

Language Based Isolation of Untrusted JavaScript

Ankur Taly

Dept. of Computer Science, Stanford University

Joint work with Sergio Maffei (Imperial College London) and
John C. Mitchell (Stanford University)

Outline

- 1 Web 2.0 and the Isolation Problem
 - Web Mashups
 - Isolation Problem
- 2 Existing Sandboxing Approaches
 - FBJS
 - ADSafe
 - Attacks on FBJS and ADSafe
- 3 Previous Research
 - Formal Semantics of JavaScript
 - Sub-language J_B
- 4 Solving the Isolation Problem
 - Formal Definition
 - Achieving Host Isolation
 - Achieving Inter-Component Isolation
 - Authority-Safety property
- 5 Conclusions and Future Work

Web 2.0

*All about mixing and merging content (data and code) from multiple content providers in the users browser, to provide high-value applications known as **mashups***

- Terminology:
 - Individual contents being mixed - *Components*.
 - Content Providers - *Principals*.
 - Publisher of the mashup- *Host*.
- Execution environment- Web Browser.
- Web page (DOM) - Shared resource.
- Most common language for mashups- **JavaScript**.
- Examples:
 - Basic mashups: Any web page with advertisements, iGoogle.
 - More complex mashups: Yelp, Yahoo Newsglobe ...

Example: Basic Mashup (Advertisements)

The screenshot shows a Windows Internet Explorer browser window displaying the article "How Web Advertising Works" on the HowStuffWorks website. The browser's address bar shows the URL: `http://computer.howstuffworks.com/web-advertising1.htm`. The page features a navigation menu with "Computer", "Internet", and "Web Design & Development" tabs. A search bar is prominently displayed with the text "Search HowStuffWorks and the web:" and a "SEARCH" button. The article content includes a list of 12 items under the heading "Inside This Article", with the second item, "In the Beginning: Banner Ads", highlighted. To the right of the article, there is a "Featured Video" section for "Cool Dual Monitors" and a large advertisement for Dell and Sony promoting a contest to win a digital camera by submitting photos. The advertisement includes the text "WHIP YOUR PHOTOS INTO SHAPE." and "EXPLORE & LEARN WITH DELL AND GET FREE TIPS ON ORGANIZING, EDITING AND PRINTING YOUR DIGITAL PHOTOS." The browser's status bar at the bottom shows "Done" and "Internet" with a 100% zoom level.

Example: Complex Mashup (Yelp)

yelp
Real People. Real Reviews.™

Search for (e.g. bars, baker, movie) **Italian food** Near (Address, City, State or Zip) **Palo Alto** Search

Welcome About Me Write a Review Find Reviews Invite Friends Messaging Talk Events Member Search | Account | Log In

Italian food Palo Alto

1 to 10 of 230 - Results per page: 10

Hide Filters

Sort By	Cities	Distance	Features	Price	Category
<ul style="list-style-type: none"> Best Match Highest Rated Most Reviewed 	<ul style="list-style-type: none"> <input type="checkbox"/> Palo Alto <input type="checkbox"/> Mountain View <input type="checkbox"/> Redwood City <input type="checkbox"/> Menlo Park ... More Cities » 	<ul style="list-style-type: none"> Bird's-eye View Driving (5 mi.) Biking (2 mi.) Walking (1 mi.) Within 4 blocks ... More features » 	<ul style="list-style-type: none"> <input type="checkbox"/> Open Now (7:00pm) <input type="checkbox"/> Good for Groups <input type="checkbox"/> Take-Out <input type="checkbox"/> Takes Reservations ... More features » 	<ul style="list-style-type: none"> \$\$\$\$ \$\$\$ \$\$ \$ 	<ul style="list-style-type: none"> <input type="checkbox"/> Italian <input type="checkbox"/> Food <input type="checkbox"/> Pizza <input type="checkbox"/> Dessert ... More categories »

Bella Vista Restaurant
Categories: French, Italian
Sponsored Result
★★★★☆ 85 reviews
13451 Skyline Blvd
Woodside, CA 94062
(650) 851-1220
There are just some places in this world that allow me to live in the past and the present. The Bella Vista is one such place. I have been coming here since I was a baby. Family scores say that I...

