | ● | ○ | ○ | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ○ | ● | ● |
| | | TS4,embed | id=y & n=x | id=y & n=x | child | 4 | 1 | ○ | ○ | ○ | ● | ● | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ○ | ○ |
| | | | n=x | id=y & n=x | child | 1 | 1 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ○ | ○ | ○ | ○ |
| | | | | | child | | | | | | | | | | | | | | | | | | | | | |

# RQ1: Clobbering Markups – Online Demo

domclob.xyz/domc_payload_generator/

Home Wiki Markups Browser Testing Payload Generator Detection Download

## DOMC Payload Generator

Generates DOM Clobbering Attack Payload

**Clobbering Target**

window.globalConfig

Enter the target variable or expression you want to clobber here.

**Clobbering Value**

malicious.js

Enter the clobbered value for `href` or `src` of HTML markups.

Generate

**Attack Payload(s)**

```
<a id="globalConfig" href="malicious.js"></a>
```

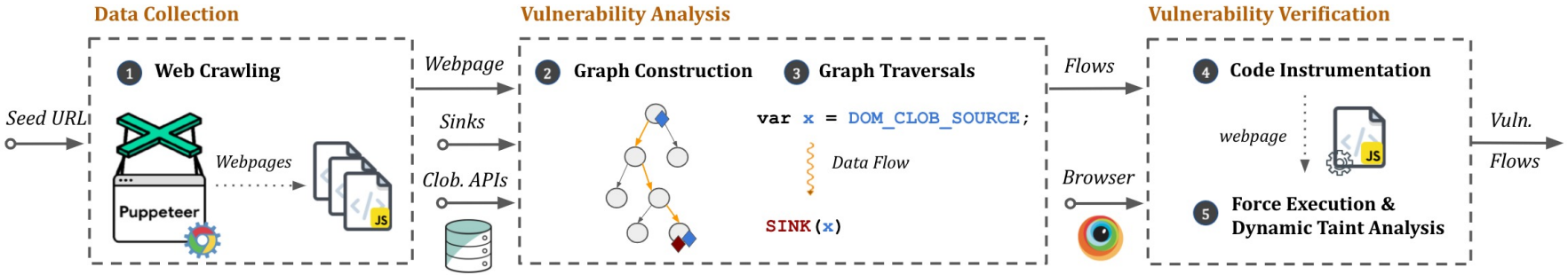
```
<customtag id="globalConfig"></customtag>
```

```
<article id="globalConfig"></article>
```

# RQ2: Vulnerability Detection – TheThing (JAW v2.x)

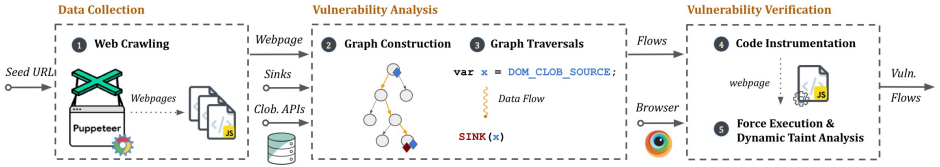
- Proposed an open source, static-dynamic tool for detecting DOM Clobbering at scale
- **Components**
  - Data Collection
  - Vulnerability Analysis
  - Vulnerability Verification

https://ja-w.me



# RQ2: Vulnerability Prevalence

- Empirical study to quantify the prevalence of DOM clobbering in the wild




## Testbed

Tranco top 5K websites, 205.6K webpages, 18.3M scripts, 24.6B LoC

## Results

- Detected 9,467 clobberable data flows across 491 affected sites
- Exploits for 44 websites (all confirmed and patched):
  - E.g., GitHub, Trello, Vimeo, Fandom, WikiBooks and VK
  - Client-side XSS, open redirections and request forgery attacks

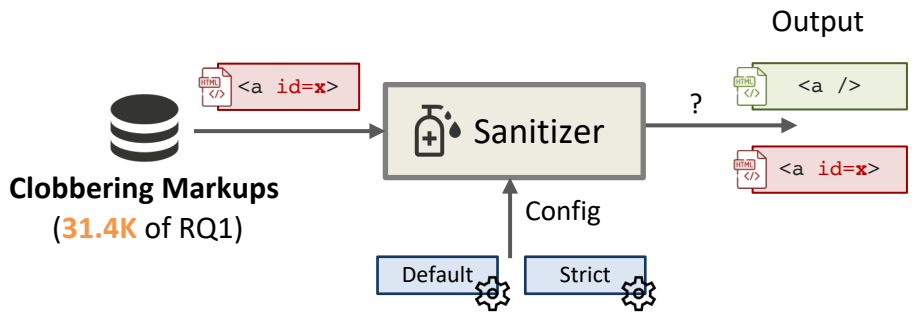
# RQ3: Defenses and their Effectiveness

## Mitigations

HTML Sanitization

Evaluated the robustness of 29 HTML sanitizers

- JS, Python, PHP, C#, and Java




## Results

In total, 16 sanitizers vulnerable to at least one clobbering markup by default

- Including popular ones like DOMPurify, Mozilla Bleach, and Google Caja
- 13 of them also vulnerable in most strict config

The other 13 sanitizers always remove named properties

- Including cases that do not lead to DOM Clobbering (e.g., `<a name=x>`)

