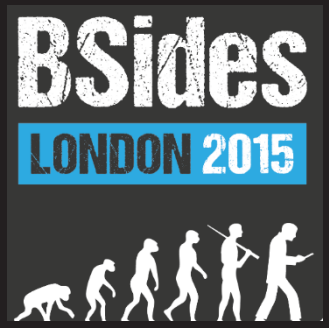
Why bother assessing popular software?

David Middlehurst
James Loureiro



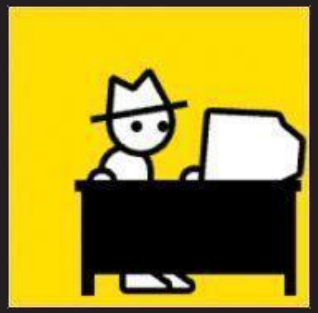
3rd June 2015

BSides London

whoami

David Middlehurst - @dtmsecurity

Simulated Attacks, Application Security, Tool Development, Vulnerability Research



James Loureiro - @nerdkernel

Vulnerability Research, Reverse Engineering, Industrial Control Systems (ICS), Embedded Systems

Introduction

“Making a difference - What is actually going to make systems significantly more secure? More of the same doesn't seem to be working.”

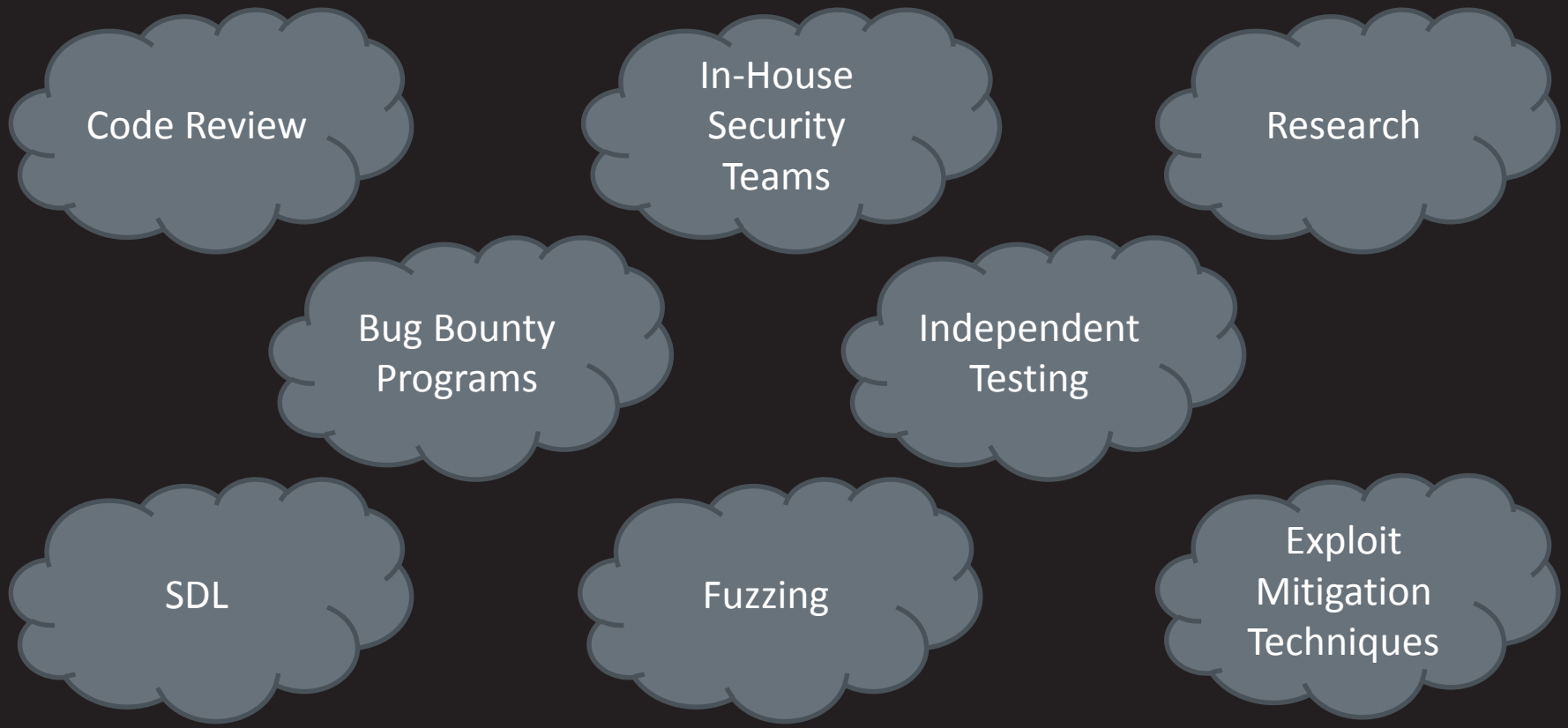
BSides London 2015 CFP

Agenda

- **Software security today**
- **Case Study: Adobe Reader**
  - Attack Surface
  - JavaScript API
  - Fuzzing
  - Sandbox
  - Mitigations
- Conclusion

