binamuse
security research

Core GraphicsMemory Corruption CVE-2014-4377

PDF Indexed colorspace buffer overflow

Apple CoreGraphics library fails to validate the input when parsing the colorspace specification of a PDF XObject resulting in a heap overflow condition. A small heap memory allocation can be overflowed with controlled data from the input in any application linked with the affected library. Using a crafted PDF file as an HTML image and combined with a information leakage vulnerability this issue leads to arbitrary code execution.

September 2014

Felipe Andres Manzano
feliam@binamuse.com

Contents

1 Target summary

Title: Apple CoreGraphics Framework memory corruption

CVE Name: CVE-2014-4377

Product: Apple IOS/OSX operating systems

Version: 6.1.x , 7.0.x, 7.1.x

Product Homepage: apple.com

Advisory: `http://support.apple.com/kb/ht6443`

2 Vulnerability brief information

Vulnerability Class	Memory Corruption
Affected Versions	6.1.x, 7.0.x, 7.1.x
Affected Platforms	iPod4,1 iPhone3,1 iPhone4S iPhone5 iPhone5c iPhone (m68ap) iPhone 3G (n82ap) iPhone 3GS (n88ap) iPhone 4(n90ap,n90bap,n92ap) iPhone 4S (n94ap) iPhone 5(n41ap,n42ap) iPhone 5c(n48ap,n49ap) iPod touch (n45ap) iPod touch 2G (n72ap) iPod touch 3G (n18ap) iPod touch 4G (n81ap) iPod touch 5G(n78ap,n78aap) iPad (k48ap) iPad 2(k93ap,k94ap,k95ap,k93aap) iPad 3(j1ap,j2ap,j2aap) iPad 4(p101ap,p102ap,p103ap) iPad mini 1G(p105ap,p106ap,p107ap) Apple TV 2G (k66ap) Apple TV 3G(j33ap,j33iap)
Reliability Rating	Completely (100%)

Supported Targets	1: iPod4,1 iOS-6.1.5 2: iPod4,1 iOS-6.1.6 3: iPhone3,1 iOS-7.0.4 5: iPhone4S iOS-7.1 6: iPhone4S iOS-7.1.1 7: iPhone5 iOS-7.1 8: iPhone5 iOS-7.1.1
Attack Vector	Client-Side File Format
Exploitation Impact	Code Execution
Exploitation Context	desktop/mobile Browser

3 Common Vulnerability Scoring System

Base Metrics		
Access **V**ector	**N**etwork	The vulnerability is exploitable with network access
Access **C**omplexity	**L**ow	Specialized access conditions or extenuating circumstances do not exist
Authentication	**N**one	Authentication is not required to exploit the vulnerability.
Confidentiality Impact	**C**omplete	There is total information disclosure, resulting in all system files being revealed
Integrity Impact	**P**artial	Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited
Availability Impact	Partial (P)	There is reduced performance or interruptions in resource availability

Temporal Metrics		
Exploitability	**F**unctional	Functional exploit code is available. The code works in most situations where the vulnerability exists
Remediation **L**evel	**O**fficial **F**ix	A complete vendor solution is available
Report **C**onfidence	**N**ot **D**efined (ND)	Assigning this value to the metric will not influence the score. It is a signal to the equation to skip this metric

Environmental Metrics		
Collateral **D**amage **P**otential	**M**edium-**H**igh	A successful exploit of this vulnerability may result in significant loss of revenue or productivity
Target **D**istribution	**H**igh	Between 76% - 100% of the total environment is considered at risk

4 Vulnerability Details

Safari accepts PDF files as native image format for the < `image` > html tag. Thus browsing an html page in Safari can transparently load multiple pdf files without any further user interaction. CoreGraphics is the responsible of parsing the PDF files.

Apple CoreGraphics framework fails to validate the input when parsing the colorspace specification of a PDF XObject. A small heap memory allocation can be overflowed with controlled data from the input enabling arbitrary code execution in the context of MobileSafari (A memory layout information leak is needed, see CVE-2014-4378).