1. Caffe del Doge
Categories: Coffee & Tea, Italian
★★★★☆ 157 reviews
419 University Ave
Palo Alto, CA, 94301
(650) 323-3600
and the food isn't that great, and the coffee woman's Italian accents do absolutely nothing to the flavor of the coffee... but I can't help myself! I am convinced that this is the best coffee in University

2. Cafe Pro Bono
Category: Italian
★★★★☆ 22 reviews
2437 Birch St
Palo Alto, CA, 94306
(650) 325-1757
Yummy, authentic Italian food, located just off California Street. Went with a group of 10 for a birthday. Some hard pasta, some fish and some meat, everyone was pleased with their order. Service

Map showing Palo Alto area with red pins indicating restaurant locations.

Security Issue: Attack Host

INDIANTAGS Home » Register Sort news by: Recently Popular | [Top Today](#) | [Yesterday](#) | [Week](#) | [Month](#) | [Year](#) |

[Published News](#) | [Upcoming News](#)

Register

Register


Username: [Verify](#)

Email: lowercase letters only [Verify](#)

Password: five character minimum

Verify password:

Please enter the number provided in the image below. If you can not read the number you may refresh your browser.



[Continue](#)

What is INDIANTAGS? [↓](#)

INDIANTAGS is social news submitting site.vote for b

[read more](#)

Rep

Fabulous Festive Offers

SHOPPERS STOP
START SOMETHING NEW

**APPAREL *
ACCESSORIES *
HOME DÉCOR *
FRAGRANCES *
TOYS**

SHOP NOW @
www.shoppersstop.com

Security Issue: Attack other components

The screenshot shows a browser window displaying the Google homepage. The address bar shows the URL `http://www.google.com/ig`. The page features the Google logo, a search bar, and navigation links. Below the search bar are several widgets:

- Evil Gadget:** A cartoon illustration of a red devil character with horns, wings, and a tail, holding a pitchfork.
- Radio Paradise:** A music player widget titled "Now Playing:" listing songs by Perry Farrell, Jethro Tull, Talvin Singh, and Beth Orton.
- My Google Groups:** A widget showing a group named "google-dnswall (1)".
- Bejeweled:** A widget for the game Bejeweled, showing a colorful gem board and a "WELCOME TO BEJEWEL" message.
- Search YouTube:** A search bar for YouTube.
- CustomRSS:** A widget displaying a list of RSS feeds, including "Iraq veterans say that war crimes are encouraged by command" and "16 year-old builds electric pickup truck".
- Advertisements:** A "JCPenney™ Bedding Sale" advertisement for 30-50% off cozy bedding, sheets, quilts, and blankets.

Security Issues

- Each principal owns part of the resources and has *integrity* and *confidentiality* constraints over them.
 - Yelp restrictions: Google map scripts should not tamper with search results.
 - Google Map restrictions: Yelp code should not re-define any functions defined by google maps.
- *Mashups should be designed such that the interests of all principals, including the host are protected.*
- **High risk associated:** Credit card fraud, identity theft, loss of sensitive information
- Cannot afford to miss a single edge case- **Need a definitive proof of correctness.**

Our Model: Basic Mashups

Basic **JavaScript** mashup with non-interacting components.



- Two trust levels: **trusted** and **untrusted**.
- Untrusted components are sequentially composed and placed in a trusted context.
- Pages with advertisements, iGoogle, Facebook Apps.
- We consider **JavaScript sandboxing** as opposed to Iframes.
 - Iframes are restrictive, less control over contents of the frame.
 - Expensive to expose a library to Iframed code.

Design



- Isolation enforced **statically** at the server.

Isolation Problem

Isolation Problem

Design isolation mechanisms for untrusted components, so that they cannot access security critical resources belonging to the **host** and also **other untrusted components**.

Split the Isolation Property.

① Host Isolation

- Example: Untrusted component should not read `document.cookie` or write to `window.location`.
- Some existing approaches: ADSafe, FBJS, Caja.