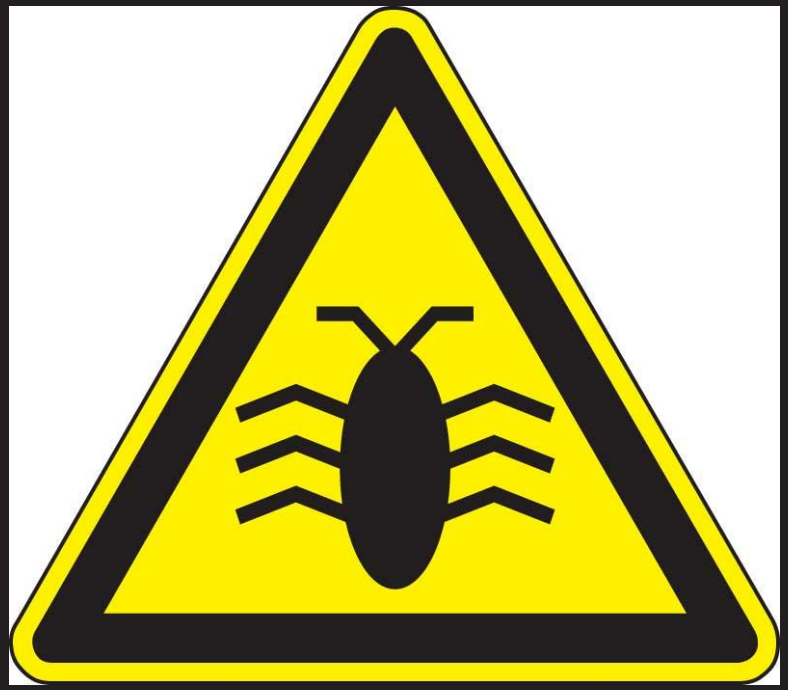
Introduction

How are vendors improving software security?



Why bother assessing popular software?

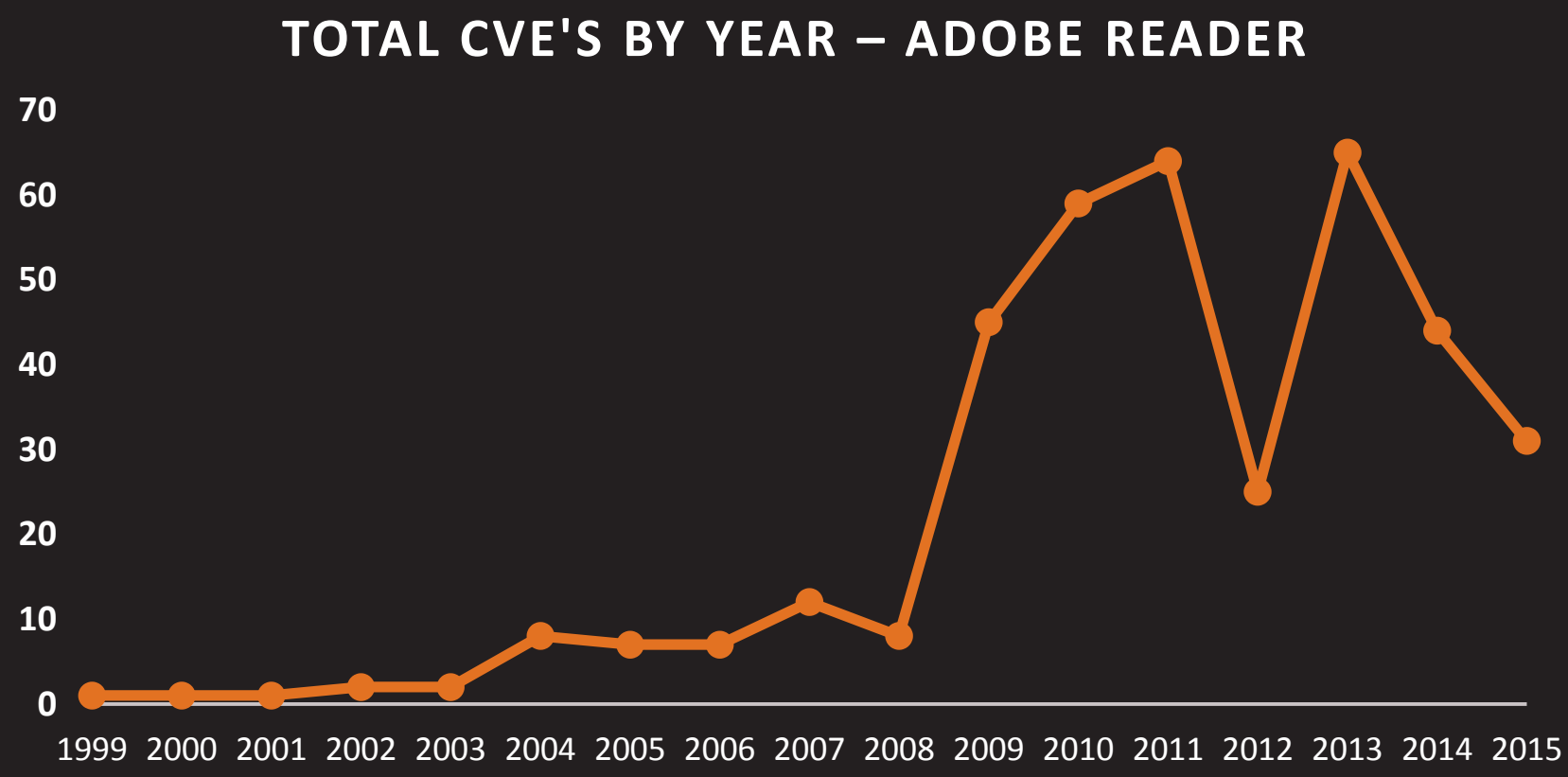
- Still bugs to be found?



Adobe Reader

- Implicitly trusted by enterprise and home users
- Adobe Reader >80% market share
- PDFs are trusted

Bugs are still found



Bugs are still found

Year	High Risk CVEs
2009	33
2010	62
2011	49
2012	30
2013	65
2014	37
2015 (To date)	31

