



Using Data-Centric Policy Enforcement

Secure Data from Unknown Third Party Gadgets

Tianhao Tong

Web Mashups – The Challenge

Web [Images](#) [Videos](#) [Maps](#) [News](#) [Shopping](#) [Gmail](#) [more](#) ▼

tongtianhao@gmail.com | [Classic Home](#) | [My Account](#) | [Sign out](#)

iGoogle

Advanced Search
Search Preferences
Language Tools

Google Search I'm Feeling Lucky

New! Add social gadgets to post updates and play games with friends. [Change theme from Blue Plasma](#) | [Add stuff](#) »

Main Page

- Weather
- Ursa's photos
- World Time Server ...
- Google Calendar

World

- Top Stories
- Yahoo! News: Top S...
- Yahoo! News: Most ...
- Hack a Day
- Currency Converter
- World Sunlight Map
- LabPixies Clock
- Digital Blasphemy 3...

Updates

Friends

Chat

Search, add, or invite

- Tianhao Tong
imAvailable

Chat with friends in iGoogle!

Rather stay offline?
[Sign out of chat.](#)

- Ling Zhang
- Antonio Valdez A
- Yan Huang
<http://docs.google.com>

Ursa's photos



World Time Server Clocks



VA Shanghai

Weather

Charlottesville, VA

16°C Current: Sunny
Wind: W at 27 km/h
Humidity: 43%

Thu	Fri	Sat	Sun
16° 12°	11° 12°	3° -4°	6° -4°

Shanghai

4°C Current: Clear
Wind: N at 0 km/h
Humidity: 81%

Fri	Sat	Sun	Mon
11° 12°	7° 2°	12° 5°	16° 6°

Hong Kong

Information is temporarily unavailable.

Google Calendar



The page isn't redirecting properly

Firefox has detected that the

New Challenges

- Mashups
 - The most interesting innovation in software development in 20 years.
 - Mashups are insecure.
- Applications in SNS
 - Submitted by 3rd part

If there is script from two or more sources, the application is not secure.

Period.

Security issues in JavaScript

- Server Side Problem
 - Code injection
- Client Side Problem
 - Browser Implementation
- Language Problem
 - a Powerful yet insecure language

Server Side Problem

- A simple example

```
http://yourdomain.com/
```

```
<script>alert("XSS");</script>
```

```
<html><body>
```

```
<p>File not found:
```

```
<script>alert("XSS");</script>
```

```
</p></body></html>
```

- The script runs with the authority of your site.

Server Side Problem

- Other examples include all fields on a web site that let you input and display the input back to the client
 - (search box, user name, etc.)
- A complete list available:
<http://ha.ckers.org/xss.html>

ha.ckers

XSS (Cross Site Scripting) Cheat Sheet Esp: for filter evasion

XSS (Cross Site Scripting):

XSS locator. Inject this string, and in most cases where a script is vulnerable with no special XSS vector requirements the word "XSS" will pop up. Use the [URL encoding calculator](#) below to encode the entire string. Tip: if you're in a rush and need to quickly check a page, often times injecting the deprecated "<PLAINTEXT>" tag will be enough to check to see if something is vulnerable to XSS by messing up the output appreciably:

```
' ;alert (String.fromCharCode (88,83,83)) //  
\';alert (String.fromCharCode  
(88,83,83)) //";alert (String.fromCharCode
```

Browser support: [IE7.0|IE6.0|NS8.1-IE] [NS8.1-G|FF2.0] [09.02]