② Inter-Component Isolation

- One untrusted component should not write to the variables defined by another untrusted component.
- Isolation between ads or two untrusted FBJS applications.
- Tricky! - FBJS, ADsafe and our earlier attempts fail.

A bit about JavaScript

- History :
 - Developed by Brendan Eich at Netscape.
 - Standardized for Browser Compatability : [ECMAScript 262-edition 3](#)
- First class functions, Prototype based language, re-definable object properties.
- Scope Objects/Stack frames can be first class JavaScript objects: Variable names \Leftrightarrow Property names.
- Implicit type conversions which can trigger user code.

```
var y = "a"; var x = {valueOf: function(){ return y;}}
x = x + 10;
js> "a10"
```

Quick Case study: Facebook *FBJS*

- Basics:

- Facebook apps are either Iframed or [integrated](#). We are interested in integrated apps.
- Integrated FaceBook applications are written in [FBML/FBJS](#): Facebook subsets of HTML and JavaScript.
- FBJS is served from Facebook, after [filtering and rewriting](#).
- Facebook libraries mediate access to the DOM ([Wrapping](#)).

- Security goals:

- No direct access to the DOM.
- No tampering with the execution environment
- No tampering with Facebook libraries.

Isolation Approach

Filtering:

- **Blacklist** security-critical variable names and disallow them.
- No `eval`, `Function`,

Rewriting:

- `this` \longrightarrow `ref(this)`.
 - $ref(x) = x$ if $x \neq window$ else $ref(x) = null$.
- `e1[e2]` \longrightarrow `e1[idx(e2)]`.
 - $idx(e)$ returns error if e evaluates to a black-listed property name and behaves as identity otherwise.

Wrapping: Facebook provides various **wrapped** DOM functions to provide **controlled** access to the DOM.

Quick Casestudy: Yahoo! *ADsafe* (Douglas Crockford)

- Basics:

- A **safe subset** of JavaScript to be used by **untrusted ad code not placed in an Iframe**.
- Hosting page first places the ADSafe library (*adsafe.js*) on its page.
- Untrusted ad code must be written in an ADSafe compliant manner. Tool for checking compliance: **JSLint**.
- All interaction with the trusted code is mediated by the ADSafe library.

- Security Goals:

- No direct access to DOM.
- No tampering with the execution environment
- No tampering with *ADsafe* libraries.

Isolation Approach

Design

```
<script>
  "use_strict";
  ADSAFE.go("WIDGETNAME_", function (dom) { // Untrusted Code });
</script>
```

- Basic Restrictions
- No *this*, *with*, *e[e]*, global variables,
- Banned variables:
 - arguments*, *callee*, *caller*, *constructor*, *eval*, *prototype*....
- Some functionality restored via 'ADSAFE' object (provided by the library).
 - ADSAFE.get(o,p): Access property *p* of object *o*.
 - ADSAFE.create(o): Create object that inherits from *o*.
 - . . .

Recent FBJs attack

Attack- Get hold of window object !

```
var f = function(){};
f.bind.apply =
  (function(old){return function(x,y){
    var getWindow = y[1].setReplay;
    getWindow(0).alert("Hacked!");
    return old(x,y)
  })(f.bind.apply)}
```

- JavaScript offers two ways to call a function: $o.f(v)$ or $f.apply(o, v)$.
- While using $f.apply(o, v)$, we need to **make sure that the apply method is non-malicious !**
- Reported to Facebook.

Recent ADSafe attack

Attack - Run arbitrary script !

```
var o = {toString:function(){o.toString =  
                function(){return "script"};  
                return "div"}}};  
dom.append(dom.tag(o).append(dom.text("alert('Hacked!')")));
```

- `dom.tag` expects a tag-name string, and creates a node if the tag-name is allowed.
- Confuse `dom.tag` by passing it an object that returns "div" when converted to string first time and "script" the second time.
- Reported to Doug Crockford.