Types of vulnerabilities

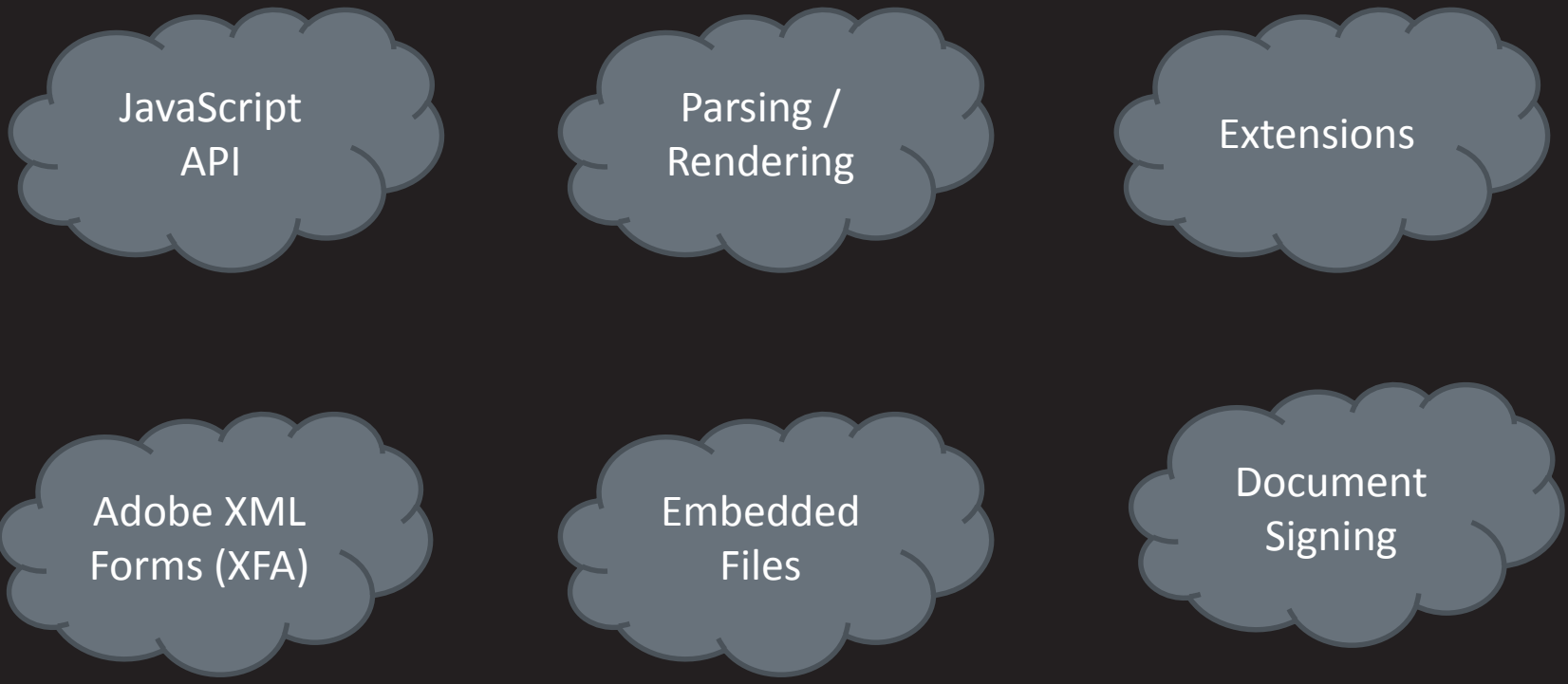
APSB15-10

Category	CVEs
JavaScript API	14
Memory Corruption	10
Use-after-free	5
Heap-based buffer overflow	1
Buffer Overflow	1
XXE	1

Useful Resources

- Adobe Standards / Documentation / Engineering Team
- Corkami PDF101
- Didier Stevens PDF Tools / Research
- @MOLNAR_G - The Life of an Adobe Reader JavaScript bug
- Fortinet “Breeding Sandworms” - BH Europe 2012
- “When the broker is broken” – CanSecWest 2013
- Government Hardening Guidelines (NSA / DSD)

Attack Surface



Attack Surface - JS

"JavaScript in Adobe Acrobat software implements objects, methods, and properties that enable you to manipulate PDF files, produce database-driven PDF files, modify the appearance of PDF files, and much more."

- Based on JavaScript version 1.5
- ECMAScript

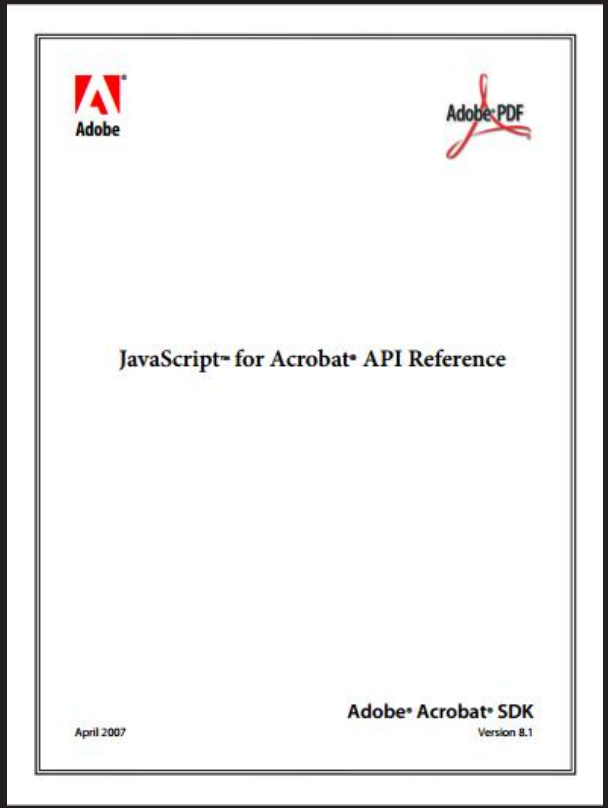
Attack Surface - JS

Read:

- JavaScript API Documentation
- Decompile SpiderMonkey bytecode

Play with JavaScript:

- Make PDFs with JS embedded
- JavaScript Console

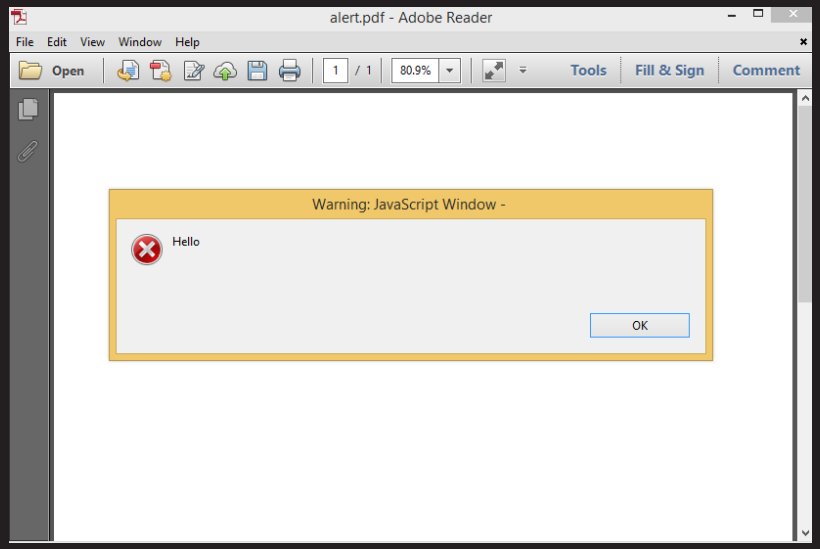


JS – Simple Tricks

```
%PDF-1.5
%
1 0 obj<</Type/Catalog/Outlines 2 0 R/Pages 3 0 R/OpenAction 5 0 R>>endobj
2 0 obj<</Type/Outlines/Count 0>>endobj
3 0 obj<</Type/Pages/Kids[4 0 R]/Count 1>>endobj
4 0 obj<</Type/Page/Parent 3 0 R/MediaBox[0 0 612 792]>>endobj
5 0 obj<</Type/Action/S/JavaScript/JS 6 0 R>>endobj
6 0 obj<</Length 565>>
stream
```

```
app.alert('Hello');
```

```
endstream
endobj
xref
0 7
0000000000 65535 f
0000000013 00000 n
0000000089 00000 n
0000000130 00000 n
0000000180 00000 n
0000000244 00000 n
0000000297 00000 n
trailer<</Size 7/Root 1 0 R>>
startxref
915
%%EOF
```



JS – Simple Tricks

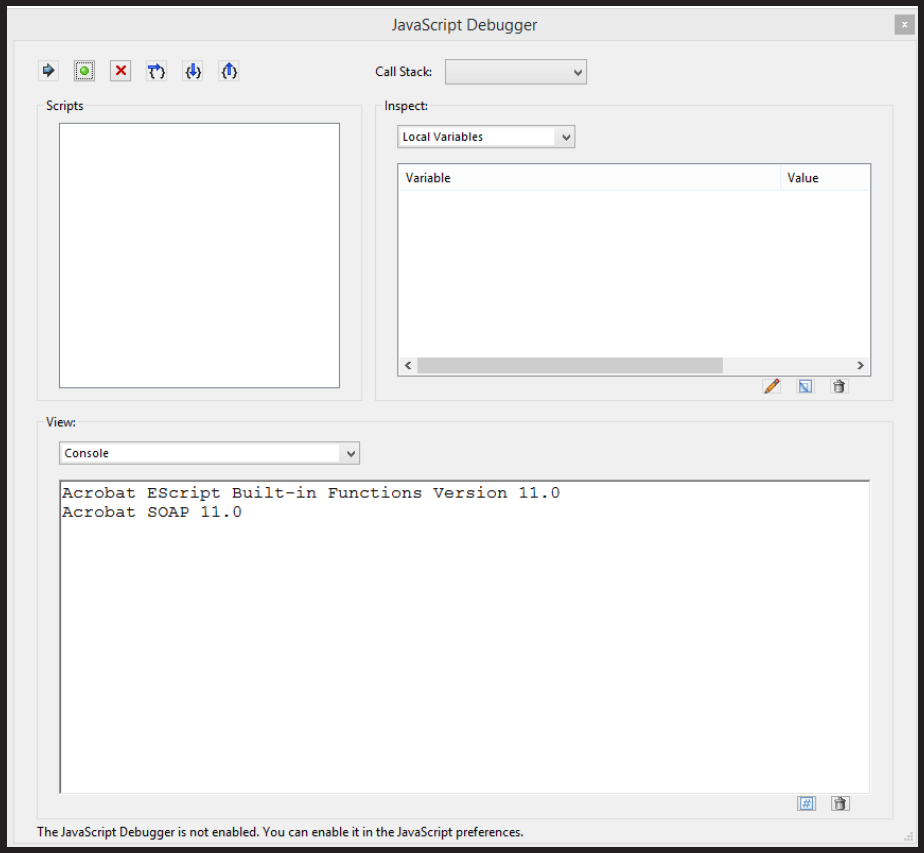
```
Usage: make-pdf-javascript.py [options] pdf-file

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -j JAVASCRIPT, --javascript=JAVASCRIPT
                        javascript to embed (default embedded JavaScript is
                        app.alert messagebox)
  -f JAVASCRIPTFILE, --javascriptfile=JAVASCRIPTFILE
                        javascript file to embed (default embedded JavaScript
                        is app.alert messagebox)

  make-pdf-javascript, use it to create a PDF document with embedded JavaScript
that will execute automatically when the document is opened
  Source code put in the public domain by Didier Stevens, no Copyright
  Use at your own risk
  https://DidierStevens.com
```