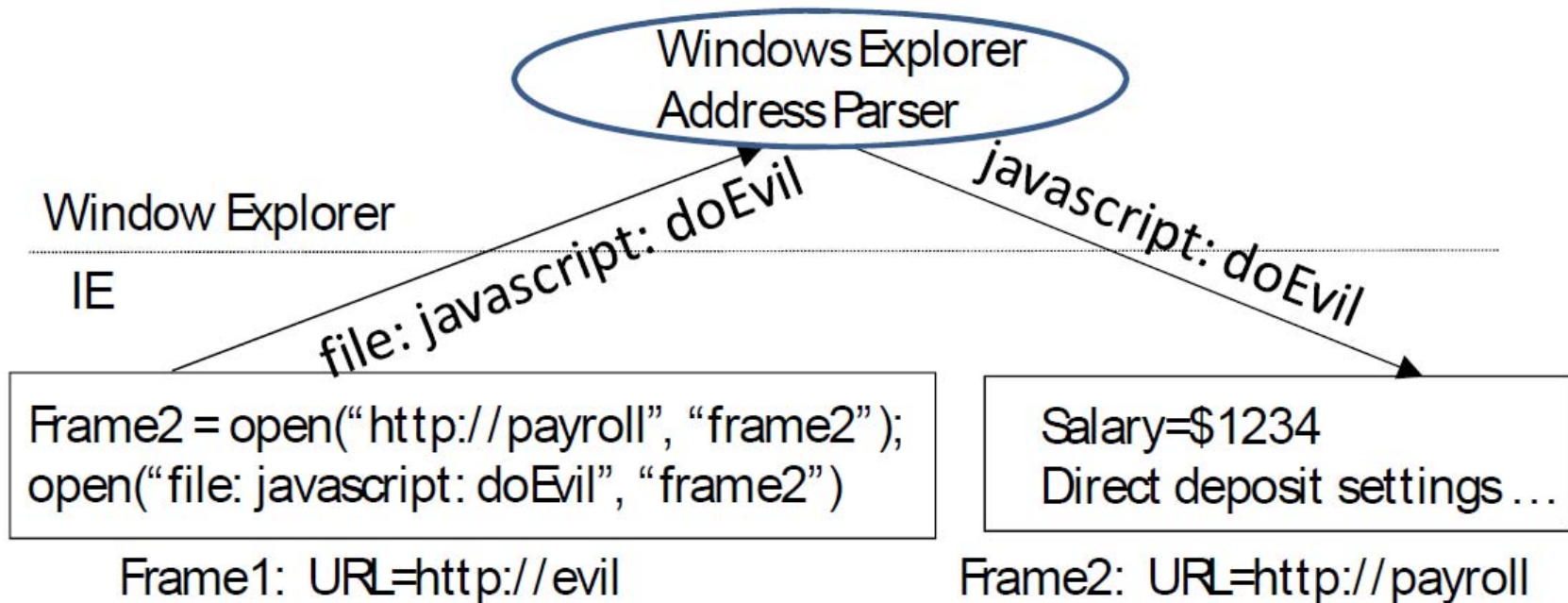
XSS locator 2. If you don't have much space and know there is no vulnerable JavaScript on the page, this string is a nice compact XSS injection check. View source after injecting it and look for <XSS verses <XSS to see if it is vulnerable:

```
' ' ;!--"<XSS>=&{ ( ) }
```

Browser support: [IE7.0|IE6.0|NS8.1-IE] [NS8.1-G|FF2.0] [09.02]

