Defending against Prototype Pollution

Marcin Hoppe

@marcin_hoppe marcinhoppe.com

Overview

JavaScript prototypal inheritance

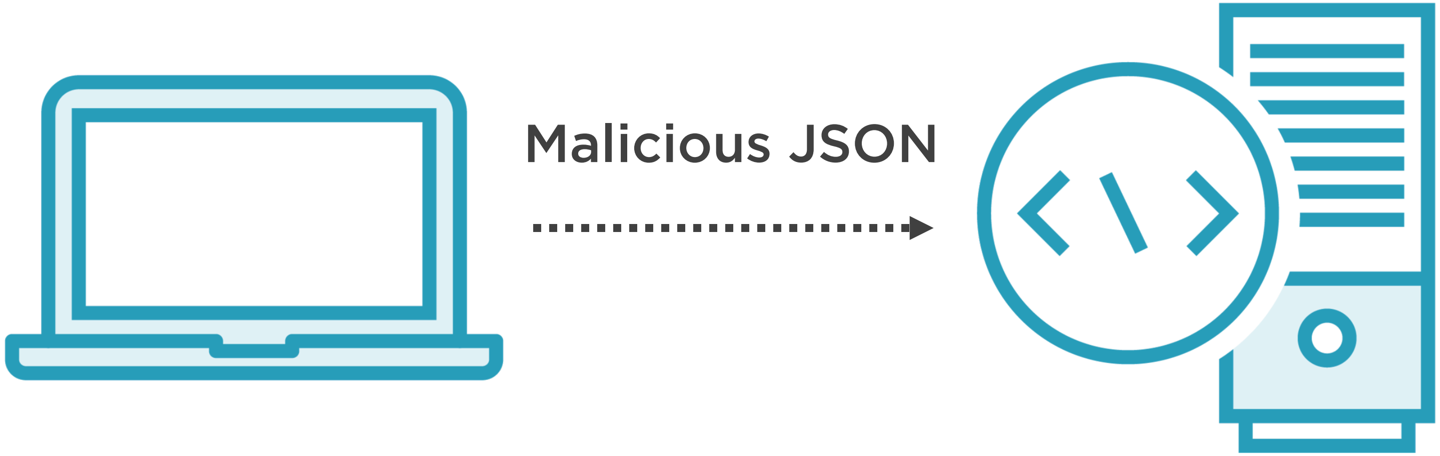
Modifying the prototype chain

- Parsing JSON data
- Dynamic property keys

Impact of prototype pollution

Hardening code against attacks

Prototype Pollution Attacks



Malicious JSON

Modify code behavior

Parse JSON

Merge parsed data with application objects

Inheritance Models

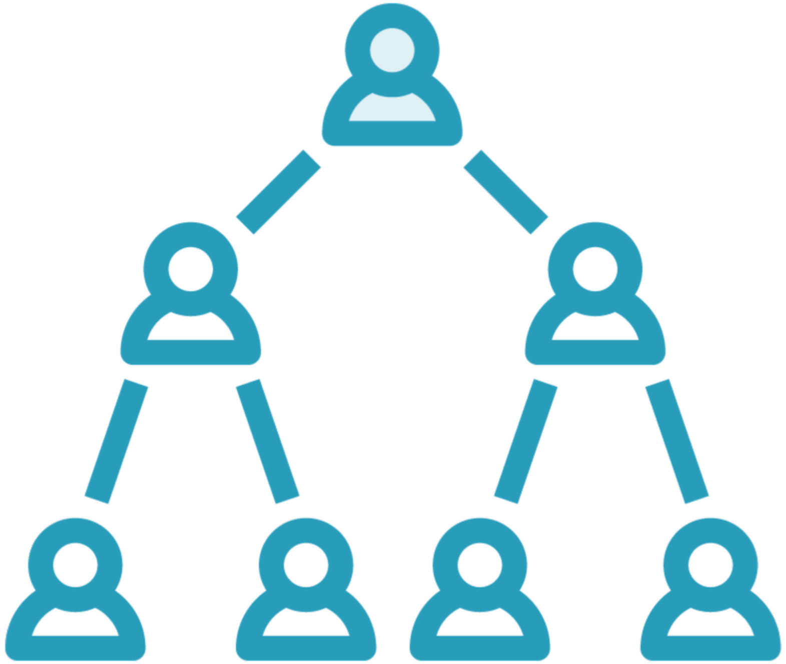
Classes

Static hierarchy of types

Prototypes

Dynamic chain of objects

Prototype Chain



Each object has a prototype

The chain ends will `null`

Inherited properties

Only own properties are mutated

The __proto__ Property

```
const parent = { a: 99 };

const child = Object.create(parent);

console.log(child.a);                        // 99

console.log(child.__proto__ === parent);     // true
```

JavaScript classes make it easier to set up prototype chains

Denial of service

`for-in` loop manipulation

Property injection

- Security check bypass
- SQL and NoSQL injections

Remote code execution

Prototype Pollution Example

```
const user = { name: ‘Full Name’ };    // Regular user

const malicious = { isAdmin: true };   // isAdmin is true for administrators only

user[‘__proto__’] = malicious;         // Pollution!

console.log(user.isAdmin);             // true. Escalation of privilege!
```

Prototype Pollution Code Smells

- Property mutation with untrusted key and value
- Recursive object merging
- Object cloning
- Property access by path

Demo

User profile management

Known attack sequence

- Hijack
- Inject
- Deliver

Denial of service

Session fixation

Validate JSON schema

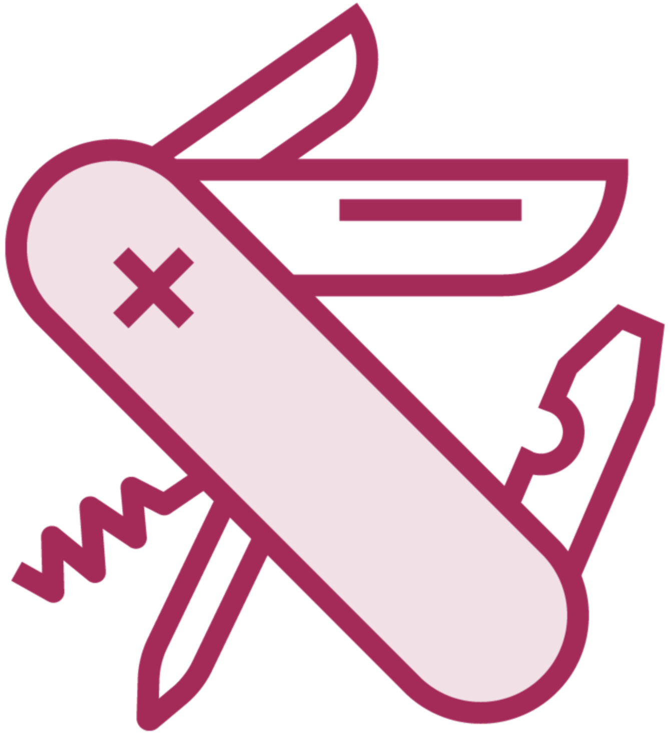
Freeze the prototype

- `Object.freeze`

Create objects without prototype

- `Object.create(null, ...)`

Use `Map` instead of `{}`



Utility libraries

Merging, cloning, extending

Examples

- jQuery
- Lodash
- Hapi

Summary

Prototype inheritance can be exploited

Property mutation with `__proto__` key

Mitigation techniques

- Input validation
- `Map` instead of `{}`
- Freezing or removing the prototype