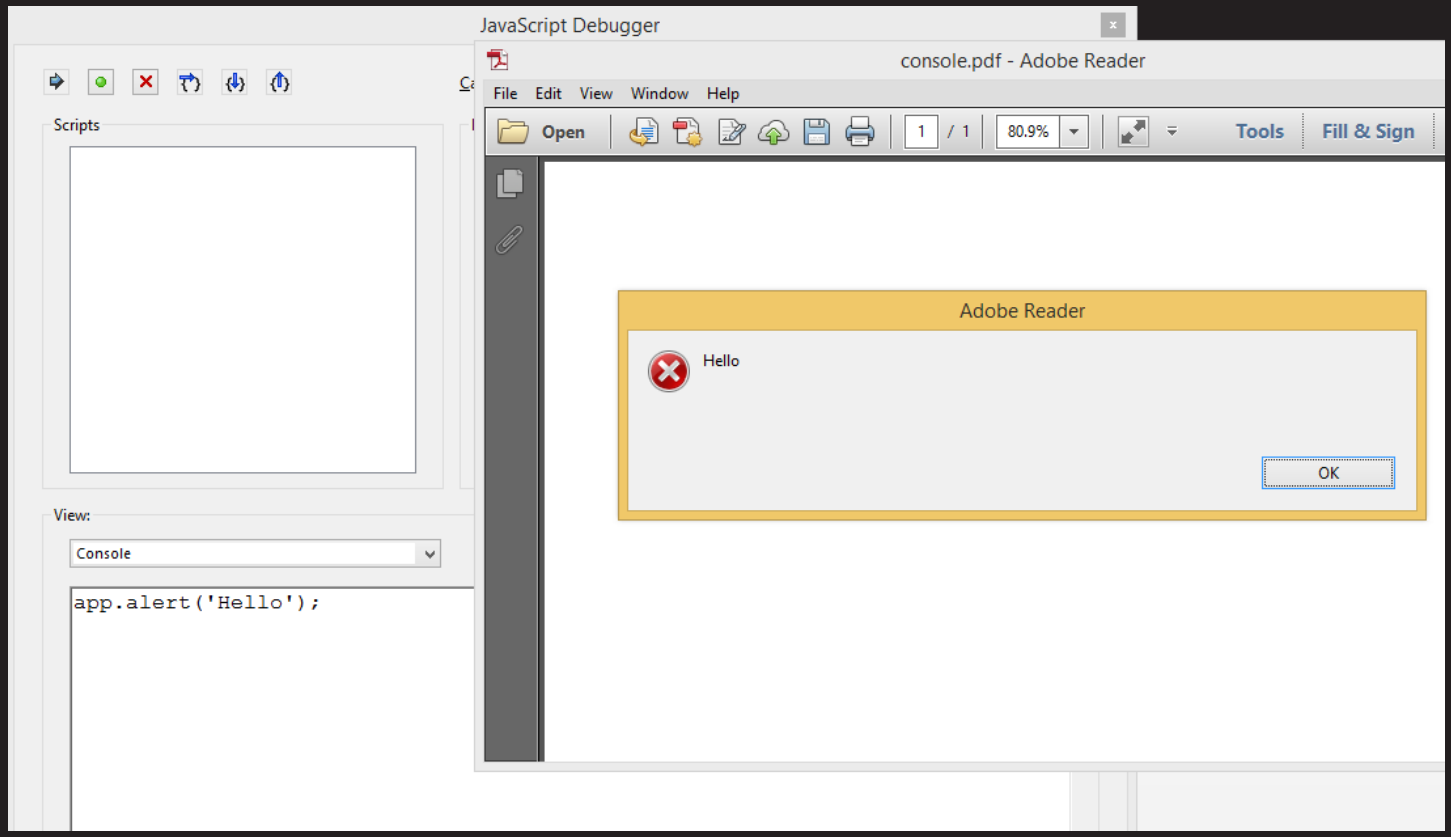
JS – Simple Tricks

```
console.show();
```



JS – Simple Tricks

app.alert('Hello'); <CTRL-ENTER>



JS – Simple Tricks

```
app.alert(acrohelp);

====> cMsg: string
====> [nIcon: integer]
====> [nType: integer]
====> [cTitle: string]
====> [oDoc: object]
====> [oCheckbox: object]
```

JS – Introspection

var output = ""; for(var a in app) { output = output + ", " + a; }; console.println(output);

toolbar, toolbarVertical, toolbarHorizontal, language, viewerType, viewerVersion, platform, openInPlace, activeDocs, viewerVariation, printerNames, printColorProfiles, addressBookAvailable, alert, beep, response, goBack, goForward, popUpMenu, popUpMenuEx, execMenuItem, hideMenuItem, hideToolbarButton, addMenuItem, addSubMenu, listMenuItems, listToolbarButtons, browseForDoc, browseForMultipleDocs, mailMsg, mailMsgWithAttachment, getResolvedAddresses, mailGetAddrs, newDoc, openDoc, setTimeOut, clearTimeOut, setInterval, clearInterval, getString, getPath, setProfile, trustedFunction, trustPropagatorFunction, beginPriv, endPriv, launchURL, isValidSaveLocation, constants, user, plugIns, numPlugIns, getNthPlugInName, fs, fsUseTimer, fsUsePageTiming, fsLoop, fsEscape, fsClick, fsTransition, fsTimeDelay, fsColor, fullscreen, fsCursor, thermometer, capabilities, openFDF, newFDF, exportFiles, runtimeHighlight, calculate, formsVersion, focusRect…..

JS – Simple Tricks

console.println(app.alert.toString());

```
function alert() {
    [native code]
}
```

console.println(app.media.getAnnotStockEvents.toString());

```
function (windowType) {
    var events = new (app.media.Events);
    if (app.media.trace) {
        events.add(app.media.getAnnotTraceEvents());
    }
    events.add({onDestroy: function (e) {if (e.target.player) { .....
```

Attack surface - JS

- Developed PoC tool
- Interacts with JavaScript Console via Win32 API
- Enumeration
- Fuzzing

```
IntPtr javascriptDebugger = FindWindow("#32770",
"JavaScript Debugger");

IntPtr classHandle = EnumAllWindows(javascriptDebugger,
"RICHEDIT50W").ElementAt(1);
```