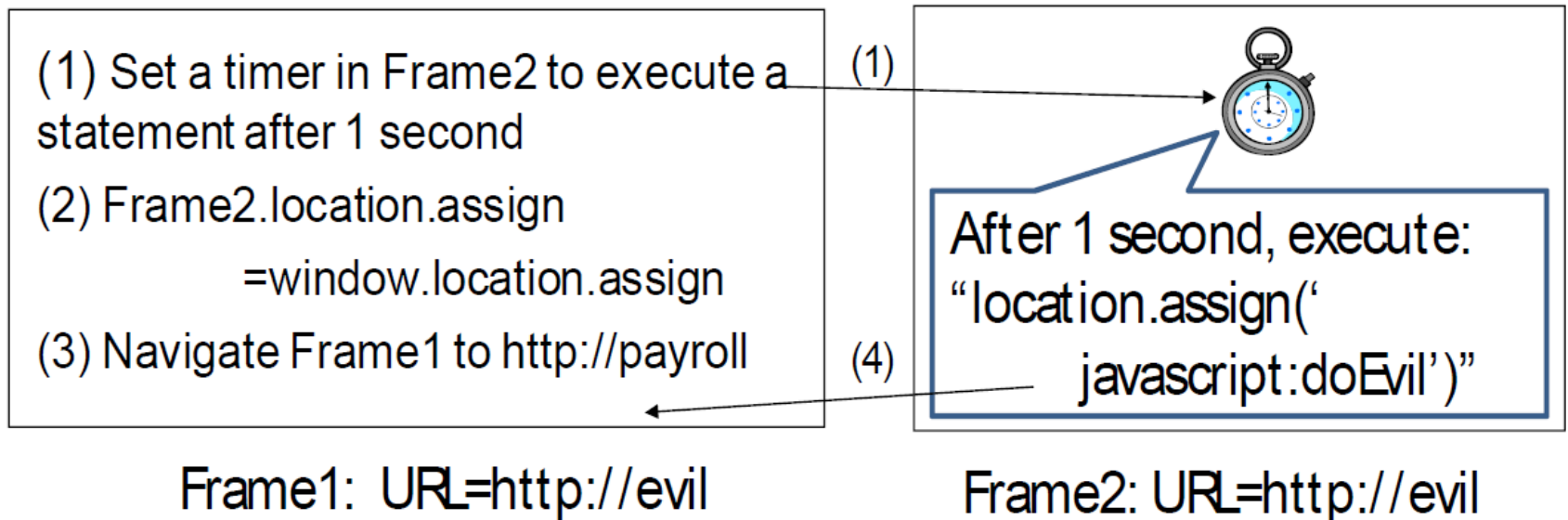
Client Side Problem

- Exploiting the Interactions between IE and Windows Explorer



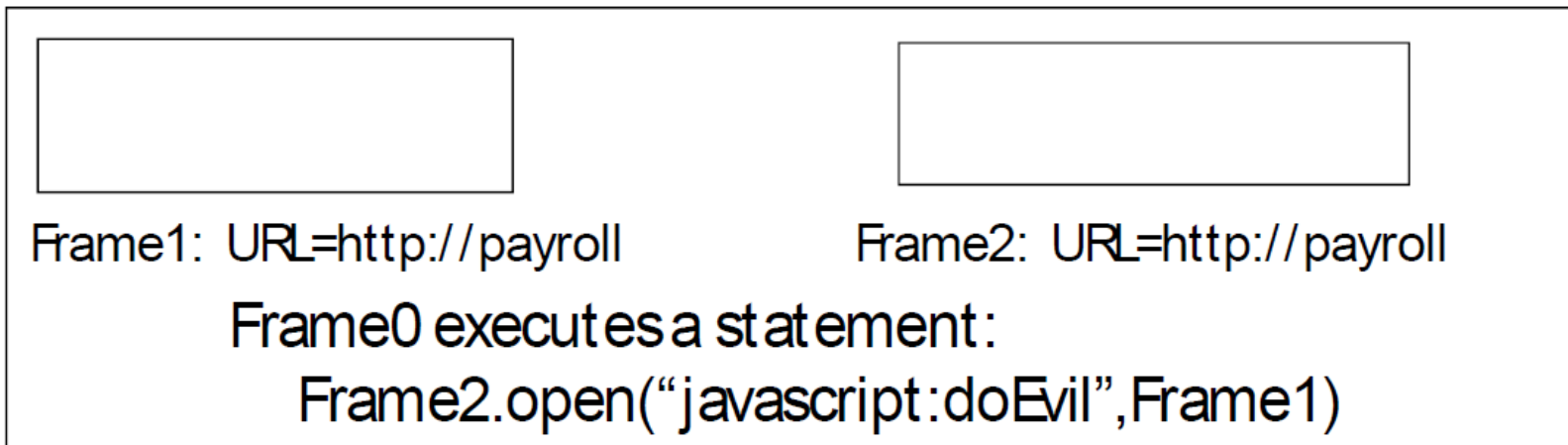
Client Side Problem

Exploiting Function Aliasing



Client Side Problem

- Exploiting the Excessive Expressiveness of Frame Navigation Calls



Frame0: URL=http://evil

Improvement of the Browser



Language Problem

- Argument Exposes Caller

```
function foo(a){  
    bar();  
}
```

```
function bar(){  
    alert(arguments.callee.caller.arguments[0]);  
}
```

```
foo('test');
```

Language Problem

■ Cross-frame poisoning

```
// Frame 1
var stolen;
function fromOtherFrame(o) {
  o.constructor.prototype.valueOf = function () {
    stolen = this;
    return '[Object]';
  };
}
// Frame 1 uses some mechanism like document.domain juggling
// to pass a function to another.
// Frame 2
var secretObject = ...;
fromOtherFrame({ x: 4 });
if (secretObject === ''); // implicitly calls Object.prototype.valueOf
// Frame 2 passes an object containing no sensitive information.
// Frame 1 changes the definition of Frame 2's object, so it can
// steal access to secretObject even though secretObject is never
// explicitly passed to any function, and never has any of its methods
// explicitly called.
```

Language Problem

- More language problem?
- <http://code.google.com/p/google-caja/wiki/AttackVectors>

AttackVectors

Interpreter&Browser properties that can be exploited to escalate priv.