Conclusion

- All attacks found till date are edge cases which the sandboxing technique misses.
- Sandbox designer does not account for all possible future states !
- We need a **systematic design** followed by a **proof of correctness** to make sure that we have covered all cases.

Outline

- 1 Web 2.0 and the Isolation Problem
 - Web Mashups
 - Isolation Problem
- 2 Existing Sandboxing Approaches
 - FBJs
 - ADSafe
 - Attacks on FBJs and ADSafe
- 3 Previous Research
 - Formal Semantics of JavaScript
 - Sub-language J_B
- 4 Solving the Isolation Problem
 - Formal Definition
 - Achieving Host Isolation
 - Achieving Inter-Component Isolation
 - Authority-Safety property
- 5 Conclusions and Future Work

Our previous research: Provably correct sandboxing

Two main contributions:

- 1 Formal Semantics of JavaScript
- 2 Sub-language $J_{\mathcal{B}}$ and source-source rewriting $Enf_{\mathcal{B}}$, for enforcing a black-list \mathcal{B} .
 - **Property:** No rewritten program can access properties from the black-list \mathcal{B} or get hold of the global object.
 - Rigorous proof of correctness.
 - As expressive as *FBJS*.
 - Developed in a series of papers - CSF'09, W2SP'09, ESORICS'09.

Rest of this talk:

- Review 1 and 2
- Analyze isolation goals that **can** and **cannot** be achieved using the sandbox $J_{\mathcal{B}}, Enf_{\mathcal{B}}$.

Our previous research: Provably correct sandboxing

Two main contributions:

- ① Formal Semantics of JavaScript
- ② Sub-language $J_{\mathcal{B}}$ and source-source rewriting $Enf_{\mathcal{B}}$, for enforcing a black-list \mathcal{B} .
 - **Property:** No rewritten program can access properties from the black-list \mathcal{B} or get hold of the global object.
 - Rigorous proof of correctness.
 - As expressive as *FBJS*.
 - Developed in a series of papers - CSF'09, W2SP'09, ESORICS'09.

Rest of this talk:

- Review 1 and 2
- Analyze isolation goals that **can** and **cannot** be achieved using the sandbox $J_{\mathcal{B}}$, $Enf_{\mathcal{B}}$.

Formal Semantics of JavaScript

Formalized all of [ECMA-262-3rd](#) edition ($JS_{ecma262}$).

- Small step style operational semantics.
 - Meaning of a program \Leftrightarrow sequence of actions that are taken during its execution.
 - Specify sequence of actions as transitions of an [Abstract machine](#)
- Developed formal semantics as basis for proofs (APLAS'08)
 - Very long (70 pages of ascii).
 - DOM is just treated as a library object.
 - We experimented with available browsers and shells
 - Defining an operational semantics for a real programming language is hard: sheer size and JavaScript peculiarities.

We are in the process of migrating to ES5 but current semantics is adequate for analyzing *ADsafe* and *FBJS*.

A glimpse of the rules

State

Program state is represented as a triple $\langle H, l, t \rangle$.

- H : Denotes the Heap, mapping from the set of locations (\mathbb{L}) to objects. H_0 is used to denote the initial heap.
 - Objects are maps from property names (\mathbb{P}) to values (v).
- l : Location of the current scope object (or current activation record).
- t : Term being evaluated.

- General form of a rule $\frac{\langle \text{premise} \rangle}{H_1, l_1, t_1 \rightarrow H_2, l_2, t_2}$.
- We use H_0 to denote the initial JavaScript heap and l_G to denote the global object.

Language $J_{\mathcal{B}}$, Rewriting $Enf_{\mathcal{B}}$

Goal: Prevent access to property names from blacklist \mathcal{B} and global object.

JavaScript Facts:

- Two kinds of Property Access:
 - Explicit: x , $e1.p$, $e1[e2]$
 - Implicit: `toString`, `valueOf` ...
We found the complete list \mathcal{P}_{nat} .
- Ways to access global object:
 - `this`
 - Calling native methods of the form `function()(... return this)`.
- Dynamic Code Generation: `eval`, `Function`, `constructor`.