Attack surface - JS

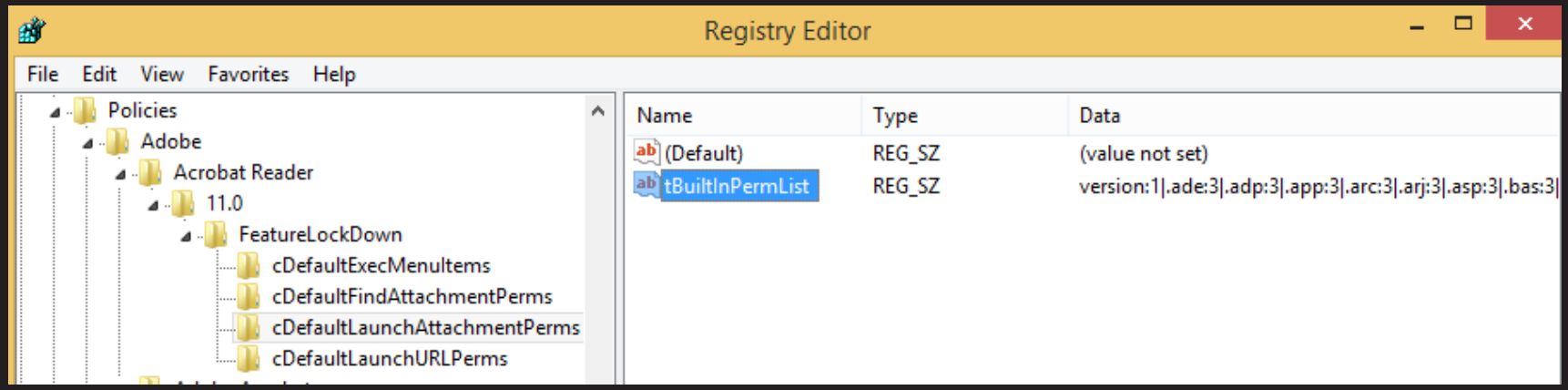
DEMO

Attack surface - JS

- Decompile SpiderMonkey bytecode
- Prettified >27,000 lines of JavaScript

Attack surface – Embedding Files

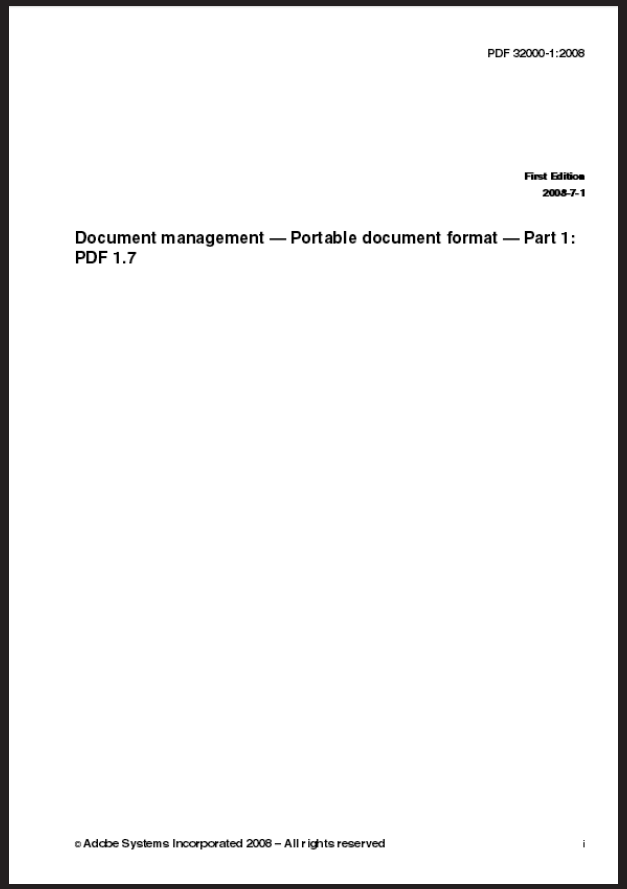
HKLM\SOFTWARE\Policies\Adobe\Acrobat Reader\11.0\FeatureLockDown



.ade:3|.adp:3|.app:3|.arc:3|.arj:3|.asp:3|.bas:3|.bat:3|.bz:3|.bz2:3|.cab:3|.chm:3|.class:3|.cmd:3|.com:3|.command:3|.cpl:3|.crt:3|.csh:3|.desktop:3|.dll:3|.exe:3|.fxp:3|.gz:3|.hex:3|.hlp:3|.hqx:3|.hta:3|.inf:3|.ini:3|.ins:3|.isp:3|.its:3|.job:3|.js:3|.jse:3|.ksh:3|.lnk:3|.lzh:3|.mad:3|.maf:3|.mag:3|.mam:3|.maq:3|.mar:3|.mas:3|.mat:3|.mau:3|.mav:3|.maw:3|.mda:3|.mdb:3|.mde:3|.mdt:3|.mdw:3|.mdz:3|.msc:3|.msi:3|.msp:3...

