REPORT ON OPEN SOURCE PRODUCT (APACHE OPEN OFFICE)

Technical Report

Gunarathna A.P.S.D – IT17136334

Kabilashan P. - IT17146234

B.Sc. (Hon) Degree in Information Technology

Specializing in Cyber Security

Department of Information System Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

May 2020

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Description
UNO	Unlimited Networks of Opportunities
OO	Open Office
OOo	Open Office.org

1. INTRODUCTION

According to Secure Software Engineering module this documentation extracts the research work that has completed on open source product. Apache open office is the open source product that has selected. Under this product assessment this document contains about Domain and historical analysis of the product, Design analysis of the product, and Code inspection for vulnerability assessment as main chapters.

Open source software is a software with source code that anyone can inspect, modify and enhance according to any perspective. Open-source software may be implemented in a collaborative public manner. According to that any developer can engage with the software and do add their own created features and can report and fix if there any kind of bugs and weakness available.

Some software has source code that only one person or else one team can access the source code who has created it and they have every control over the software product. This kind of software are known as closed source software. Only the original authors of proprietary software can legally copy or do any modification in the software. In case if any other party that is trying to inspect the code of closed software will be considered as the unauthorized activity and even could be penalized. Adobe Photoshop and Microsoft office are example for closed source software. Open source software in not only important to computer programmers because open source technology and open thinking both beneficial to programmers and non-programmers.

2. DOMAIN AND HISTORICAL ANALYSIS

2.1. Product overview

2.1.1. What is open office?

OpenOffice, sometimes abbreviated as OO, is a free and open-source office productivity software suite offered by The Apache Software Foundation (ASF) for word processing, spreadsheets, presentations, databases, graphics, and more.

The office productivity software suite is available in many languages and is compatible with all major operating systems, including Apple macOS, Microsoft Windows, and Linux. It includes four main applications: Writer, Calc, Impress, and Base which are competitors to Microsoft Word, Excel, PowerPoint, and Access, respectively. Refer Figure 2.1 to see how Apache's open office user interface looks. Because OO is free, it can save you money that you would need to buy costly office productivity suites in the market. This cost can be an unwanted expense if your business is on a tight budget.

2.1.2. Why should I use Apache open office?

Apache OpenOffice is available in many languages and works on all common computers. It stores all your data in an international open standard format and can also read and write files from other common office software packages. It can be downloaded and used completely free of charge for any purpose.

Apache OpenOffice is the result of over twenty years' software engineering. Designed from the start as a single piece of software, it has a consistency other product cannot match. A completely open development process means that anyone can report bugs, request new features, or enhance the software.

2.1.3. How Apache open office works?

OO offers a high degree of compatibility with none of the costs or license worries. It has a friendly user interface and feature set like those of 'commercial' office suites. These premium features have in recent years encouraged many organizations across the board to try the program.

Among the organizations and industries that Apache lists as having adopted OO are:

- Governments
- Education
- Businesses
- Not for profits
- IT Businesses
- F/OSS advocates

Small businesses from corner-shops to grocery stores and restaurant chains love Apache OpenOffice because it is a free and secure product that guarantees trouble-free usage. Behind the scenes, OO stores all your valuable data in a format approved by the International Organization for Standardization (ISO). It also allows you to exchange data between office software, accounting software, planning software any software as easily as opening and saving a file.

Figure 2. 1: Apache Open Office Features

2.1.4. Reasons to use Apache open office

- **Price** - Apache OpenOffice is free. Small businesses looking for professional-caliber office productivity suite of programs for free may find OpenOffice a good alternative to other options.
- **Easy to learn** - If you are already using another office software package, you will take to OpenOffice straight away, says Apache. And if you already have files from another office package OpenOffice will likely read them with no difficulty.
- **Software can use for any purpose** - commercial, domestic, educational, public administration. You may also install OO on as many computers as you like and make copies and give them away to family, friends, employees, anyone.
- **allows for the use of "extensions" and document "templates"** - An extension is a third-party tool that brings OpenOffice new functions, explains ASF. This can be done through addons, add-ins deployed by Unlimited Networks of Opportunities (UNO) packages. Templates, on the other hand, are document setups designed for specific uses.

2.2. Product Assets

2.2.1. Apache open office base

Base is a fully featured desktop database management system, designed to meet the needs of a broad array of users, from

- Just tracking your personal CD collection, to
- Producing a corporate monthly departmental sales report.

Base offers wizards to help users new to database design (or just new to Base) to create Tables, Queries, Forms and Reports, along with a set of predefined table definitions for tracking Assets, Customers, Sales Orders, Invoices and much more.

When a personal use database is all you need, Base offers the full HSQL relational database engine, configured for single user, with the data stored right in the Base file, as well as native support for dBase flat files.

For power users in the enterprise, Base delivers native support drivers for a variety of multi-user database engines: MySQL, MS Access and PostgreSQL. In addition, support for JDBC and ODBC standard drivers allows you to connect to virtually any existing database.

Base integrates seamlessly into the rest of the Apache OpenOffice suite applications, for example:

- Supplying address book data for mail merge in WRITER using the industry standard LDAP protocol, or common address book formats such as Microsoft Outlook, Microsoft Windows, and Mozilla.
- Creating linked data ranges in CALC files for data pilot analysis or as the basis for charts.

