DOYENSEC

Security Advisory
Font parsing vulnerabilities in macOS, iOS, tvOS, watchOS

Created by John Villamil
04/11/2017

Overview

This document summarizes the results of a vulnerability research activity aimed at discovering font parsing vulnerabilities in Apple's macOS. While security testing was not meant to be comprehensive in term of attack and code coverage, we have identified four (4) vulnerabilities that could lead to code execution and information leakage through parsing of malicious font files.

On Mar 27th 2017, Apple has released an update to address these issues affecting macOS, iOS, tvOS and watchOS.

About Us

Doyensec is an independent security research and development company focused on vulnerability discovery and remediation. We work at the intersection of software development and offensive engineering to help companies craft secure code.

Research is one of our founding principles and we invest heavily in it. By discovering new vulnerabilities and attack techniques, we constantly improve our capabilities and contribute to secure the applications we all use.

macOS, iOS, tvOS, watchOS CarbonCore Buffer Overflow

Vendor	Apple
Severity	High
Vulnerability Class	Memory Corruption
Component	CarbonCore
Status	Patched
CVE	CVE-2017-2379
Credits	John Villamil @day6reak

Summary

A memory corruption vulnerability was identified in a core component of Apple's font parsing - CarbonCore. This issue could allow an attacker to execute code during the parsing of a malicious Datafork TrueType font.

Technical Description

When parsing the *dfont* file format, CarbonCore reads a DWORD from the file and uses it to index a memory address without any validation. The "size" argument of a call to bcopy is read from this attacker controlled index.

In the following instruction, *rax* is attacker controlled.

```
0x7fff92c48824 <+418>: movzx  edx, byte ptr [rcx + rax]

    frame #0: 0x00007fff92c48824 CarbonCore`GetResourcePtrCommon + 418
    frame #1: 0x00007fff92c4b7bc CarbonCore`RMGetIndexedResource + 42
    frame #2: 0x00007fff8f00599e
libFontParser.dylib`TResourceForkFileReference::GetIndexedResource(unsigned int, unsigned int,
short*, unsigned long*, unsigned char*) const + 54
    frame #3: 0x00007fff8f005927
libFontParser.dylib`TResourceFileDataReference::TResourceFileDataReference(TResourceForkSurrog
ate
const&, unsigned int, unsigned int) + 157
    frame #4: 0x00007fff8f00584e
libFontParser.dylib`TResourceFileDataSurrogate::TResourceFileDataSurrogate(TResourceForkSurroga
te
const&, unsigned int, unsigned int) + 66
    frame #5: 0x00007fff8f05006c
libFontParser.dylib`TFont::CreateFontEntities(char const*, bool, TSimpleArray<TFont*>&, short, char
const*, bool) + 890
    frame #6: 0x00007fff8f0011a6
libFontParser.dylib`TFont::CreateFontEntitiesForFile(char const*, bool, TSimpleArray<TFont*>&, bool,
short, char const*) + 176
    frame #7: 0x00007fff8f000b72
libFontParser.dylib`FPFontCreateFontsWithPath + 209
    frame #8: 0x00000001074e7ba9
libCGXType.A.dylib`create_private_data_with_path + 19
    frame #9: 0x00007fff93576620 CoreGraphics`CGFontCreateFontsWithPath + 56
```

Exploitation of this vulnerability allows an attacker to execute code on the victim's machine through parsing of a malicious file.

Proof-of-Concept has not been included in this report.

Remediation

Apple has released an update to address this issue:

- https://support.apple.com/en-us/HT207615 (macOS)
- https://support.apple.com/en-us/HT207617 (iOS)
- https://support.apple.com/en-us/HT207602 (watchOS)
- https://support.apple.com/en-us/HT207601 (tvOS)

Disclosure Timeline

12/22/2016	Vulnerability disclosed to Apple via product-security@apple.com
03/27/2017	Advisory and patches released by Apple

macOS, iOS, tvOS, watchOS CoreText Corrupted Loop Index

Vendor	Apple
Severity	High
Vulnerability Class	Memory Corruption
Component	CoreText
Status	Patched
CVE	CVE-2017-2435
Credits	John Villamil @day6reak

Summary

A memory corruption vulnerability was identified in a core component of Apple's font parsing - CoreText. Through a malicious True Type Collection (ttc) font file, CoreText will enter a loop unintentionally referencing out of bounds memory.