PDF Rendering Engine

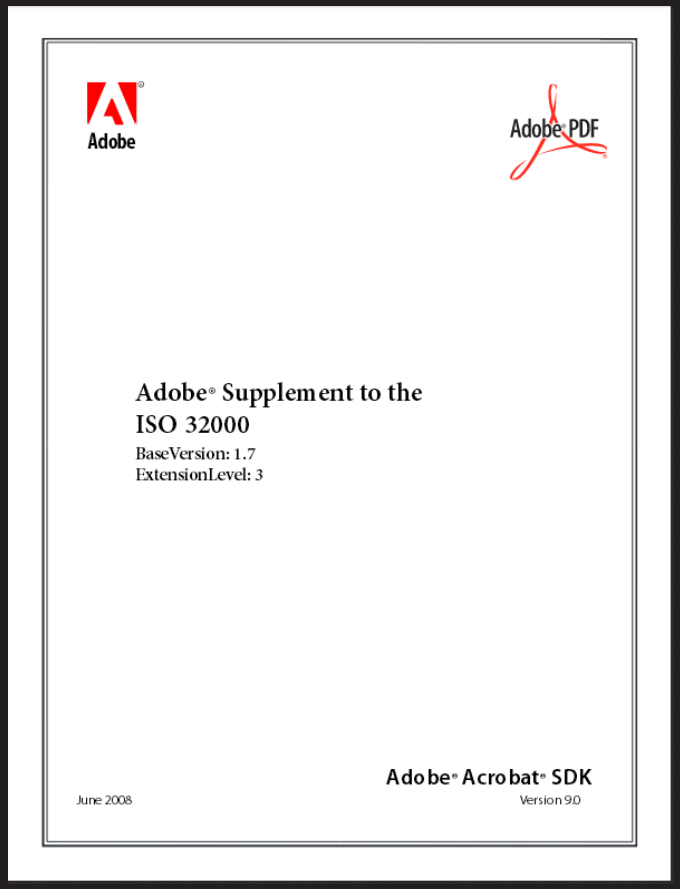
Big attack surface here…



PDF 32000-1:2008

First Edition
2008-7-1

Document management — Portable document format — Part 1: PDF 1.7



Adobe

Adobe PDF

Adobe® Supplement to the ISO 32000

BaseVersion: 1.7
ExtensionLevel: 3

Adobe® Acrobat® SDK
Version 9.0

June 2008

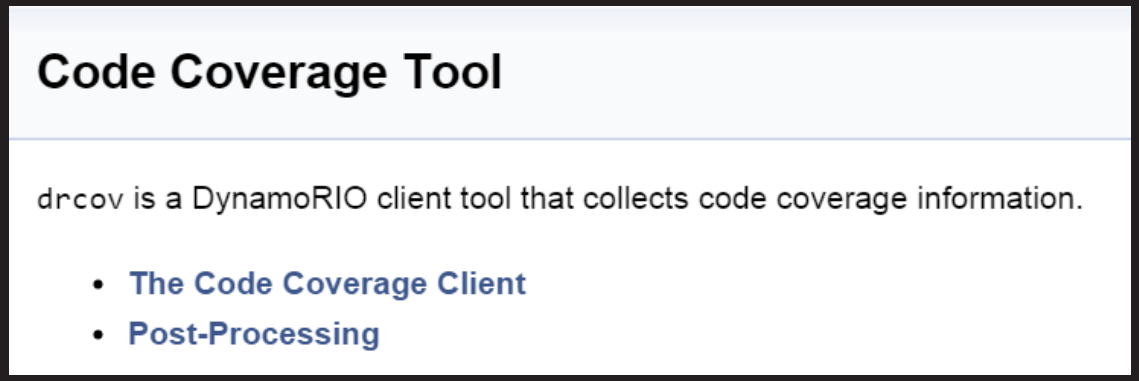
Attack surface – PDF files

Code coverage for fuzzing is important

Getting the coverage…

We can generate the PDF's ourselves…

WGET -r theinterwebz FTW!



Attack surface – PDF files

Expand PDF’s

Attack surface – PDF files

But why bother fuzzing – is that not what everyone is doing?

Need to fuzz smarter!

- Get as much coverage as possible – see AFL
- Be distributed

We have found crashes using this method, setup fuzzer and ran for about a week…

Attack surface – PDF files

Fuzz!

Simple bit flip used

Bugs (so far…)

1 x UAF

1 x Null Pointer Dereference

A lot of boring rubbish ones

Not finished triage process (~Around 100 crashes here)

Attempting to exploit UAF – watch out for POC ☺

UAF

Not yet been patched by Adobe…

Error in way font library is handled

```
0:000> !heap -p -a esi
    address 3280ae88 found in
    _DPH_HEAP_ROOT @ 89d1000
    in busy allocation (  DPH_HEAP_BLOCK:         UserAddr         UserSize -         VirtA
                                32733dd0:         3280add8              228 -         3280a
*** WARNING: Unable to verify checksum for C:\Program Files (x86)\Adobe\Reader 11.0\Reader\
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\Progran File
          ? AcroForm!std::_Init_locks::operator=+27f20a
    523d94ec verifier!AVrfDebugPageHeapAllocate+0x0000023c
    770a57b7 ntdll!RtlDebugAllocateHeap+0x0000003c
    770477ce ntdll!RtlpAllocateHeap+0x0004665a
    77001134 ntdll!RtlAllocateHeap+0x0000014d
    711f0269 MSVCR100!malloc+0x0000004b
    711f233b MSVCR100!operator new+0x0000001f
    24d5acc5 AcroForm!PlugInMain+0x00068778
    24d5a513 AcroForm!PlugInMain+0x00067fc6
    24d57d3b AcroForm!PlugInMain+0x000657ee
    24fba653 AcroForm!DllUnregisterServer+0x00121e94
```

Reader Mitigations – an overview

Mitigations

Sandbox

Previous work on Adobe 10

- Blackhat EU – Breeding Sandworms
- CanSecWest – When the broker is broken

Both really good presentations on Reader Sandbox

Both still applicable in Reader 11.

Based on Chrome Sandbox, but with a lot more calls to support ‘rich’ feature set.