2.2.2. Apache OpenOffice Wiki Database
Table 2 1: Modules in the Database Access Project

Module	Functionality
connectivity	Base connectivity. This module contains database driver implementations for ODBC 3.0, JDBC, ADO, dBase, and CSV files. The way to access these drivers is very similar to JDBC as the drivers implement an API which covers the JDBC API.
db access	Database access layer. This module contains code for accessing databases from applications. It also contains the following: • Core implementations for configuration of data sources • Additional implementations like Row Sets and abstractions like database meta information • A graphical user interface (GUI) for customizing data sources and access to data

Table 2 2: Modules in other Projects, related to Database Access

Module	Functionality
forms	This module, though currently part of the GSL project, is related to Database access as well. It contains most of the code necessary to build up a form layer (means the components for logical forms, control models, and controls). This project is self-contained, means it is only exporting UNO components.
svx	Additional implementations for integrating the form layer into the applications can be found in the SVX module, project graphics. Here, the directories svx/source/form and svx/source/fmcomp belong to Database Access.

2.2.3. Application Programming Interfaces

The OpenOffice.org API is based on the OpenOffice.org component technology and consists of a wide range of interfaces defined in a CORBA-like IDL. While the component technology determines how the components or applications communicate with each other, the OpenOffice.org API defines the interface for accessing office functionality from different programming languages. This interface structure is very important in determining the degree to which re-application of a development is possible.

The interfaces defined by the OpenOffice.org API are characterized as follows:

- They are completely defined component **interfaces** with the environment.
- They are **version independent** and **scalable**.
- They are **durable**.
- They are **re-applicable**.

Unlike other office suite APIs, the OpenOffice.org API does not simply reflect the features of preexisting implementations. It offers programming interfaces for nearly all OpenOffice.org components and makes it possible to integrate new components.

2.2.4. Application Areas

There are multiple ways to use OpenOffice.org APIs. First, there is the typical macro programming for running certain tasks automatically. Secondly, parts of OpenOffice.org can be run as components of other programs. For Example, OpenOffice.org components are accessible as JavaBeans components.

A more advanced application is to modify OpenOffice.org components by wrapping them into replacement components or integrating completely new components with OpenOffice.org. A very interesting application area is to replace the user interface of OpenOffice.org and build a completely different application domain.

3. VULNERABILITY HISTORY AND SAMPLE ATTACKS

3.1. Multiple Vulnerabilities in Apache OpenOffice Could Allow for Arbitrary Code Execution
Overview

Multiple vulnerabilities have been discovered in OpenOffice, which could allow for arbitrary code execution. OpenOffice is an open-source productivity software suite that contains a word processor, spreadsheet application, presentation application, drawing application, formula editor, and a database management application. Successfully exploiting these vulnerabilities could allow for arbitrary code execution in the context of the affected application. Depending on the privileges associated with the application, an attacker could install programs; view, change, or delete data; or create new accounts with full user rights. Failed exploitation could result in a denial-of-service condition.

Threat intelligence

There are currently no reports of these vulnerabilities being exploited in the wild.

Systems affected

Apache OpenOffice prior to 4.1.4

3.2. Vulnerabilities that allow for arbitrary code execution
3.2.1. CVE-2017-9806
A vulnerability in the OpenOffice Writer DOC file parser, specifically in the WW8Fonts Constructor, allows attackers to craft malicious documents that cause denial of service (memory corruption and application crash) potentially resulting in arbitrary code execution. This vulnerability can be causes to availability of the system.

Table 3. 1:CVSS Scores & Vulnerability Types for CVE-2017-9806

CVSS Score	6.8
Confidentiality Impact	Partial (There is considerable informational disclosure.)
Integrity Impact	Partial (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 (There is reduced performance or interruptions in resource availability.)
Access Complexity	Medium (The access conditions are somewhat specialized. Some preconditions must be satisfied to exploit)
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	Denial Of Service Execute Code Memory corruption
CWE ID	787

This vulnerability is present in Apache OpenOffice (formerly OpenOffice.org), a free open source office suite. A specially crafted DOC file can lead to an out of bound write and ultimately to remote code execution.

Let us investigate this vulnerability. After Open Office Writer opens the malformed doc file, we see the following state.

Figure 3. 1: I - Source code that investigate

Figure 3. 2: II - Source code that investigate

So we see that write access violation appeared in the [**WW8Fonts: :WW8**] Fonts constructor. The definition of this function is in file
[**/storage/aoo-4.1.3/main/sw/source/filter/ww8/ww8scan.cxx:6571**]