Design

Controlling $e.x$ and x .

Filter 1

Filter all terms containing an identifier or property name from $\mathcal{B} \cup \{eval, Function, constructor\}$ and also any $\$$ -prefixed property name.

Controlling $e1[e2]$.

- Approach: Rewrite $e1[e2]$ to $e1[IDX(e2)]$
- Need to avoid “confused IDX” attacks.

Design

Our IDX function.

Init 1

```

var $String = String;
var $BL = {p1:true,...,pn:true, eval:true,...,$:true,...}

```

Rewrite 1

Rewrite every occurrence of $e1[e2]$ by $e1[IDX(e2)]$

```

IDX(e2) = ($=e2.toString:function()return ($=$String($),CHECK_$))
CHECK_$ = ($BL[$] ? "bad" :
    ($ == "constructor" ? "bad" : $ == "eval" ? "bad" :
    ($ == "Function" ? "bad" : ($[0] == "$" ? "bad" : $))))

```

Preventing access to global object

Taking care of `this`: Rewrite `this` with suitable check.

Rewrite 1

Rewrite every occurrence of `this` to `NOGLOBALTHIS`.

```
NOGLOBALTHIS = (this== $\$g$ ?null;this)
```

Save global object in `$\$g$` .

Init 1

```
var  $\$g$  = this;
```

Other ways of getting hold of global object:

- Method `valueOf` of `Object.prototype` and `sort`, `concat`, `reverse` of `Array.prototype` can potentially return pointer to global object.
- Define `wrappers` with `NOGLOBAL` check on return value.

Wrapping native methods

Init 2 (Wrapper)

```
$OPvalueOf = Object.prototype.valueOf;  
$OPvalueOf.call = Function.prototype.call;  
Object.prototype.valueOf =  
function(){var $= $OPvalueOf.call(this); return ($==$g?null:$)}
```

- Similarly *Init3*, 4, 5 for `sort`, `concat`, `reverse`.
- A copy of original `call` method is saved, motivated by another *FBJS* attack.
- Wrapping `eval` and `Function`: doable, but need to define a JavaScript expression that parses, filters and rewrites strings meant to represent JavaScript terms. (`constructor` will be the only thing left then!)

Result

- Define $J_{\mathcal{B}}$ as JavaScript with *Filter 1* applied.
- Define $Enf_{\mathcal{B}}$ as the composition of functions *Rewrite 1*, *Rewrite 2*.
- Define H_{wrap} as the heap obtained after executing *Init 1* and *Init 2* on the initial JavaScript heap H_0 .

Let l_G be the global object.

Theorem

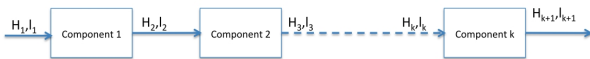
For all user terms $t \in J_{\mathcal{B}}$, the following holds for the reduction trace of $Enf_{\mathcal{B}}(t)$ starting from H_{wrap}, l_G

- 1 **Blacklist:** No property from the black-list \mathcal{B} is accessed (provided $\mathcal{B} \cap \mathcal{P}_{nat} = \emptyset$).
- 2 **No Global:** Final value returned is never the global object.

Outline

- 1 Web 2.0 and the Isolation Problem
 - Web Mashups
 - Isolation Problem
- 2 Existing Sandboxing Approaches
 - FBJS
 - ADSafe
 - Attacks on FBJS and ADSafe
- 3 Previous Research
 - Formal Semantics of JavaScript
 - Sub-language J_B
- 4 Solving the Isolation Problem
 - Formal Definition
 - Achieving Host Isolation
 - Achieving Inter-Component Isolation
 - Authority-Safety property
- 5 Conclusions and Future Work

Isolation problem



- Let blacklist \mathcal{B}_{host} denote critical elements of hosting page .
- Let t_1, \dots, t_n be programs running in components $1, \dots, n$.

Isolation Problem

Define an initial environment H_{mash}, l_{mash} and an enforcement technique for each component such that:

- 1 **Host Isolation:** For all i , reduction trace of component i starting from H_i, l_i does not access any property from \mathcal{B}_{host} .
- 2 **Inter-Component Isolation:** For all $i, j, i < j$, reduction of component i does not write to any heap location that component j reads from.