The Core Graphics framework is a C-based API that is based on the Quartz advanced drawing engine. It provides low-level, lightweight 2D rendering. This is used in a wide range of applications to handle path-based drawing, transformations, color management, offscreen rendering, patterns, gradients and shadings, image data management, image creation, masking, and PDF document creation, display, and parsing.

CoreGraphics library implements the functionality to load and save several graphic formats such as PDFs and it is used by most common applications that deal with images, including Safari, Preview, Skype, etc. The 32 bit version of the framework also implements the `x_alloc` heap, a set of low level procedures to manage an internal heap memory structure used when processing images of different types (including PDF files). The functions `x_calloc` and `x_free` are used very frequently to allocate and de-allocate memory fast. The `x_calloc` function behaves as a normal calloc except it allocates a bit extra memory for metadata. It actually allocates an extra `2*sizeof(void*)` bytes and pad the resulting size to the next 16 byte border. When the chunk is in use (allocated) this extra space is used to hold the size of the chunk, thus making it super fast to free. On the other hand when the chunk is free the metadata space is used to form a free-list linked list, the first metadata value in a free chunk points to the next free chunk. This is its pseudocode. Think `x_mem_alloc0_size` as a plain malloc.

```
void* __cdecl x_calloc(size_t nmemb, size_t size)
{
  void * buffer = x_mem_alloc0_size((nmemb * size + 31) & 0xfffffff0);
  *(size_t *)buffer = (nmemb * size + 31) & 0xfffffff0;
  return buffer + 16;
}
```

This function is prone to integer overflow because it doesn't check for `(nmemb * size)` to be less than `MAXUINT-16`. Then if a programmer tries to allocate a size in the range `[-16,-1]` `x_alloc` will alloc 0(zero) or 16 bytes (instead of the immense value required) without triggering any exception.

4.1 A PDF x_calloc bug.

There is a `x_calloc` related bug in how PDF handles the colorspace for embedded XObjects. Forms and images of different types can be embedded in a PDF file using an `XObject` pdf stream. As described in the pdf specification, an `XObject` can specify the colorspace in which the intended image is described. The CoreGraphics library fails to validate the input when parsing the `/Indexed` colorspace definition of an `XObject` pdf stream and enables an attacker to reach `x_calloc` with a size in the `[16,-1]` range.

The indexed colorspace definition is table based and relies on a base colorspace. For example, the following definition defines an indexed colorspace of 200 colors based on a RGB base colorspace with the table defined in the indirect object `8 0 R`. For more info on indexed colorspaces see section `8.6.6.3` of the pdf specification [2].

```
/ColorSpace [/Indexed /DeviceRGB 200 8 0 R]
```