Figure 3. 3: Checking the pTmp pointer

When an attempt to write to the address pointed to by the pTmp pointer, we encounter an access violation will because it points to mapped "libunoxml.so" library.

```
1    WW8Fonts::WW8Fonts( SvStream& rSt, WW8Fib& rFib )
2        : pFontA(0), nMax(0)
3    {
4    //  sal_uInt16 nMax;
5    (...)
6        rSt.Seek( rFib.fcSttbfffn );
7
8        sal_Int32 nFFn = rFib.lcbSttbfffn - 2;
9
10       // allocate Font Array
11       sal_uInt8* pA   = new sal_uInt8[ nFFn ];
12       memset(pA, 0, nFFn);
13       WW8_FFN* p = (WW8_FFN*)pA;
14
15       ww::WordVersion eVersion = rFib.GetFIBVersion();
16
17       if( eVersion >= ww::eWW8 )
18       {
19           // bVer8: read the count of strings in nMax
20           rSt >> nMax;
21       }
22
23   (...)
24           WW8_FFN_Ver8* pVer8 = (WW8_FFN_Ver8*)pA;
25           sal_uInt8 c2;
26           for(sal_uInt16 i=0; i<nMax; ++i, ++p)
27           {
28
29               (...)
30                   sal_uInt8 nLen = 0x28;
31                   sal_uInt8 nLength = sizeof( pVer8->szFfn ) / sizeof( SVBT16 );
32                   nLength = std::min( nLength, sal_uInt8( pVer8->cbFfnM1+1 ) );
33                   for( sal_uInt16* pTmp = pVer8->szFfn;
34                       nLen < nLength; ++pTmp, nLen+=2 )
35                   {
36                       *pTmp = SVBT16ToShort( *(SVBT16*)pTmp );
37                   }
38               (...)
39               // Zeiger auf Ursprungsarray einen Font nach hinten setzen
40               pVer8 = (WW8_FFN_Ver8*)( ((sal_uInt8*)pVer8) + pVer8->cbFfnM1 + 1 );
41           }
```

Figure 3. 4: Source code of libunoxml.so library

The loop at line 26 is based on nMax value. Each time at the end of this loop (at line 40) pVer8 pointer is set to new location based on the cbFfnM1 field value. pVer8 is a pointer to a dynamically allocated buffer, which is allocated at line 11, with a size equal to rFib.lcbSttbfffn - 2. As we can see there is no check to see whether after first iteration pVer8 is pointing outside buffer range or not. That situation leads to out of bound read/writes in certain places and finally can lead to remote code execution.

```
gdb-peda$ p rFib.lcbSttbfffn
$30 = 0xb6
gdb-peda$ p rFib.fcSttbfffn
$31 = 0x501
gdb-peda$ p nMax
$32 = 0xffff
gdb-peda$ p eVersion
$33 = ww::eWW8
gdb-peda$ p *(WW8_FFN_Ver8*)pA
$38 = {
  <WW8_FFN_BASE> = {
    cbFfnM1 = 0x0,
    prg = 0x0,
    fTrueType = 0x0,
    ff = 0x0,
    wWeight = 0xffff,
    chs = 0x0,
    ibszAlt = 0x0
```

Figure 3. 5: A dump of some important fields

Table 3. 2: Products Affected By CVE-2017-9806

	Product Type	Vendor	Product	Version
1	Application	Apache	Openoffice	4.0.0
2	Application	Apache	Openoffice	4.0.1
3	Application	Apache	Openoffice	4.1.0
4	Application	Apache	Openoffice	4.1.1
5	Application	Apache	Openoffice	4.1.2
6	Application	Apache	Openoffice.org	-

3.2.2. CVE-2017-12607

A vulnerability in OpenOffice's PPT file parser, specifically in PPT Style Sheet, allows attackers to craft malicious documents that cause denial of service (memory corruption and application crash) potentially resulting in arbitrary code execution. **This vulnerability can be causes to availability of the system.**

Table 3. 3: CVSS Scores & Vulnerability Types for CVE-2017-12607

CVSS Score	6.8
Confidentiality Impact	Partial (There is considerable informational disclosure.)
Integrity Impact	Partial (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 (There is reduced performance or interruptions in resource availability.)
Access Complexity	Medium (The access conditions are somewhat specialized. Some preconditions must be satistified to exploit)
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	Denial Of Service Execute Code Memory corruption
CWE ID	787