Technical Description

The following is a stack trace recorded at the time of crash. The flaw happens during glyph processing.

```
CoreText`TRunGlue::GetAdvance(long) + 71, queue = 'com.apple.main-thread',
stop reason = EXC_BAD_ACCESS (code=1, address=0x1066d8000)
  * frame #0: 0x00007fff90246a9d CoreText`TRunGlue::GetAdvance(long) + 71
    frame #1: 0x00007fff902a025c
CoreText`TAATKerxEngine::MatchCoordinates(TRunGlue::TGlyph, TRunGlue::TGlyph, int, short, short) +
216
    frame #2: 0x00007fff9029fee0
CoreText`TAATKerxEngine::KerxControlPointTable::ProcessGlyphs(SyncState&) + 1154
    frame #3: 0x00007fff9029f416
CoreText`TAATKerxEngine::ProcessKerxControlPointTable(KerxControlPointHeader const*, unsigned
int, SyncState&) + 82
    frame #4: 0x00007fff9029f0c6
CoreText`TAATKerxEngine::KernRuns(SyncState&, KerningStatus&) + 602
    frame #5: 0x00007fff90241fed
CoreText`TKerningEngine::PositionGlyphs(TLine&, TCharStream const*) + 497
```

Exploitation of this vulnerability allows an attacker to execute code on the victim's machine through parsing of a malicious file.

Proof-of-Concept has not been included in this report.

Remediation

Apple has released an update to address this issue:

- https://support.apple.com/en-us/HT207615 (macOS)
- https://support.apple.com/en-us/HT207617 (iOS)
- https://support.apple.com/en-us/HT207602 (watchOS)
- https://support.apple.com/en-us/HT207601 (tvOS)

Disclosure Timeline

12/16/2016	Vulnerability disclosed to Apple via product-security@apple.com
03/27/2017	Advisory and patches released by Apple

macOS, iOS, tvOS, watchOS FontParser Infoleak

Vendor	Apple
Severity	Medium
Vulnerability Class	Information Disclosure
Component	FontParser
Status	Patched
CVE	CVE-2017-2439
Credits	John Villamil @day6reak

Summary

An information leakage vulnerability (out-of-bounds read) was discovered in Apple's FontParser, which could allow an attacker to disclose the process memory. This issue could facilitate further exploitation.

Technical Description

A loop iteration can be controlled, causing it to read into unmapped memory.

The loop below calls *FindIndexedString*. This function will return a pointer to a 0. That will be the first byte of a hard coded style table. While *esi* is 0 this table won't be parsed past the first byte. The registers *rdx* and *r12* are attacker controlled.

```
#TFONDData::GetPostscriptName(short, unsigned char*, unsigned long)
00000000000070a2      mov      r15, rcx        ; CODE
XREF=__ZNK9TFONDData17GetPostscriptNameEsPhm+266
00000000000070a5      movzx    esi, byte [r15]                 ;CRASH
00000000000070a9      mov      rdi, qword [rbp+var_40]
00000000000070ad      call     FindIndexedString(FontNameTable_BE const&, unsigned long)
00000000000070b2      mov      rcx, rax
00000000000070b5      movzx    edx, byte [rcx]
00000000000070b8      lea      r13, qword [rdx+r12]
00000000000070bc      cmp      r13, qword [rbp+var_30]        ;var_30 is 0xff
00000000000070c0      mov      eax, 0x0
00000000000070c5      jae      loc_70f3
```

```
* frame #0: 0x00007fff8c6110a5 libFontParser.dylib`TFONDData::GetPostscriptName(short, unsigned
char*, unsigned long) const + 195
frame #1: 0x00007fff8c610ef3 libFontParser.dylib`TFONDData::GetPostscriptName(short) const + 69
frame #2: 0x00007fff8c610de2 libFontParser.dylib`TTrueTypeResourceFont::GetPostscriptName()
const + 64
frame #3: 0x00007fff8c60d4fa
libFontParser.dylib`TArrayOfFontsWithUniquePostscriptNames::Append(TFont* const&) + 48
frame #4: 0x00007fff8c65b42f libFontParser.dylib`TFont::CreateFontEntities(char const*, bool,
TSimpleArray<TFont*>&, short, char const*, bool) + 1853
```

