A Collection of Real World (JavaScript) Security Problems

Examples from 2$\frac{1}{2}$ Applications Areas of JavaScript

Achim D. Brucker
achim.brucker@sap.com

SAP AG, Central Code Analysis Team

July 2, 2014

A Collection of Real World (JavaScript) Security Problems

Abstract

JavaScript is gaining more and more popularity as an implementation language for various applications types such as Web applications (client-side), mobile applications, or server-side applications.
We outline a few security challenges that need to be prevented in such applications and, thus, for which there is a demand for analysis methods that help to detect them during during development.

Agenda

1 Motivation and Basics

2 SAP UI5: Client-side JavaScript

3 Apache Cordova: JavaScript on Mobile

4 HANA XS Engine: Server-side JavaScript

Agenda

1 Motivation and Basics

2 SAP UI5: Client-side JavaScript

3 Apache Cordova: JavaScript on Mobile

4 HANA XS Engine: Server-side JavaScript

What We Want to Find

Programming Patterns That May Cause Security Vulnerabilities

Mainly two patterns

Local issues (no data-flow dependency), e. g.,

- Insecure functions

```
var x = Math.random();
```

- Secrets stored in the source code

```
var password = 'secret';
```

Data-flow related issues, e. g.,

- Cross-site Scripting (XSS)

```
var docref  = document.location.href;
var input = docref.substring(
              docref.indexOf("default=")+8);
var fake = function (x) {return x;}
var cleanse = function (x) {
                return 'hello␣world';}
document.write(fake(input));
document.write(cleanse(uinput));
```

- Secrets stored in the source code

```
        var foo = 'secret';
        var x = decrypt(foo,data);
```

Functions as First-Class Objects

```
var href = document.location.href;
var unsafeInput = href.substring(href.indexOf("default=")+8) // unsafe input
var safeInput = "1+2";                                       // safe input

// aliasing eval
var exec = eval;
var doit = exec;

var func_eval1     = function (x) {eval(x);};
var func_eval2     = function (x,y) {eVaL(y);};

var func_eval_eval = function (x) {func_eval1(x);};
var func_doit      = function (x) {doit(x);};
var func_exec      = function (x) {exec(y);};
var run            = func_eval1;
var inject_code    = func_exec;

doit(safeInput);   // secure
doit(unsafeInput); // code injection
```

Where is The Code of my Application?

```
var input  = document.location.href.substring(document.location..indexOf("default=")+8);
var fake = function (x) {return x;}
var cleanse = function (x) {return 'hello␣world';}

var uinput = unknown(input); // unknown is nowhere defined
document.write(uinput); // secure!?

var finput = fake(input);
document.write(finput);  // not secure

var cinput = cleanse(input);
document.write(cinput); // secure

var extfinput = extfake(input);  // defined externally (part of scan)
document.write(extfinput);  // not secure

var extcinput = extcleanse(input); defined externally (part of scan)
document.write(extcinput); // secure

var nobodyKnows = toCleanOrNotToCleanse(input); multiply defined (underspecified)
document.write(nobodyKnows); // not secure!?
```

Agenda

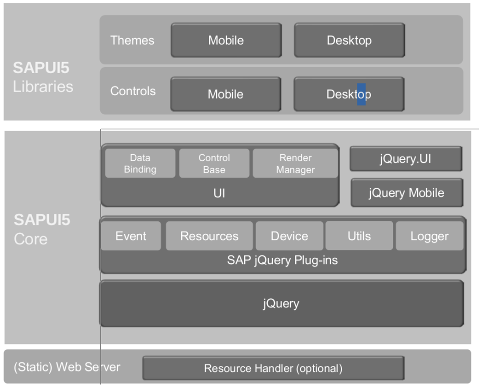
1 Motivation and Basics

2 SAP UI5: Client-side JavaScript

3 Apache Cordova: JavaScript on Mobile

4 HANA XS Engine: Server-side JavaScript

The SAP UI5 Architecture



Prototype-based Inheritance

```
var vl = new sap.ui.commons.layout.VerticalLayout();
sap.ui.core.Control.extend("foo.Label", {
    metadata : {
        properties : {
            "text" : "string"
        }
    },
    renderer : function(oRm, oControl) {
        oRm.write("<span>XSSLabel:␣");
        oRm.write(oControl.getText());
        oRm.write("</span>");
    }
});
var p = jQuery.sap.getUriParameters().get("xss");
vl.addContent(new foo.Label({text:p}));
return vl;
```

CSRF Prevention

You need to know your frameworks

```
var request = {
        headers : {
                "X-Requested-With" : "XMLHttpRequest",
                "Content-Type" : "application/atom+xml",
                "X-CSRF-Token" : "Fetch"
        },
};
if (Appcc.CSRFToken)
        var request = {
                headers : {
                        "X-Requested-With" : "XMLHttpRequest",
                        "Content-Type" : "application/atom+xml",
                        "X-CSRF-Token" : Appcc.CSRFToken
                },
        };
else  var request = {
                headers : {
                        "X-Requested-With" : "XMLHttpRequest",
                        "Content-Type" : "application/atom+xml",
                        "X-CSRF-Token" : "etch"  // secure?
                },
        };
```