This vulnerability is present in the Apache OpenOffice (formerly OpenOffice.org) a free open source office suite. A specially crafted PPT file can lead to an out of bound write and ultimately to remote code execution.

```
Let's investigate this vulnerability. After opening Impress with the malformed PPT file we see following state:

gdb-peda$ context
[--------------------------------registers---------------------------------]
RAX: 0xb70 ('p\x0b')
RBX: 0x28a
RCX: 0x7fffc2ef0fdc --> 0x800000000010000
RDX: 0x7fffc2a4fd88 --> 0x64000020220000 ('')
RSI: 0x7fffc2a45170 --> 0x0
RDI: 0x7fffc2ef0ff0 --> 0x800000000010000
RBP: 0x28a
RSP: 0x7fffffffa7b0 --> 0x1
RIP: 0x7fffc2571dfc (mov    WORD PTR [rdi+0x10],ax)
R8 : 0x5140 ('@Q')
R9 : 0x0
R10: 0x7fffc2a4f9c8 --> 0x7ffff09eca80 --> 0x7ffff07bfa64 (<_ZN16SotStorageStream7GetDataEPvm>: push    rbp)
R11: 0xc8
R12: 0x0
R13: 0x28a
R14: 0x7fffc2a3f148 --> 0x7fffc2a77848 --> 0x7fffc2a413e8 --> 0x7b ('{')
R15: 0x7fffc2a4f9c8 --> 0x7ffff09eca80 --> 0x7ffff07bfa64 (<_ZN16SotStorageStream7GetDataEPvm>: push    rbp)
EFLAGS: 0x10202 (carry parity adjust zero sign trap INTERRUPT direction overflow)
[---------------------------------code-------------------------------------]
   0x7fffc2571df0:  mov    rax,QWORD PTR [rcx+0x8]
   0x7fffc2571df4:  mov    QWORD PTR [rdi+0x8],rax
   0x7fffc2571df8:  movzx  eax,WORD PTR [rcx+0x10]
=> 0x7fffc2571dfc:  mov    WORD PTR [rdi+0x10],ax
   0x7fffc2571e00:  mov    rax,QWORD PTR [rsp+0x28]
   0x7fffc2571e05:  movzx  r12d,r9b
   0x7fffc2571e09:  mov    r8d,ebx
   0x7fffc2571e0c:  mov    r9d,r12d
[---------------------------------stack------------------------------------]
0000| 0x7fffffffa7b0 --> 0x1
0008| 0x7fffffffa7b8 --> 0x966f08 --> 0x7ffff7ffe480 --> 0x6db1b0 --> 0x7ffff7ffe1c8 --> 0x0
0016| 0x7fffffffa7c0 --> 0x7fffffffb160 --> 0x1
0024| 0x7fffffffa7c8 --> 0x7ffff7ed8808 --> 0x7fffc27ab310 --> 0x7fffc25730b0
(<_ZNK19SdrPowerPointImport9ImportOLEElRK7GraphicRK9RectangleS5_il>:    push    r15)
0032| 0x7fffffffa7d0 --> 0x7fffffffb0b0 --> 0xfc9000f0000b10f
0040| 0x7fffffffa7d8 --> 0x7fffc2a45170 --> 0x0
0048| 0x7fffffffa7e0 --> 0x7ffff7ed9000 --> 0x10000000b
0056| 0x7fffffffa7e8 --> 0x3e8a00000000
[--------------------------------------------------------------------------]
```

Figure 3. 6: I - Source code that investigate

```
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
gdb-peda$ bt
#0  0x00007fffc2571dfc in ?? () from /opt/openoffice4/program/libmsfilter.so
#1  0x00007fffc2577c4d in SdrPowerPointImport::SdrPowerPointImport(PowerPointImportParam&, String const&) () from /opt/
openoffice4/program/
libmsfilter.so
#2  0x00007fffc27c702e in ?? () from /opt/openoffice4/program/libsdfilt.so
#3  0x00007fffc27c74e9 in ?? () from /opt/openoffice4/program/libsdfilt.so
#4  0x00007fffc27ceafb in ImportPPT () from /opt/openoffice4/program/libsdfilt.so
#5  0x00007fffc326c4c8 in ?? () from /opt/openoffice4/program/../program/libsd.so
#6  0x00007fffc318884d in sd::DrawDocShell::ConvertFrom(SfxMedium&) () from /opt/openoffice4/program/../program/libsd.so
#7  0x00007ffff4ccc02f in SfxObjectShell::DoLoad(SfxMedium*) () from /opt/openoffice4/program/libsfx.so
#8  0x00007ffff4cf2202 in SfxBaseModel::load(com::sun::star::uno::Sequence<com::sun::star::beans::PropertyValue> const&)
() from /opt/
openoffice4/program/libsfx.so
#9  0x00007ffff4d47093 in ?? () from /opt/openoffice4/program/libsfx.so
#10 0x00007fffee2a6750 in ?? () from /opt/openoffice4/program/libfwk.so
#11 0x00007fffee2a707e in ?? () from /opt/openoffice4/program/libfwk.so
#12 0x00007fffee2619a0 in ?? () from /opt/openoffice4/program/libfwk.so
#13 0x00007fffee261c0e in ?? () from /opt/openoffice4/program/libfwk.so
#14 0x00007ffff5d38868 in
comphelper::SynchronousDispatch::dispatch(com::sun::star::uno::Reference<com::sun::star::uno::XInterface> const&,
rtl::OUString const&, rtl::OUString const&, int, com::sun::star::uno::Sequence<com::sun::star::beans::PropertyValue> const&)
() from /opt/
openoffice4/program/libcomphelpgcc3.so
#15 0x00007ffff77971a3 in ?? () from /opt/openoffice4/program/libsofficeapp.so
#16 0x00007ffff77a33a6 in ?? () from /opt/openoffice4/program/libsofficeapp.so
#17 0x00007ffff77831f0 in ?? () from /opt/openoffice4/program/libsofficeapp.so
#18 0x00007ffff7784d0a in ?? () from /opt/openoffice4/program/libsofficeapp.so
#19 0x00007ffff2ec6ab3 in ?? () from /opt/openoffice4/program/libvcl.so
#20 0x00007fffea8f9b27 in SalDisplay::DispatchInternalEvent() () from /opt/openoffice4/program/libvclplug_gen.so
#21 0x00007fffeabb63b4 in ?? () from /opt/openoffice4/program/libvclplug_gtk.so
#22 0x00007fffe9aed175 in g_main_dispatch (context=0x65ea80) at gmain.c:3154
#23 g_main_context_dispatch (context=context@entry=0x65ea80) at gmain.c:3769
#24 0x00007fffe9aed4e8 in g_main_context_iterate (context=context@entry=0x65ea80, block=block@entry=0x0,
dispatch=dispatch@entry=0x1,
self=<optimized out>) at gmain.c:3840
#25 0x00007fffe9aed58c in g_main_context_iteration (context=0x65ea80, may_block=0x0) at gmain.c:3901
#26 0x00007fffeabb61f0 in ?? () from /opt/openoffice4/program/libvclplug_gtk.so
#27 0x00007ffff2caa45f in ?? () from /opt/openoffice4/program/libvcl.so
#28 0x00007ffff2ca9667 in Application::Execute() () from /opt/openoffice4/program/libvcl.so
#29 0x00007ffff777e93e in ?? () from /opt/openoffice4/program/libsofficeapp.so
```

