Type-based Dependency Analysis for JavaScript

PLAS'13

University of Freiburg

Matthias Keil, Peter Thiemann
Institute for Computer Science
University of Freiburg
Freiburg, Germany

June 20, 2013, Seattle, WA, USA.

JavaScript

- JavaScript is the most important language for web sites
  - 92 % of all websites use JavaScript
- Web-developers rely on third-party libraries
  - e.g. for calendars, maps, social networks

```
<script type="text/javascript"
    src="http://example.org/api/js/?ARGS">
</script>
```

- Dynamic programming language
  - e.g. eval, mashups
- JavaScript has no security awareness
  - No namespace or encapsulation management
  - Global scope for variables/functions
  - All scripts have the same authority
- Security aspects of JavaScript received much attention
  - Static or dynamic analysis techniques
  - Guarantees by reducing the functionality

Security Scenarios

University of Freiburg

1. Libraries may get access to sensitive data
2. User code may be prone to injection attacks

- *Naive approach:* detect information flow
  - Pure information flow is too unflexible for investigating injection attacks
  - Ignores sanitized values
- *Our approach:* dependency analysis
  - Addresses both scenarios
  - Sanitized values are acceptable
  - Static Analysis
  - Implemented as extension of a type analyzer

Information flow

```
var t = Cookie.get('access_token');
// processing
// ...
Ajax.request('example.org', t);
```

Examples

University of Freiburg

Information flow

```
var t = Cookie.get('access_token');
// processing
// ...
Ajax.request('example.org', t);
```

Sanitization

```
var input = document.getElementById('id');
function sanitizer(value) {
    // clean up value
}
// processing
// ...
Ajax.request('example.org', sanitizer(input));
```

Design Principles

- Dependency Analysis
  - Information flow pioneered by Denning
  - Determines potential data flow between program points
  - Related to simple security types
- Flow-sensitive analysis
  - Abstracts data tainting
  - Stated as type-system

- Built-in **trace**$^{\ell}$(e[, id]) and **untrace**(e, id) function
  - ℓ – unique taint
  - id – tag name
  - Behaves like an identity function
- Implicit classes: UNSAFE, SAFE
- Policy-file
  - For predefined values (e.g. DOM, JavaScript API)

trace.policy

```
# Object Trace
trace: HTMLDocument.cookie; Ajax.request;
Array.prototype; Math.abs;
```

```
var userHandler = function(uid) {
    var name = '';
    var onSuccess = function(response) {name = response;};
    if (alreadyLoaded) {
        Cookie.request(uid, onSuccess);
    } else {
        Ajax.request('example.org', uid, onSuccess);
    }
    return name;
};
var name = userHandler(trace("uid"));
```

- Security properties on a fine level of granularity
    - Distinguish different sources

```javascript
var userHandler = function(uid) {
    var name = '';
    var onSuccess = function(response) {name = response;};
    if (alreadyLoaded) { // alreadyLoaded=true
        Cookie.request(uid, onSuccess);
    } else {
        Ajax.request('example.org', uid, onSuccess);
    }
    return name;
};
var name = userHandler(trace("uid"));
```

- Security properties on a fine level of granularity
    - Distinguish different sources

Application Scenario

Sensitive Data


```
var userHandler = function(uid) {
    var name = '';
    var onSuccess = function(response) {name = response;};
    if (alreadyLoaded) { // alreadyLoaded=(true|false)
        Cookie.request(uid, onSuccess);
    } else {
        Ajax.request('example.org', uid, onSuccess);
    }
    return name;
};
var name = userHandler(trace("uid"));
```

- Security properties on a fine level of granularity
    - Distinguish different sources

```
loadForeignCode = trace(function() {
    Array.prototype.foreach = function(callback) {
        // do something
    };
});
loadForeignCode();
// do something
array.foreach(function(k, v) {
    result = k + v;
});
```

