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.

Copyright 2017. Doyensec LLC. All rights reserved.

Permission is hereby granted for the redistribution of this advisory, provided that it is not altered except by reformatting it, and that due credit is given. Permission is explicitly given for insertion in vulnerability databases and similar, provided that due credit is given. The information in the advisory is believed to be accurate at the time of publishing based on currently available information, and it is provided as-is, as a free service to the community by Doyensec LLC. There are no warranties with regard to this information, and Doyensec LLC does not accept any liability for any direct, indirect, or consequential loss or damage arising from use of, or reliance on, this information.
macOS, iOS, tvOS, watchOS CarbonCore Buffer Overflow

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Apple</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Vulnerability Class</td>
<td>Memory Corruption</td>
</tr>
<tr>
<td>Component</td>
<td>CarbonCore</td>
</tr>
<tr>
<td>Status</td>
<td>Patched</td>
</tr>
<tr>
<td>CVE</td>
<td>CVE-2017-2379</td>
</tr>
<tr>
<td>Credits</td>
<td>John Villamil @day6reak</td>
</tr>
</tbody>
</table>

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.

```
0x7ff92c48824 <+418>: movzx edx, byte ptr [rcx + rax]
```

```
frame #0: 0x00007ff92c48824 CarbonCore`GetResourcePtrCommon + 418
frame #1: 0x00007ff92c4b7bc CarbonCore`RMGetIndexedResource + 42
frame #2: 0x00007ff8f00599e
libFontParser.dylib`TResourceForkFileReference::GetIndexedResource(unsigned int, unsigned int, short*, unsigned long*, unsigned char*) const + 54
frame #3: 0x00007ff8f005927
libFontParser.dylib`TResourceFileDataReference::TResourceFileDataReference(TResourceForkSurrogate const&, unsigned int, unsigned int) + 157
frame #4: 0x00007ff8f00584e
libFontParser.dylib`TResourceFileDataSurrogate::TResourceFileDataSurrogate(TResourceForkSurrogate
```
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:


Disclosure Timeline

12/22/2016  Vulnerability disclosed to Apple via product-security@apple.com
03/27/2017  Advisory and patches released by Apple
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: 0x00007ff90246a9d CoreText`TRunGlue::GetAdvance(long) + 71
  frame #1: 0x00007ff902a025c
CoreText`TAATKerxEngine::MatchCoordinates(TRunGlue::TGlyph, TRunGlue::TGlyph, int, short, short) + 216
  frame #2: 0x00007ff902gfee0
CoreText`TAATKerxEngine::KerxControlPointTable::ProcessGlyphs(SyncState&) + 1154
  frame #3: 0x00007ff902f416
CoreText`TAATKerxEngine::ProcessKerxControlPointTable(KerxControlPointHeader const*, unsigned int, SyncState&) + 82
  frame #4: 0x00007ff902f0c6
CoreText`TAATKerxEngine::KernRuns(SyncState&, KerningStatus&) + 602
  frame #5: 0x00007ff90241fed
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:


Disclosure Timeline

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>12/16/2016</td>
<td>Vulnerability disclosed to Apple via <a href="mailto:product-security@apple.com">product-security@apple.com</a></td>
</tr>
<tr>
<td>03/27/2017</td>
<td>Advisory and patches released by Apple</td>
</tr>
</tbody>
</table>
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, ox0
00000000000070c5 jae loc_70f3
```
Proof-of-Concept has not been included in this report.

Remediation

Apple has released an update to address this issue:


Disclosure Timeline

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

```
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: 0x00007fff95c69fed
CoreText TK kerningEngine::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:


Disclosure Timeline

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/10/2017</td>
<td>Vulnerability disclosed to Apple via <a href="mailto:product-security@apple.com">product-security@apple.com</a></td>
</tr>
<tr>
<td>03/27/2017</td>
<td>Advisory and patches released by Apple</td>
</tr>
</tbody>
</table>