Figure 3. 7: II - Source code that investigate

```
#26 0x00007fffeabb61f0 in ?? () from /opt/openoffice4/program/libvclplug_gtk.so
#27 0x00007ffff2caa45f in ?? () from /opt/openoffice4/program/libvcl.so
#28 0x00007ffff2ca9667 in Application::Execute() () from /opt/openoffice4/program/libvcl.so
#29 0x00007ffff777e93e in ?? () from /opt/openoffice4/program/libsofficeapp.so
#30 0x00007ffff2cad7eb in ?? () from /opt/openoffice4/program/libvcl.so
#31 0x00007ffff2cad8b6 in SVMain() () from /opt/openoffice4/program/libvcl.so
#32 0x00007ffff77a5f8c in soffice_main () from /opt/openoffice4/program/libsofficeapp.so
#33 0x0000000000400f7b in main ()
#34 0x00007ffff6646f45 in __libc_start_main (main=0x400f70 <main>, argc=0x6, argv=0x7fffffffdf18, init=<optimized out>,
fini=<optimized out>, rtld_fini=<optimized out>,
stack_end=0x7fffffffdf08)
    at libc-start.c:287
#35 0x0000000000400eb9 in _start ()


let's check the write address `rdi+0x10`:

gdb-peda$ vmmap $rdi+0x10
Start              End                Perm  Name
0x00007fffc2ef1000 0x00007fffc3568000 r-xp  /opt/openoffice4/program/libsd.so
```

Figure 3. 8: III - Source code that investigate

As we can see an attempt to write is made in an address range of the mapped file "**libsd.so**", which results in an access violation because of the pages that contain this mapped file are set to read and execute permissions, but not write. To understand why this vulnerability appears, we will look at the vulnerable function in the source code.

```
filter\source\msfilter\svdfppt.cxx : 4343

PPTStyleSheet::PPTStyleSheet( const DffRecordHeader& rSlideHd, SvStream& rIn, SdrPowerPointImport& rManager,
                            const PPTTextCharacterStyleAtomInterpreter& /

    *rTxCFStyle*/, const PPTTextParagraphStyleAtomInterpreter& rTxPFStyle,
                            const PPTTextSpecInfo& rTextSpecInfo ) :

Line 4373    DffRecordHeader* pEnvHeader = rManager.aDocRecManager.GetRecordHeader( PPT_PST_Environment );
Line 4374    if ( pEnvHeader )
Line 4375    {
Line 4376        pEnvHeader->SeekToContent( rIn );
Line 4377        DffRecordHeader aTxMasterStyleHd;
Line 4378        while ( rIn.Tell() < pEnvHeader->GetRecEndFilePos() )
Line 4379        {
Line 4380            rIn >> aTxMasterStyleHd;
Line 4381            if ( aTxMasterStyleHd.nRecType == PPT_PST_TxMasterStyleAtom )
Line 4382            {
Line 4383                sal_uInt16 nLevelAnz;
Line 4384                rIn >> nLevelAnz;
Line 4385
Line 4386                sal_uInt16 nLev = 0;
Line 4387                sal_Bool bFirst = sal_True;
Line 4388                bFoundTxMasterStyleAtom04 = sal_True;
Line 4389                while ( rIn.GetError() == 0 && rIn.Tell() < aTxMasterStyleHd.GetRecEndFilePos() &&
nLev < nLevelAnz )
Line 4390                {
Line 4391                    if ( nLev )
Line 4392                    {
Line 4393                        mpParaSheet[ TSS_TYPE_TEXT_IN_SHAPE ]->maParaLevel[ nLev ] =
mpParaSheet[ TSS_TYPE_TEXT_IN_SHAPE ]->maParaLevel[ nLev - 1 ];
Line 4394                        mpCharSheet[ TSS_TYPE_TEXT_IN_SHAPE ]->maCharLevel[ nLev ] =
mpCharSheet[ TSS_TYPE_TEXT_IN_SHAPE ]->maCharLevel[ nLev - 1 ];
Line 4395                    }
Line 4396                    mpParaSheet[ TSS_TYPE_TEXT_IN_SHAPE ]->Read( rManager, rIn, sal_True,
nLev, bFirst );
             (...)
Line 4412                    mpCharSheet[ TSS_TYPE_TEXT_IN_SHAPE ]->Read( rIn, sal_True, nLev,
bFirst );
Line 4413                    mpParaSheet[ TSS_TYPE_TEXT_IN_SHAPE ]->UpdateBulletRelSize(  nLev,
mpCharSheet[ TSS_TYPE_TEXT_IN_SHAPE ]->maCharLevel[ nLev ].mnFontHeight );
Line 4414                    bFirst = sal_False;
```