Agenda

1 Motivation and Basics

2 SAP UI5: Client-side JavaScript

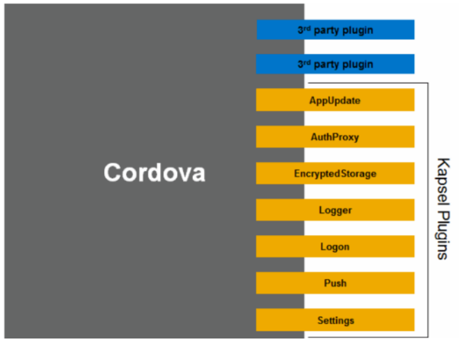
3 Apache Cordova: JavaScript on Mobile

4 HANA XS Engine: Server-side JavaScript

Apache Cordova (SAP Kapsel): Overall Idea

An integrated platform for developing hybrid mobile apps

- Apache Cordova plus
  - App management
  - Encrypted Storage
  - Authentication
  - Logging
  - . . .
- Application management (SMP)
- Can be used with device management solutions



Exploiting the JavaScript to Java Bridge

- We can expose Java methods in JavaScript

```
        foo.addJavascriptInterface(new FileUtils(), "FUtil");
```

- And use them in JavaScript easily

```
<script type="text/javascript">// <![CDATA[
filename = '/data/data/com.livingsocial.www/' + id +'_cache.txt';
FUtil.write(filename, data, false);
// ]]></script>
```

- Which might expose much more than expected

```
  function execute(cmd){
     return
     window._cordovaNative.getClass().forName('java.lang.Runtime').
                    getMethod('getRuntime',null).invoke(null,null).exec(cmd);
  }
```

Agenda

1 Motivation and Basics

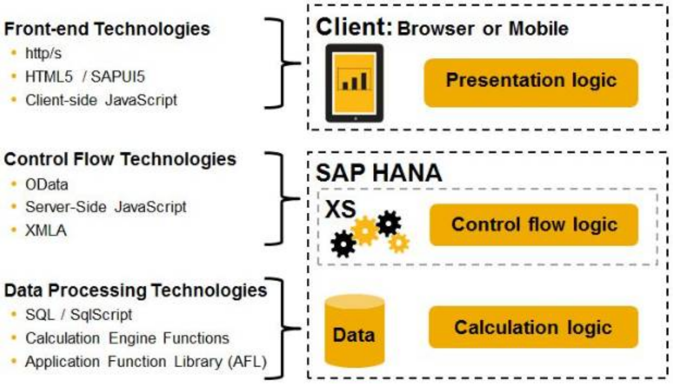
2 SAP UI5: Client-side JavaScript

3 Apache Cordova: JavaScript on Mobile

4 HANA XS Engine: Server-side JavaScript

The HANA XS Engine Architecture

Overview



History Repeats: SQL Injection

```
$.response.contentType = "text/html";
var userInput = $.request.parameters.get(’userStuff’);

// We assume
// - $.db.getConnection().prepareStatement(x0, ..., xn) is secure iff x0 is *not*
//   influenced by user input
// - sql_sanitize() safeguards us against SQL injections.
// - any other preparedStatement call is evil regardless if it is influenced by
//   user input or not

if (userInput) {

        var sql      = "select␣*␣from␣SFLIGHT.SNVOICE␣where␣CustomID␣=’"
                     + userInput + "’";
        var safe_sql = "select␣*␣from␣SFLIGHT.SNVOICE␣where␣CustomID␣=’"
                     + sql_sanitize(userInput) + "’";

        var db_object = $.db;
        var conn      = db_object.getConnection();

        var pstmt00 = $.db.getConnection().prepareStatement(sql);              // SQL injection
        var pstmt01 = $.db.getConnection().prepareStatement(safe_sql);         // secure
```

History Repeats: SQL Injection

```
var sql      = "select␣*␣from␣SFLIGHT.SNVOICE␣where␣CustomID␣=’"
             + userInput + "’";
var safe_sql = "select␣*␣from␣SFLIGHT.SNVOICE␣where␣CustomID␣=’"
             + sql_sanitize(userInput) + "’";

var db_object = $.db;
var conn      = db_object.getConnection();

var pstmt00 = $.db.getConnection().prepareStatement(sql);              // SQL injection
var pstmt01 = $.db.getConnection().prepareStatement(safe_sql);         // secure

var pstmt02 = db_object.getConnection().prepareStatement(sql);         // SQL injection
var pstmt03 = db_object.getConnection().prepareStatement(safe_sql);    // secure

var pstmt04 = conn.prepareStatement(sql);                              // SQL injection
var pstmt05 = conn.prepareStatement(safe_sql);                         // secure

var pstmt06 = conn.prepareStatement("...␣where␣ID␣=␣’$1’",userInput);   // secure
var pstmt07 = myconn.prepareStatement("...␣where␣ID␣=␣’$1’",userInput); // SQL injection

var pstmt08 = $.mydb.getConnection().prepareStatement(sql);            // SQL injection
var pstmt09 = $.mydb.getConnection().prepareStatement(safe_sql);       // SQL injection
```

© 2014 SAP AG. All rights reserved