PoDoFo Tickets

A PDF parsing, modification and creation library.
Brought to you by: domseichter

#49 memory leakage at src/base/PdfTokenizer.cpp:334 in IsNextToken(const char*)

Milestone: SVN TRUNK **Status:** closed **Owner:** Matthew Brincke **Labels:** None
Updated: 2021-08-18 **Created:** 2019-04-04 **Creator:** Tao **Private:** No

Hi, there is a memory leakage bug at `src/base/PdfTokenizer.cpp:334 IsNextToken`

```
bool PdfTokenizer::IsNextToken( const char* pszToken )
{
    if( !pszToken )
    {
        PODOFO_RAISE_ERROR( ePdfError_InvalidHandle );
    }

    const char* pszRead;
    bool gotToken = this->GetNextToken( pszRead, NULL );

    if (!gotToken)
    {
        PODOFO_RAISE_ERROR( ePdfError_UnexpectedEOF );
    }

    return (strcmp( pszToken, pszRead ) == 0);
}
```

The backtrace is shown as following. But before the process exited, it didn't `free` memory `new`'d at `PoDoFo::PdfMemDocument::Load`.

```
void PdfMemDocument::Load( const char* pszFilename, bool bForUpdate )
{
    if( !pszFilename || !pszFilename[0] )
    {
        PODOFO_RAISE_ERROR( ePdfError_InvalidHandle );
    }

    this->Clear();

    if( bForUpdate )
    {
        int lLen = strlen( pszFilename );
        m_pszUpdatingFilename = static_cast<char *>( podofo_malloc( sizeof( char ) * ( lLen
        memcpy( m_pszUpdatingFilename, pszFilename, lLen );
        m_pszUpdatingFilename[lLen] = '\0';
    }

    // Call parse file instead of using the constructor
    // so that m_pParser is initialized for encrypted documents
    **m_pParser = new PdfParser( PdfDocument::GetObjects() );**
    m_pParser->ParseFile( pszFilename, true );
    InitFromParser( m_pParser );
}
```

```
`cmd : podofopdfinfo poc`
#0  PoDoFo::PdfTokenizer::IsNextToken (this=0x617000000080, pszToken=0x8ad0c0 <.str.23> "tr
#1  0x000000000075d53a in PoDoFo::PdfParser::ReadNextTrailer (this=0x617000000080) at /home
#2  0x000000000075cfb4 in PoDoFo::PdfParser::ReadXRefContents (this=0x617000000080, lOffset
#3  0x0000000000758dd7 in PoDoFo::PdfParser::ReadDocumentStructure (this=0x617000000080) at
#4  0x000000000075772e in PoDoFo::PdfParser::ParseFile (this=0x617000000080, rDevice=..., b
#5  0x0000000000756734 in PoDoFo::PdfParser::ParseFile (this=0x617000000080, pszFilename=0x
#6  0x00000000006bc080 in PoDoFo::PdfMemDocument::Load (this=0x614000000040, pszFilename=0x
#7  0x00000000006bbb90 in PoDoFo::PdfMemDocument::PdfMemDocument (this=0x614000000040, pszF
Python Exception <class 'gdb.error'> There is no member named _M_dataplus.:
#8  0x00000000005c1972 in PdfInfo::PdfInfo (this=0x7fffffffe150, inPathname=) at /home/lt/v
#9  0x00000000005cc279 in main (argc=0x2, argv=0x7fffffffe528) at /home/lt/vuln-fuzz/progra
```

The result shown by valgrind is as following:

```
==114144==
==114144== HEAP SUMMARY:
==114144==     in use at exit: 78,160 bytes in 6 blocks
==114144==   total heap usage: 124 allocs, 118 frees, 96,166 bytes allocated
==114144==
==114144== 5,456 (744 direct, 4,712 indirect) bytes in 1 blocks are definitely lost in loss
==114144==    at 0x4C2E0EF: operator new(unsigned long) (in /usr/lib/valgrind/vgpreload_memo
==114144==    by 0x5B9295: Load (PdfMemDocument.cpp:255)
==114144==    by 0x5B9295: PoDoFo::PdfMemDocument::PdfMemDocument(char const*, bool) (PdfMem
==114144==    by 0x43E8A2: PdfInfo::PdfInfo(std::__cxx11::basic_string<char, std::char_trai
==114144==    by 0x436E4E: main (podofopdfinfo.cpp:110)
==114144==
==114144== LEAK SUMMARY:
==114144==    definitely lost: 744 bytes in 1 blocks
==114144==    indirectly lost: 4,712 bytes in 4 blocks
==114144==      possibly lost: 0 bytes in 0 blocks
==114144==    still reachable: 72,704 bytes in 1 blocks
==114144==         suppressed: 0 bytes in 0 blocks
==114144== Reachable blocks (those to which a pointer was found) are not shown.
==114144== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==114144==
==114144== For counts of detected and suppressed errors, rerun with: -v
==114144== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

1 Attachments

mem-leak

Discussion

Matthew Brincke - *2019-05-08*

- **summary**: memory leakage at --> memory leakage at src/base/PdfTokenizer.cpp:334 in IsNextToken(const char*)
- Description has changed:

Diff:

```
--- old
+++ new
@@ -1,4 +1,4 @@
-Hi, there is a memory leakage bug at  `src/base/PdfToenizer.cpp:334 IsNextToken`
+Hi, there is a memory leakage bug at  `src/base/PdfTokenizer.cpp:334 IsNextToken`
 ~~~
 bool PdfTokenizer::IsNextToken( const char* pszToken )
 {
@@ -18,7 +18,7 @@
     return (strcmp( pszToken, pszRead ) == 0);
 }
 ~~~
-The backtrace is shown following. But why the process exit, it didn't `free` memory `new`  at `PoDoFo::
+The backtrace is shown as following. But why the process exited, it didn't `free` memory `new`'d  at `P
 ~~~
 void PdfMemDocument::Load( const char* pszFilename, bool bForUpdate )
 {
```

Matthew Brincke - *2019-05-31*

My proposed fix is attached here, I fully intend to test it today (UTC), haven'T yet done so.

fix-issue49.diff

Matthew Brincke - *2019-05-31*

That patch doesn't work, I've got a better one tested OK (with ASan, not valgrind) on a newer system with GCC 7.4 and with clang 7.0, attached here (yet to test with GCC 4.8 & clang 3.8).

fix-issue49-try2.patch

Matthew Brincke - *2019-06-01*

- Description has changed:

Diff:

```
--- old
    ...18,7 +1...
     return ( strcmp( pszToken, pszRead ) == 0);
 }
 ~~~
-The backtra...
+The backtrace is shown as following. But before the process exited, it didn't `free` memory `new`ed ...
 ~~~
 void PdfMemDocument::Load( const char* pszFilename, bool bForUpdate )
 {
```

Log in to post a comment.

Matthew Brincke - 2019-06-01

- **status**: pending --> closed

[illegible]

for the test case. [illegible] usage share.

Considering how old this ticket is and the fact the code is changed, I'm closing this, but please reopen with the test case, if you can. Thanks in advance.

Last edit: Matthew Brincke 2019-06-01

- **status**: open --> pending
- **assigned_to**: Matthew Brincke

Matthew Brincke - *2019-06-02*

The Windows-only wchar_t overload of PdfMemDocument::Load isn't changed because I (typically at least, I'm not sure if I absolutely adhered to that in the past) don't commit changes I can't test (I don't use Windows). If you use Windows, please make the change also for that overload and test it, then report back (after that, if you're a committer and all tests went OK, I'd very much like it for that to include clang builds, please commit).