Figure 3. 9: Source code of libsd.so

First of all code in the while loop at Line 4378 searches for a PPTPSTTxMasterStyleAtom record ([MS-PPT] 2.9.35 TextMasterStyleAtom). It finds it in the file at offset 0x957c.

```
(gdb) p aTxMasterStyleHd
$8 = {nRecVer = 0 '\000', nRecInstance = 4, nImpVerInst = 64, nRecType = 4003, nRecLen = 4294901870, nFilePos = 380}
```

Figure 3. 10: PPT_PST_TxMasterStyleAtom record

Next, we see that nLevelAnz is read at line 4384 please Refer Figure 3.9. According to documentation.

but in our case its value is equal. We also see the following:

```
        PPTParaLevel    maParaLevel[ 5 ];
        and
        PPTCharLevel    maCharLevel[ 5 ];
```

Figure 3. 11: nLevelAnz source code

The lack of enforcement of the constraint that nLevelAnz must be less than 5 results in the vulnerability. The variables maParaLevel and maCharLevel are written to at lines 4393-4394 Refer Figure 3.9. Our invalid value will cause nLev to be bigger than 4 in the loop, which will result in an out of bound write. This can then lead to arbitrary code execution.

Table 3. 4: Products Affected By CVE-2017-12607

	Product Type	Vendor	Product	Version
1	Application	Apache	Openoffice	4.0.0
2	Application	Apache	Openoffice	4.0.1
3	Application	Apache	Openoffice	4.1.0
4	Application	Apache	Openoffice	4.1.1
5	Application	Apache	Openoffice	4.1.2
6	Application	Apache	Openoffice.org	-
7	OS	Debian	Debian Linux	7.0
8	OS	Debian	Debian Linux	8.0

3.2.3. CVE-2017-12608

A vulnerability in OpenOffice Writer DOC file parser, specifically in ImportOldFormat Styles, allows attackers to craft malicious documents that cause denial of service (memory corruption and application crash) potentially resulting in arbitrary code execution. **This Vulnerability can be causes to the availability of the system.**

Table 3. 5: CVSS Scores & Vulnerability Types for CVE-2017-12608

CVSS Score	6.8	
Confidentiality Impact	Partial (There is considerable informational disclosure.)	
Integrity Impact	Partial (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 (There is reduced performance or interruptions in resource availability.)	
Access Complexity	Medium (The access conditions are somewhat specialized. Some preconditions must be satisfied to exploit)	
Authentication	Not required (Authentication is not required to exploit the vulnerability.)	

Gained Access	None	
Vulnerability Type(s)	Denial Of ServiceExecute CodeMemory corruption	
CWE ID	787	

This vulnerability is present in Apache OpenOffice (formerly OpenOffice.org), a free open source office suite. A specially crafted DOC file can lead to an out-of-bounds write and ultimately to remote code execution.

Let's investigate this vulnerability. After opening Writer with a malformed doc file, we see the following state.

```
com::sun::star::uno::Sequence<com::sun::star::beans::PropertyValue> const&) ()from
/opt/openoffice4/program/libcomphelpgcc3.so
#17 0xb7dd9bf4 in ?? () from /opt/openoffice4/program/libsofficeapp.so
#18 0xb7de2a92 in ?? () from /opt/openoffice4/program/libsofficeapp.so
#19 0xb7dc61cd in ?? () from /opt/openoffice4/program/libsofficeapp.so
#20 0xb7dc650b in ?? () from /opt/openoffice4/program/libsofficeapp.so
#21 0xb7dc65b3 in ?? () from /opt/openoffice4/program/libsofficeapp.so
#22 0xb64784dd in ?? () from /opt/openoffice4/program/libvcl.so
#23 0xb66dd92e in ?? () from /opt/openoffice4/program/libvcl.so
#24 0xb2fb7de9 in ?? () from /opt/openoffice4/program/libvclplug_gen.so
#25 0xb2fc3b52 in SalDisplay::DispatchInternalEvent() () from
/opt/openoffice4/program/libvclplug_gen.so
#26 0xb3074fa9 in ?? () from /opt/openoffice4/program/libvclplug_gtk.so
#27 0xb3074fd8 in ?? () from /opt/openoffice4/program/libvclplug_gtk.so
#28 0xb2d82610 in ?? () from /lib/i386-linux-gnu/libglib-2.0.so.0
#29 0xb2d85d9b in g_main_context_dispatch () from /lib/i386-linux-gnu/libglib-2.0.so.0
#30 0xb2d86189 in ?? () from /lib/i386-linux-gnu/libglib-2.0.so.0
#31 0xb2d86254 in g_main_context_iteration () from /lib/i386-linux-gnu/libglib-2.0.so.0
#32 0xb3074d80 in ?? () from /opt/openoffice4/program/libvclplug_gtk.so
#33 0xb2fcafb9 in X11SalInstance::Yield(bool, bool) () from
/opt/openoffice4/program/libvclplug_gen.so
#34 0xb6484ff2 in ?? () from /opt/openoffice4/program/libvcl.so
#35 0xb6481dbe in Application::Yield(bool) () from /opt/openoffice4/program/libvcl.so
#36 0xb6483ccb in Application::Execute() () from /opt/openoffice4/program/libvcl.so
#37 0xb7dc32a0 in ?? () from /opt/openoffice4/program/libsofficeapp.so
#38 0xb6488d8b in ?? () from /opt/openoffice4/program/libvcl.so
#39 0xb6488e79 in SVMain() () from /opt/openoffice4/program/libvcl.so
#40 0xb7de3e10 in soffice_main () from /opt/openoffice4/program/libsofficeapp.so
#41 0x08048c84 in main ()
#42 0xb789a637 in __libc_start_main (main=0xab9bf618, argc=0xab8884df,
argv=0xab90d3fc, init=0xab963010, fini=0xbfffd1a8, rtld_fini=0xab88d9a3,
stack_end=0xaae2c5a8) at ../csu/libc-
start.c:291

The write to `eax+0x4` causes an access violation because :

gdb-peda$ vmmap $eax+4
Start      End        Perm      Name
0xab73e000 0xab795000 r-xp      /opt/openoffice4/program/libunoxml.so

Let's investigate the vulnerable code:

sw\source\filter\ww8\ww8par2.cxx
Line 4462   void WW8RStyle::ImportOldFormatStyles()
            {
                (...)
Line 4474           sal_uInt16 cstcStd;
Line 4475           rSt >> cstcStd;
Line 4476
Line 4477           sal_uInt16 cbName;
Line 4478           rSt >> cbName;
Line 4479           sal_uInt16 nByteCount = 2;
Line 4480           sal_uInt16 stcp=0;
Line 4481           while (nByteCount < cbName)
                    {
```