The following is an excerpt of the function `_cg_build_colorspace`. It was taken from the `dyld_shared_cache_armv` of the iPhone3,1 (iPhone4) iOS version 7.1.1. The function can be disassembled at address `0x2D59F260` considering that the `dyld_shared_cache` is loaded at `0x2C000000`.

```
/* cs_index_array should represent something like this:
    [/Indexed /DeviceRGB 200 8 0 R]
*/

/* Sanity check, array must have 4 elements */
if ( CGPDFArrayGetCount(cs_index_array) != 4 ) {
    message = "invalid 'Indexed' color space: incorrect number of
       entries in color space array.";
    goto EXIT;
}

/* Sanity check, 2nd element should be an object */
if ( !CGPDFArrayGetObject(cs_index_array, 1, &base_cs_obj) ) {
```

```
    message = "invalid 'Indexed' color space: second color space array
       entry is not an object.";
    goto EXIT;
}

/* build the base colorspace */
base_cs = cg_build_colorspace(base_cs_obj);
if ( !base_cs ) {
    message = "invalid 'Indexed' color space: invalid base color space.";
    goto EXIT;
}

/* get the 3rd element. N, the number of indexed colors in the table */
if ( CGPDFArrayGetInteger(cs_index_array, 2, &N) ) {
    message = "invalid 'Indexed' color space: high value entry is not an
       integer.";
    goto RELEASE_EXIT;
}

/* Sanity check. N should be positive */
if ( N <= -1 ) {
    message = "invalid 'Indexed' color space: high value entry is
       negative.";
    goto RELEASE_EXIT;
}

/* cs is the resultant colorspace, init to NULL */
cs = 0;

/* if 4th element is a pdf stream get it and do stuff ...*/
if ( CGPDFArrayGetStream(cs_index_array, 3, &lookup_stream) == 1 ) {

    lookup_buffer = CGPDFStreamCopyData(lookup_stream);
    if ( lookup_buffer ) {

        /* N is a fully controled _positive_ integer and the number of
           components
           of the base colorspace is up to 32. Thus lookup_size is
              almost arbitrary.*/
        lookup_size = (N + 1) *
           CGColorSpaceGetNumberOfComponents(base_cs);

        /* Unsigned check. This will not hold with a lookup_size big
           enough (e.x. -10)*/
        if ( CFDataGetLength(lookup_buffer) >= lookup_size ) {
            data = CFDataGetBytePtr(lookup_buffer);
            cs = CGColorSpaceCreateIndexed(base_cs, N_, data);
        }
        else {
            /* HERE is the interesting bit. A lookup_size in the
               [-16,-1] range
               will silently allocate a very small buffer */
            overflow_buffer = x_calloc_2D5143B4(1, lookup_size);
```

```
            _data = CFDataGetBytePtr(lookup_buffer);
            _size = CFDataGetLength(lookup_buffer);
            /* But memove will copy all the available data in the stream
               OVERFLOW!!
             */
            memmove(overflow_buffer, _data, _size);

            /* CGColorSpaceCreateIndexed is a nop when N is greater than
               256 */
            cs = CGColorSpaceCreateIndexed(base_cs, N_, overflow_buffer);
            if ( overflow_buffer )
              x_free(overflow_buffer);
        }
        CFRelease(lookup_buffer);
    }

    goto RELEASEANDEXIT;
}

/* else if 4th element is a pdf string get it and do stuff ...*/
if ( CGPDFArrayGetString(cs_index_array, 3, &lookup_str) == 1 ) {
    lookup_size_ = (N + 1) * CGColorSpaceGetNumberOfComponents(base_cs);
    if ( lookup_size_ != CGPDFStringGetLength(lookup_str) ) {
        message = "invalid 'Indexed' color space: invalid lookup entry.";
        goto RELEASEANDEXIT;
    }
    buffer = CGPDFStringGetBytePtr(lookup_str);
    cs = CGColorSpaceCreateIndexed(base_cs, N__, buffer);
    goto RELEASEANDEXIT;

}

/* at this point cs is NULL */
message = "invalid 'Indexed' color space: lookup entry is not a stream
   or a string.";

RELEASE_EXIT:
    CGColorSpaceRelease(base_cs);
EXIT:
    log_2D5A0DA8(message);
/* result in cs */
```

`CGPDFArrayGetInteger` gets an object from an arbitrary position in a pdf array, starting at 0.

The number of indexed colors is read from the third array position (index 2), multiplied by the number of color components of the base colorspace and then (if using pdf stream) allocated with a `x_calloc`. An attacker could control to some degree the size passed to the `x_calloc` function. If the resultant value is for instance -10, `x_calloc` will allocate a small amount of memory and return a valid pointer. If the innner colorspace is `/DeviceRGB` with 3 color components, passing a value of 0x55555555 will do the trick.

Then the `memmove` will potentially overflow the small buffer with an arbitrary number of arbitrary bytes. The function `CGColorSpaceCreateIndexed` can be considered as a `no-op` as it will use this buffer only if our controlled size is a positive less than `0xff`, not the interesting case.

5 Exploitation

A 100% reliable PoC exploit can be downloaded from the github project [3]. This exploit needs a companion information leakage vulnerability to bypass(see [4]) ASLR, DEP and Code signing iOS exploit mitigations. The exploit is presented as a cgi script that expects to get the `dyld_shared_cache address`, the shellcode address and the iOS version as GET parameters. It executes arbitrary code in the context of SafariMobile.

6 References

1 `http://en.wikipedia.org/wiki/Quartz_(graphics_layer)`

2 `http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/PDF32000_2008.pdf`

3 `https://github.com/feliam/CVE-2014-4377`

4 `https://github.com/feliam/CVE-2014-4378`