A brief look at the Sandbox

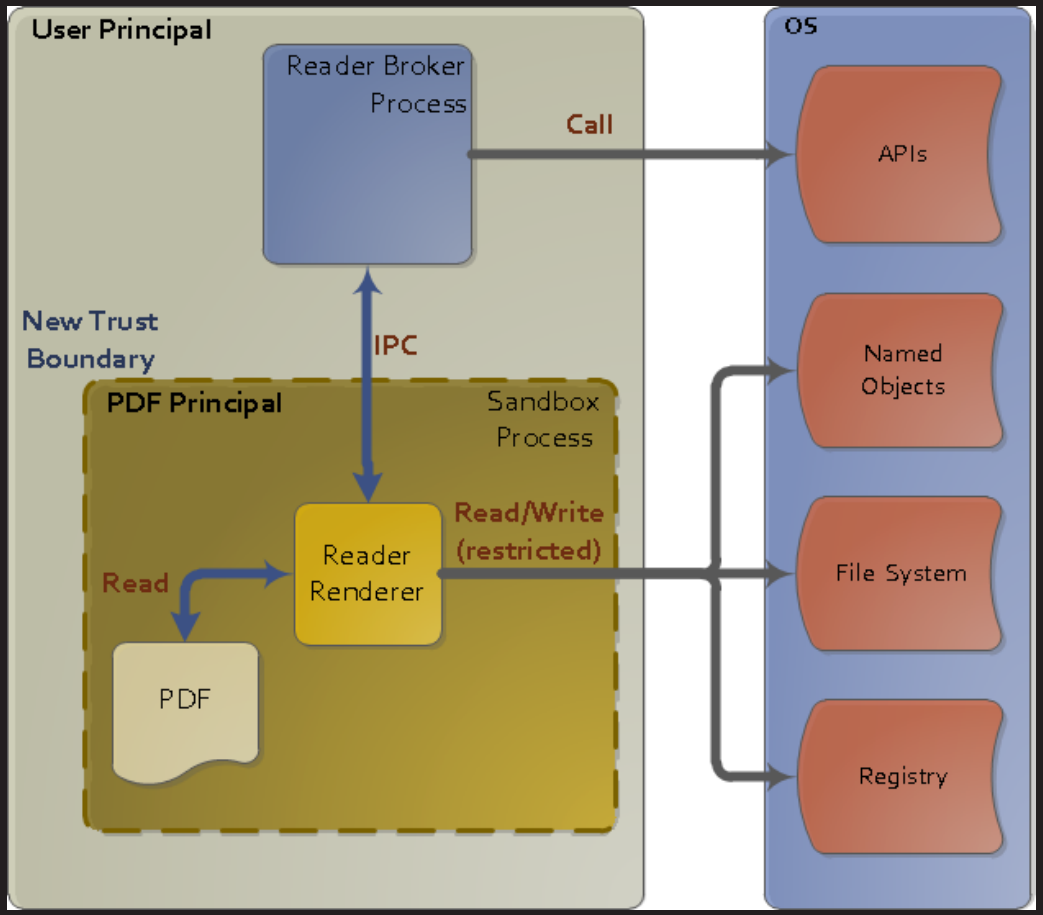


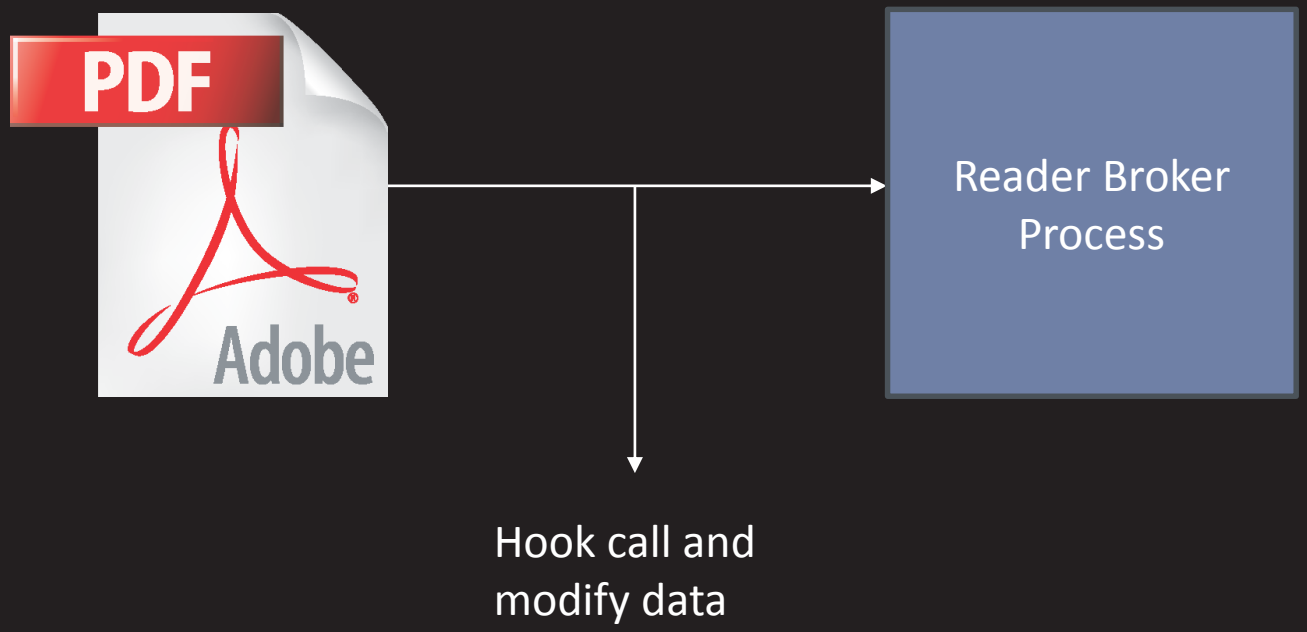
Image from http://blogs.adobe.com/security/2010/10/inside-adobe-reader-protected-mode-part-1-design.html

Sandbox – how to escape?

A number of areas:

- Kernel 0-days
- Logical flaws in cross calls
- Memory corruption in cross calls
  - Sandbox fuzzer

Sandbox fuzzing



JS privileges

- Mitigation implemented for JavaScript API
- Privilege vs. Non Privileged Context

JS privileges

Privileged versus non-privileged context

Some JavaScript methods, marked in this book by Ⓢ in the third column of the quick bar, have security restrictions. These methods can be executed only in a *privileged context*, which includes console, batch and application initialization events. All other events (for example, page open and mouse-up events) are considered *non-privileged*.

launchURL

7.0		Ⓢ	

Launches a URL in a browser window.

Note: This method does not support URLs that begin with either scheme name `javascript` or `file`.

JS privileges

```
foo = app.trustedFunction(
    function(bar) {
        app.beginPriv();
         <Privileged Stuff>
        app.endPriv();
    }
);
```

MWR LABS

JS privileges

PoC DEMO

What have we learned?

- Understand the attack surface
- How this can be reduced through hardening
- Other security controls which be implemented to mitigate risks?

Conclusion

- Identified a number of bugs
  - Use-after-free
  - JavaScript Privilege Escalation
  - More to triage

Future Work

- We are applying this methodology to other products
- Tactics seem to be working!
- Bugs already found in Microsoft Visio

Thanks to

- Nils and Yong at MWR for their help
- BSides crew

Questions?

@dtmsecurity

@NerdKernel

@mwrlabs