Figure 3. 12: I – Source code that investigate

At line 4480 we see that stcp is initialized with a 0 value. Next, if read directly from the file, cbName value won't be bigger than 2, stcp won't be increased and stay with initialized value (0).

```
                  {
                      (...)
Line 4518                 stcp++
                  }
                  (...)
Line 4521         sal_uInt16 nStyles=stcp;
Line 4522
Line 4523         std::vector<pxoffset> aCHPXOffsets(stcp);
Line 4524         sal_uInt16 cbChpx;
Line 4525         rSt >> cbChpx;
Line 4526         nByteCount = 2;
Line 4527         stcp=0;
Line 4528         std::vector< std::vector<sal_uInt8> > aConvertedChpx;
Line 4529         while (nByteCount < cbChpx)
Line 4530         {
Line 4531             sal_uInt8 cb;
Line 4532             rSt >> cb;
Line 4533             nByteCount++;
Line 4534
Line 4535             aCHPXOffsets[stcp].mnSize = 0;
                  (...)
Line 4553             stcp++;
                  }
```

Figure 3. 13: II – Source code that investigate

Based on stcp at line 4523, the aCHPXOffsets vector is allocated. The cbChpx variable value is read directly from the file at line 4525 and then used as a constrain in a while loop. The while loop will be executed as many times as indicated by cbChpx, there is no check to see whether its value is greater than stcp, which leads to an out-of-bounds write at line 4535. That situation causes memory corruption and can lead to arbitrary code execution by the attacker.

Table 3. 6: Products Affected By CVE-2017-12608

	Product Type	Vendor	Product	Version
1	Application	Apache	Openoffice	4.0.0
2	Application	Apache	Openoffice	4.0.1
3	Application	Apache	Openoffice	4.1.0
4	Application	Apache	Openoffice	4.1.1
5	Application	Apache	Openoffice	4.1.2
6	Application	Apache	Openoffice.org	-
7	OS	Debian	Debian Linux	7.0
8	OS	Debian	Debian Linux	8.0

Successfully exploiting these vulnerabilities could allow for arbitrary code execution in the context of the affected application. Depending on the privileges associated with the application, an attacker could install programs; view, change, or delete data; or create new accounts with full user rights. Failed exploitation could result in a denial-of-service condition.

3.3. Sample Attacks

Now Cisco Talos believes attackers are trying out slightly different formats that have a reputation for being overlooked by a computer's defenses. OpenDocument files (.odt) are associated with Apache OpenOffice and LibreOffice.

"Whilst less people may avail of these pieces of software the actor may have a higher success rate due to low detections. The potential for specifically targeted attacks can also increase with the use of lesser used file formats," wrote researchers Warren Mercer and Paul Racegoers. Using .odt files is not common, the report showed, but if proven successful could lead to wider spread use in the future.

In two of the attacks studied, one against English language speakers and the other Arabic, the recipient was required to open the document. At this point the object linking and embedding (OLE) object, a Microsoft technology that allows embedding and linking to documents and other objects, deployed and executed an HTA file which in turn led to a RAT being downloaded. For the Arabic targets it was NJRAT, and RevengeRAT was used in the English campaign. The final stage has the AZORult information stealer being injected into the machine.

3.3.1. Apache.openoffice.Text.Document.Malicious.Macro.Execution
Description

This indicates an attack attempt to exploit a remote Code Execution vulnerability in Apache OpenOffice.