Properties of Interpreters or the Browser allow Privilege Escalation

Below is a list of known attack vectors. We discuss the EcmaScript could allow privilege escalation so that we can come up with tests for

Attack Vectors at the EcmaScript/JavaScript I

- [GlobalObjectPoisoning](#) -- Global object poisoning
- [EvalArbitraryCodeExecution](#) -- eval and the Function constructor
- [ArgumentsMaskedByVar](#) -- function arguments array reassignment
- [CrossScopeParameterModification](#) -- arguments array mutation
- [ArgumentsExposesCaller](#) -- arguments Array and function caller
- [FunctionMemberCrossScopeParameterAccess](#) -- function member access
- [TypeofInconsistent](#) -- typeof inconsistent for regular expressions
- [InaccessibleLocalVariables](#) -- Inaccessible local variables
- [CatchBlocksScopeBleed](#) -- catch blocks may cause global scope bleed
- [GlobalScopeViaThis](#) -- Global scope reachable via this
- [DeleteUnmasksGlobals](#) -- Delete can unmask globals
- [FunctionConstructor](#) -- Function constructor accessible
- [ObjectEvalArbitraryCodeExecution](#) -- Object.eval allow
- [ObjectWatch](#) -- Object.watch allows stealing and poisoning
- [ObjectToSourceLeaksPrivates](#) -- Object.toSource and
- [FunctionMethodsLeakGlobalScope](#) -- Function.call or
- [ConditionalCompilationComments](#) -- Conditional compilation
- [StringObfuscationIsEasy](#) -- Approaches that rely on deobfuscation
- [ParentCircumventsScoping](#) -- The javascript1.2 feature
- [JsControlFormatChars](#) -- [:CF:] can be used to hide code
- [InconsistentlyReservedKeywords](#) -- Different reserved keywords
- [ErrorExposesParameterValues](#) -- The stack property of Error
- [HiddenControlFlowHazard](#) -- Seemingly safe Caja data
- [RegexpLeakMatchGlobally](#) -- Any regular expression
- [EvalBreaksClosureEncapsulation](#) -- Eval extensions allow
- [PostIncrementAndDecrementCanReturnNonNumber](#) -- which property is being accessed
- [MisOptimizations](#) -- Some interpreters try to optimize javascript
- [\(PostIncrementAndDecrementCanReturnNonNumber\)](#) is
- [CompoundAssignmentsCanReturnNonNumber](#) -- The t

- What we have now

Same Origin Policy

- The only built-in” safety mechanism
 - Simple but very powerful
- All or nothing solution
- Not act very well in Web 2.0
- Violates the Principle Of Least Authority, need to be improved

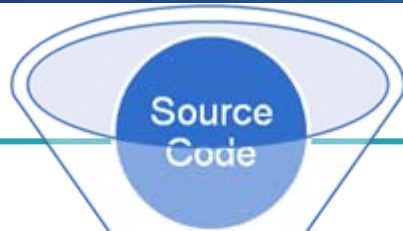
New Approach

- (I'm trying to do right now)
- **Data Centric Policy Enforcement**
 - The sensitive information is in data. Secure sensitive information from the core
 - Every read or write operation on data will be checked
 - Policy should be bounded with data
 - Achieve POLA
 - Not vulnerable to unknown attack. No way to get around, no worry on new attack method
 - Could pass data with policy

Data Centric Policy Enforcement

- Translate JavaScript at runtime from Abstract Syntax
- Wrap every data object to perform the check at the point of every read & write
- Combine with the policy defined independently





Js Parser

AST



```
var a=0;

//Original Program
function f(){
    print('a=',a);
    print('a=a+1, a=', a+=1);
}

f();
```



```
var A=new ProtectedObject(0, pol, 'a', typeof(0));

function F(){
    var id=arguments.callee.name;
    Trans('print', new ProtectedObject('a='),A /* , ... */);

    //a=a+1;
    Trans('print', new ProtectedObject('a=a+1, a='), Trans('=', A, Trans('+', A, new ProtectedObject(1))));
}

F();
```

What can we do with DCPE

- Very fine control over the policy
 - Data A could allow B to read/write some of its properties/methods
- Not vulnerable to unknown attack
 - As long as this philosophy holds
- Combine data with policy and transmit them together

THANKS