Isolation technique

We first evaluate the following isolation technique:

- Initial environment H_{wrap}, I_G .
- Enforcement technique Enf_i for component (t_i, id_i) :
 - ① Check containment in J_B
 - ② Rewrite program t_i to $Enf_B(t_i)$.
 - ③ Rewrite every variable x in $Enf_B(t_i)$ to $id_i x$.
- Intuitively this seems correct.
 - 1 and 2 should give host isolation.
 - 3 should give inter-component isolation.

Lets be systematic !

Host Isolation

- From the correctness theorem for sandbox $J_{\mathcal{B}}, \text{Enf}_{\mathcal{B}}$ we have:
 - Reduction trace of $\text{Enf}_i(t_i)$ starting from H_{wrap}, l_G will never access any property in $\mathcal{B}_{\text{host}}$.
 - But what about the trace starting from H_k, l_k ?
 - We do not know H_k, l_k in advance !
- Fortunately, we can formally show that the property holds for starting heap-scope H_j, l_j , provided all other components are also enforced.
- Therefore the isolation technique is sufficient for Host Isolation.

Host Isolation

- From the correctness theorem for sandbox $J_{\mathcal{B}}, \text{Enf}_{\mathcal{B}}$ we have:
 - Reduction trace of $\text{Enf}_i(t_i)$ starting from H_{wrap}, l_G will never access any property in $\mathcal{B}_{\text{host}}$.
 - But what about the trace starting from H_k, l_k ?
 - We do not know H_k, l_k in advance !
- Fortunately, we can formally show that the property holds for starting heap-scope H_i, l_i , provided all other components are also enforced.
- Therefore the isolation technique is sufficient for Host Isolation.

Inter-Component Isolation

Intuition:

- Global object is the common object shared between components.
- **No Global** property ensures that no component can get a handle to the global object.
 - Blocks access to global object via `e.p` and `e1[e2]`.
- Variable renaming separates namespace.
 - Isolates access to global object via `x`.

Can we conclude each component will access different portion of the global object ?

No, what if component j can reach a function defined by component i which has id_j prefixed variables !

Inter-Component Isolation

Intuition:

- Global object is the common object shared between components.
- **No Global** property ensures that no component can get a handle to the global object.
 - Blocks access to global object via `e.p` and `e1[e2]`.
- Variable renaming separates namespace.
 - Isolates access to global object via `x`.

Can we conclude each component will access different portion of the global object ?

No, what if component j can reach a function defined by component i which has id_j prefixed variables !

Communication via native objects

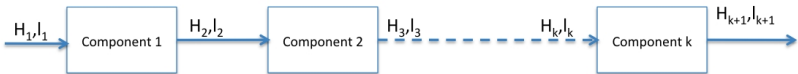
Attack

Component *i*: `f.toString.channel = function()(a = 1)`

Component *j*: `f.toString.channel()`

- `f.toString` and `f.toString` will point to the common location `Function.prototype.toString` even after namespace separation.
- Components *i* and *j* can use this location as a **communication channel**.
- We found a **real FBJs attack** where one app can (maliciously) change the meaning of another app.
- There are other communication channels as well:
`Array.prototype.push` and `Array.prototype.pop`.

What has gone wrong ?



- Our sandboxing technique restricts execution starting from H_{wrap}, I_G but does not provide any guarantees for H_k, I_k
- Execution of one component can transform the heap such that another component can break out of the sandbox !
- We are caught in the problem of **not being able to account for all future states !**

Concept: Authority

Authority ($Auth(H, l, t)$)

Authority of a term t for a given heap-scope H, l is **some over-approximation** of the set of all possible heap actions that can be performed during the reduction of the term.

- Inter-component isolation will hold if for all $i, j, i < j$, we can ensure that
 $Auth(H_i, l_i, t_i)$ does not overlap $Auth(H_j, l_j, t_j)$
- But this check is not useful as we don't know H_i, l_i !
- We can at most know authority of each component for the initial heap-scope.