# RQ3: Defenses and their Effectiveness

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

## Mitigations

HTML Sanitization

Namespace Isolation

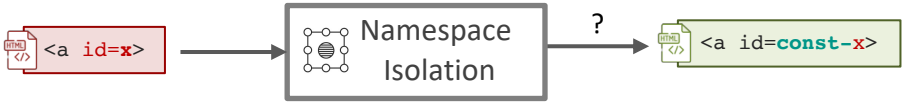
Content Security Policy

DOM Object Freezing

*See paper for more mitigations …*

**Alternative:** prefix/isolate named properties instead of removing them

- (+) mitigates almost all DOM Clobbering cases
- (-) may require some implementation changes by developers

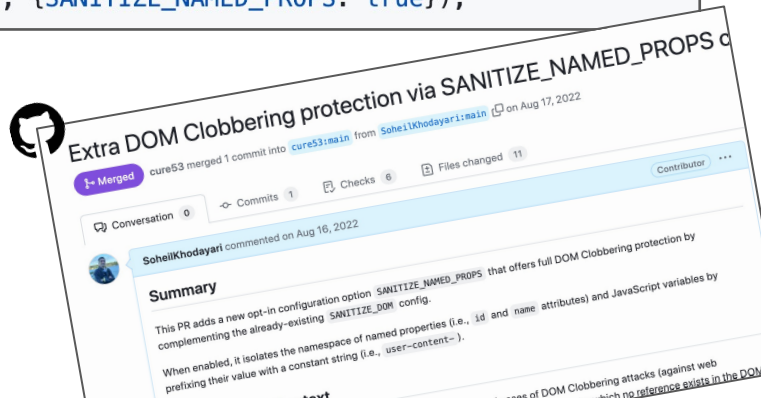



**Contribution:** implemented namespace isolation in DOMPurify

- Use the new SANITIZE_NAMED_PROPS config

```
var clean = DOMPurify.sanitize(dirty, {SANITIZE_NAMED_PROPS: true});
```

Learn more on GitHub...



# RQ3: Defenses and their Effectiveness

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

**Mitigations**

HTML Sanitization
Namespace Isolation
Content Security Policy
DOM Object Freezing

**Kill Switch**

Disabling DOM Clobbering

Infeasible

**Solution:** disable named properties at browser-level?[1]

- (+) fixes all DOM Clobbering cases
- (-) can cause breakage

**Measurement**

**Cost:** 13.3% of webpages use named properties and will break (~51% of sites)
**Benefit:** fixes the 491 vulnerable sites (i.e., 9.8% of top 5K sites)

⚠ breakage-benefit balance: ratio of ~5:1

**Proposal to W3C:** Opt-in CSP/feature policy flag to allow developers to disable name properties


[1]*Source: https://github.com/w3c/webappsec-permissions-policy/issues/349*

# RQ3: Vulnerable Patterns and Guidelines

- Identified eight common vulnerable code patterns in the wild

## Patterns

*1,214 webpages*

**A**

```
var VAR2 = window.VAR1 || CONST;
SINK(VAR2);
```

*832 webpages*

**B**

```
var VAR2 = [windoc.]API || CONST;
SINK(VAR2);
```

*655 webpages*

**C**

```
[document.VAR1 = CONST];
SINK(document.VAR1 || CONST);
```

## Guidelines

#1: Explicit Variable Declarations

```
var VAR1 = 'string';
```

#2: Strict Type Checking

```
If(!API instanceof HTMLElement)
```

#3: Do Not Use Document for Globals

```
const VAR1 = 'string';
```

# RQ3: Vulnerable Patterns and Guidelines

- Identified eight common vulnerable code patterns in the wild

**See paper for more!**

| # | Code Pattern |
|---|---|
| A | `var VAR2 = window.VAR1 \|\| CONST;`<br>`SINK(VAR2);` |
| B | `var VAR2 = [WinDoc.]BA \|\| CONST;`<br>`SINK(VAR2);` |
| C | `[document.VAR1 = CONST];`<br>`SINK(document.VAR1 \|\| CONST);` |
| D | `let VAR1 = VAR2 = CONST;`<br>`SINK(window.VAR1 \|\| CONST);` |
| E | `SINK(window.VAR1 \|\| CONST);`<br>`VAR1 = CONST;` |

**Incorporated to OWASP**

OWASP Cheat Sheet Series

Summary of Guidelines

For quick reference, below is the summary of guidelines discussed next.

| | Guidelines | Description |
|---|---|---|
| # 1 | Use HTML Sanitizers | link |
| # 2 | Use Content-Security Policy | link |
| # 3 | Freeze Sensitive DOM Objects | link |
| # 4 | Validate All Inputs to DOM Tree | link |
| # 5 | Use Explicit Variable Declarations | link |
| # 6 | Do Not Use Document and Window for Global Variables | link |
| # 7 | Do Not Trust Document Built-in APIs Before Validation | link |
| # 8 | Enforce Type Checking | link |
| # 9 | Use Strict Mode | link |

## Conclusion

# Thank You!

- Clobbering markups come in many forms (i.e., **31.4K** variants)
- DOM Clobbering is ubiquitous in the wild (i.e., **9.8%** of sites)
- Existing defenses helpful but may not completely cut it

@Soheil__K domclob.xyz github.com/SoheilKhodayari/TheThing