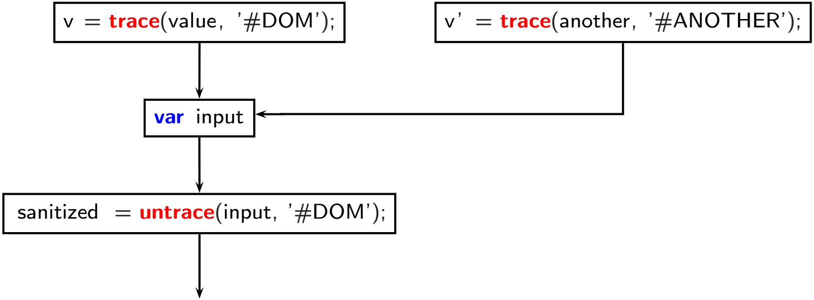
- Protect code from being compromised
    - Encapsulation of foreign code

```
$ = function(id) {
    return trace(document.getElementById(id).value, "#DOM");
}
function sanitizer(value) {
    // escape value
    return untrace(value, "#DOM");
}
// do something
var input = $("text");
var sanitizedInput = sanitizer(input);
consumer(sanitizedInput);
```

- Avoid injection attacks
  - e.g. only escaped values used
- Change taint classes

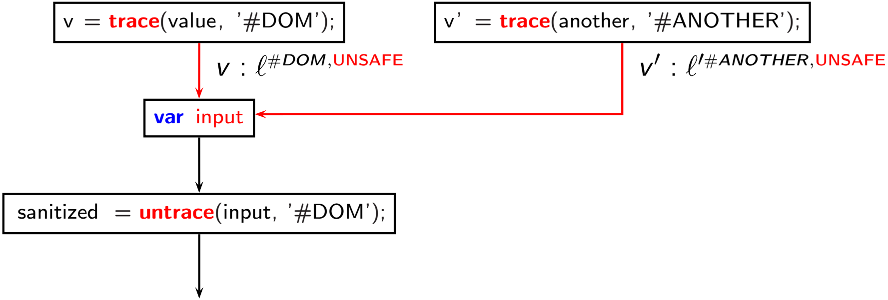
```
var sanitizedInput =
   i_know_what_i_do ? sanitizer(input) : input;
```

- Mixture of sanitized and unsanitized taints
- Flagged as an error



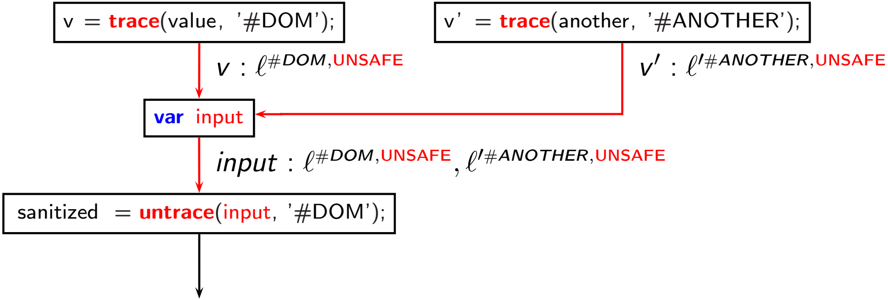
v = trace(value, '#DOM');
v' = trace(another, '#ANOTHER');
var input
sanitized = untrace(input, '#DOM');