Authority Safe language

Authority Safety

A language is said to be authority safe if there exists an authority map $Auth$ such that

- ① **Only Connectivity begets Connectivity** The execution of a term t_i starting from H, l can only affect the authority of a term t_j if $Auth(H, l, t_i)$ overlaps with $Auth(H, l, t_j)$
- ② **No Authority Amplification** The execution of a term t_i starting from H, l can at most increase the authority of another term t_j by $Auth(H, l, t_i)$.

Thus non-overlapping authorities ensure no communication is possible.

Result

Authority Isolation

Given an authority safe language, authority isolation holds for terms t_1, \dots, t_n for heap scope H, l if for all $i, j, i \neq j$
 $Auth(H, l, t_i)$ does not overlap with $Auth(H, l, t_j)$

Theorem

Authority Isolation \implies Inter-Component Isolation.

- Authority safety saves us from worrying about the intermediate heap-scopes.
- Reduces the problem to defining an appropriate source-to-source rewriting so that **initial** authorities are **non-overlapping**.
- Justifies one time source-to-source rewriting approach.

Solving the mashup isolation problem

- We restrict the language J_B and derive an authority-safe subset J_{safe} .
 - Make native object properties **read-only**.
 - Wrap native functions which can act as implicit communication channels.
 -
- We define an initial heap-scope H, I and enforcement functions Enf_1, \dots, Enf_n such that for all $i, j, i \neq j$, **$Auth(H, I, t_i)$ does not overlap with $Auth(H, I, t_j)$**

Details and rigorous proof of correctness is provided in the paper.

Insight

- Sandbox designers have a mental model of the what code placed in the sandbox can and cannot do - [Anticipated Authority](#)
- Sandboxes are calibrated so that anticipated authorities are isolated.
- Reason things break:
[Anticipated Authority < True Authority](#)
- How do we ensure that the mental model captures true authority ?

Prove Authority Safety !

Insight

- Sandbox designers have a mental model of the what code placed in the sandbox can and cannot do - [Anticipated Authority](#)
- Sandboxes are calibrated so that anticipated authorities are isolated.
- Reason things break:
[Anticipated Authority < True Authority](#)
- How do we ensure that the mental model captures true authority ?

[Prove Authority Safety !](#)

Achieving authority-isolation in general

Object Capabilities

- Capabilities can be viewed as [small bags of authority](#).
 - A pointer can be a capability with set of all reachable locations being its authority.
- The authority of a term is the union of authority arising all capabilities it possesses.
- Authority isolation can be achieved by appropriately distributing capabilities to the various components such that no two components have overlapping authority.
- Approach used by Google Caja.
- This is explained very formally in our Oakland 2010 paper.
[Object Capabilities and Isolation of Untrusted Web Applications](#)

Conclusions and Future Work

- Conclusions:
 - Building correct JavaScript sandboxing mechanism for host isolation is tricky !
 - Sandboxes can protect the host page but may not work for inter-component isolation.
 - Object Capabilities seem like a promising approach for inter-component isolation.
- Ongoing work:
 - Formalized the notion of Object-capability-safety and Authority-safety.
 - First cut at a proof of concept for Google Caja.
- Future work:
 - We plan to write the JavaScript semantics in machine readable format so that the proofs can be automated.
 - Formalize the concept of *Defensive consistency* and its connection with Object-capability-safety .

Conclusions and Future Work

- Conclusions:
 - Building correct JavaScript sandboxing mechanism for host isolation is tricky !
 - Sandboxes can protect the host page but may not work for inter-component isolation.
 - Object Capabilities seem like a promising approach for inter-component isolation.
- Ongoing work:
 - Formalized the notion of Object-capability-safety and Authority-safety.
 - First cut at a proof of concept for Google Caja.
- Future work:
 - We plan to write the JavaScript semantics in machine readable format so that the proofs can be automated.
 - Formalize the concept of *Defensive consistency* and its connection with Object-capability-safety .

Thank You !