Proof-of-Concept has not been included in this report.

Remediation

Apple has released an update to address this issue:

- https://support.apple.com/en-us/HT207615 (macOS)
- https://support.apple.com/en-us/HT207617 (iOS)
- https://support.apple.com/en-us/HT207602 (watchOS)
- https://support.apple.com/en-us/HT207601 (tvOS)

Disclosure Timeline

12/25/2016	Vulnerability disclosed to Apple via product-security@apple.com
03/27/2017	Advisory and patches released by Apple

macOS, iOS, tvOS, watchOS CoreText Infoleak

Vendor	Apple
Severity	Medium
Vulnerability Class	Information Disclosure
Component	CoreText
Status	Patched
CVE	CVE-2017-2450
Credits	John Villamil @day6reak

Summary

An information leakage vulnerability (out-of-bounds read) was discovered in Apple's CoreText, which could allow an attacker to disclose the process memory. This issue could facilitate further exploitation.

Technical Description

A value is read from a True Type Collection font file without any verification being performed. This value is added as an offset to an address. When this address is dereferenced, a crash occurs.

We see *r15* being set:

```
00000000000fd986    mov     r15d, dword [r12+rax*4]
00000000000fd98a    bswap   r15d
00000000000fd98d    mov     r14d, dword [r12+rax*4+4]
00000000000fd992    bswap   r14d
00000000000fd995    jmp     loc_fda2e
```

A DWORD is read from the font file and a bit swap is performed. The unsanitized *r15* register isn't used for a little while until it loads *rbx* with an address. Since *r15* isn't verified this address can point to almost anywhere:

```
00000000000fdad0    mov     r8, qword [rbp+var_88]
00000000000fdad7    lea     rbx, qword [r15+r8+0xa]
00000000000fdadc    cmp     rbx, r13
```

```
0000000000fdadf        ja        loc_fdb5b
```

And the access violation happens a few instructions later when it tries to read a word from the unchecked address which is unmapped in this case:

```
CoreText`TAATControlPointAccess::GetControlPointCoordinates:
->  0x7fff95d44b0f <+719>: mov    si, word ptr [rbx]
 * frame #0: 0x00007fff95d44b0f
CoreText`TAATControlPointAccess::GetControlPointCoordinates(unsigned short,unsigned short) const
+ 719
    frame #1: 0x00007fff95cc7d7b
CoreText`TAATKerxEngine::KerxControlPointTable::ProcessGlyphs(SyncState&) +797
    frame #2: 0x00007fff95cc7416
CoreText`TAATKerxEngine::ProcessKerxControlPointTable(KerxControlPointHeader const*, unsigned
int, SyncState&) + 82
    frame #3: 0x00007fff95cc70c6
CoreText`TAATKerxEngine::KernRuns(SyncState&, KerningStatus&) + 602
    frame #4: 0x00007fff95c69fed
CoreText`TKerningEngine::PositionGlyphs(TLine&, TCharStream const*) + 497
```

Proof-of-Concept has not been included in this report.

Remediation

Apple has released an update to address this issue:

- https://support.apple.com/en-us/HT207615 (macOS)
- https://support.apple.com/en-us/HT207617 (iOS)
- https://support.apple.com/en-us/HT207602 (watchOS)
- https://support.apple.com/en-us/HT207601 (tvOS)

Disclosure Timeline

01/10/2017	Vulnerability disclosed to Apple via product-security@apple.com
03/27/2017	Advisory and patches released by Apple