Dependency Tracking Semantics

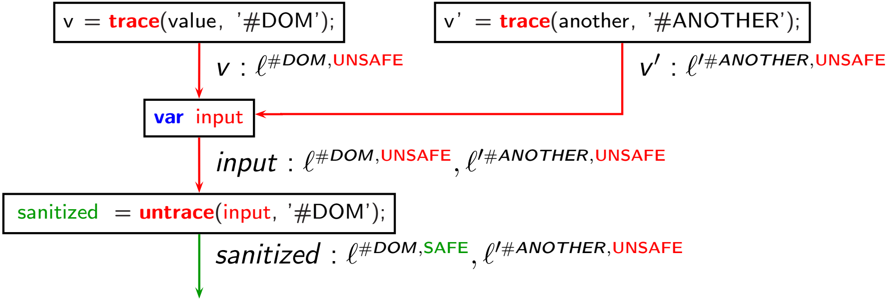


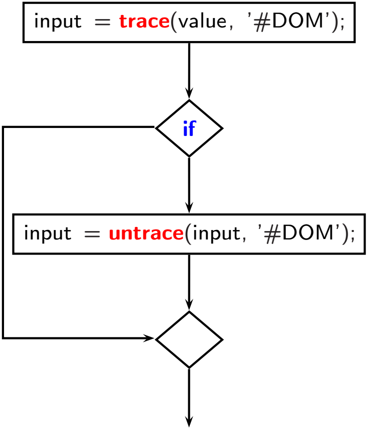
Dependency Tracking Semantics

University of Freiburg



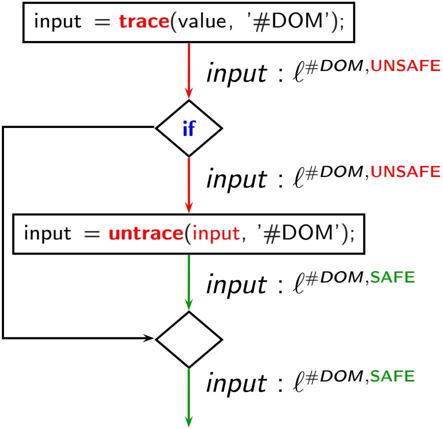
Dependency Tracking Semantics





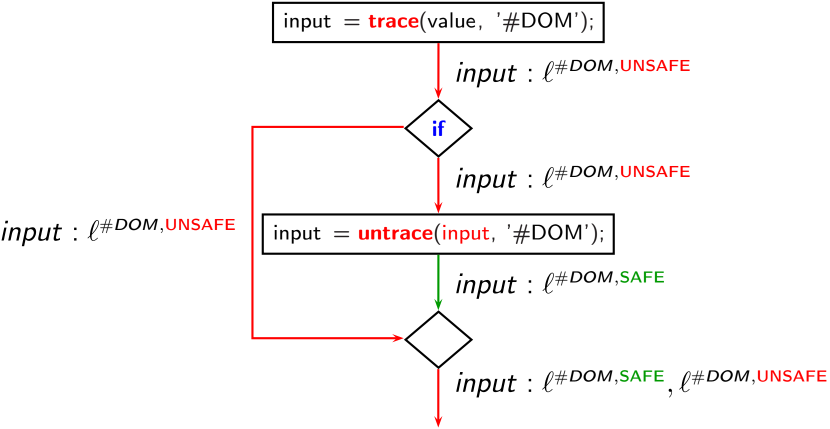
input = trace(value, '#DOM');
if
input = untrace(input, '#DOM');

Dependency Tracking Semantics



Dependency Tracking Semantics

University of Freiburg



Technical Contribution

- Formalization based on a typed JavaScript Core calculus
- Dependency Tracking Semantics
  - Marker propagation for upcoming values
  - Not meant to perform a dynamic analysis
- Static analysis based on the type system for dependency tracking
  - Termination-insensitive noninterference based on the types
  - Correct abstraction of the tracking semantics
  - Termination of the abstract analysis

Implementation

- Implementation extends TAJS, a Type Analyzer for JavaScript developed by Anders Møller and others
- Abstract values and states are extended with abstract taints
- The control flow graph is extended by special nodes for implicit flows
- **trace**$^{\ell}$ and **untrace** implemented as built-in functions
- Policy file for pre-labeling of built-in objects

- Designed and implemented a type-based dependency analysis for JavaScript
  - Analysis of information flow
  - Encapsulation of foreign code
  - Declassification of values (by changing taint-classes)
- Dependency analysis is not a security analysis
  - Investigate noninterference
  - Ensure confidentiality
  - Verify correct sanitizer placement

Questions?

Thank you for your attention.