The vulnerability is due to an error when the vulnerable software handles a crafted ODT file that contains malicious macros. A remote attacker may be able to exploit this to execute arbitrary code within the context of the application, via a crafted ODT file.

Affected products

Apache OpenOffice

Impact

System Compromise: Remote attackers can gain control of vulnerable systems. Can be harm to Confidentiality, Integrity and availability of the product.

This module generates an Apache OpenOffice Text Document with a malicious macro in it. To exploit successfully, the targeted user must adjust the security level in Macro Security to either Medium or Low. If set to Medium, a prompt is presented to the user to enable or disable the macro. If set to Low, the macro can automatically run without any warning.

References to code (For developers)

Vulnerable source code: https://github.com/rapid7/metasploit-framework/blob/master//modules/exploits/multi/misc/openoffice_document_macro.rb

History of the source code: https://github.com/rapid7/metasploit-framework/commits/master//modules/exploits/multi/misc/openoffice_document_macro.rb

4. DESIGN ANALYSIS
4.1. Architecture Overview
The OpenOffice.org source project is based on an architecture that can provide comprehensive personal productivity to different UNIX-based systems and may be ported many other platforms as well. This is because the whole technology is based on a platform-independent approach. Less than 10% of the code is platform dependent. This acts as an abstraction layer for the upper software components. Because of the availability of C++-Compilers on every major platform, C++ is used as an implementation language. This allows to port the OpenOffice.org technology to a wide range of different platforms. The decision for an object-oriented language gives the OpenOffice.org source project the opportunity deliver a fully object-oriented architecture.

The following information will give just a rough overview over the overall architecture. Some components of the OpenOffice.org source project like the help-system or the setup application are

not covered here. Many parts of the OpenOffice.org source project consists of more than one CVS module. In many cases one block in the architecture is covered by more than five CVS modules in the source tree.
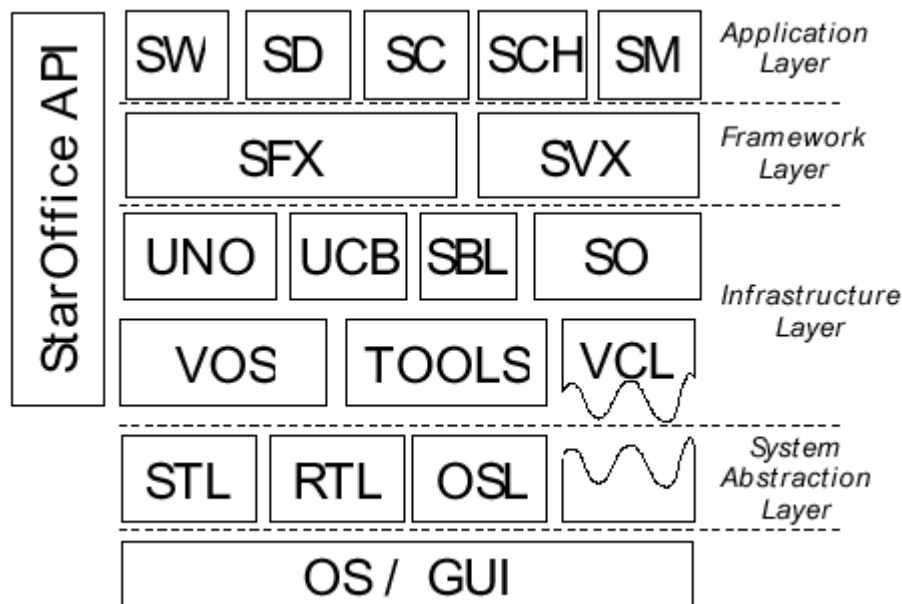
Figure 4. 1: Apache Open Office Architecture Overview

4.1.1. System Abstraction Layer

This layer encapsulates all system specific APIs and provide a consistent object-oriented API to access system resources in a platform independent manner.

Operating system layer

The operating system layer (OSL) encapsulates all the operating system specific functionality for using and accessing system specific resources like files, memory, sockets, pipes, etc. The OSL is a very thin layer with an object-oriented API. In contrast to the upper layer this object-oriented API is a C-API. This will allow to easily port this layer to different platforms using different implementation languages. For embedded systems or internet appliances for examples an assembler language can be used to realize the implementation.

Runtime library

The runtime library provides all semi platform independent functionality. There is an implementation for string classes provided. Routines for conversion of strings to different character sets are implemented. The memory management functionality resides in this module.

Standard Template library

As a generic container library, the standard template library is used. It supplies implementations for list, queues, stacks, maps, etc.

4.1.2. Infrastructure Layer

A platform independent environment for building application, components and services is provided by this layer. It covers many aspects of an object-oriented API for a complete object oriented platform including a component model, scripting, compound documents, etc.

Visual Class library

The visual class library is one of the core libraries of the OpenOffice.org technology. The VCL encapsulate all access to the different underlying GUI systems. The implementation is separated into two major parts. One is completely platform independent and includes an object oriented 2D graphics API with metafiles, fonts, raster operations and the whole widget set use by the OpenOffice.org suite. This approach virtually guarantees that all widgets have the same behavior independently of the used GUI system on the different platforms. Also the look & feel and the functionality of the widgets are on all platforms the same.

Because of this design VCL doesn't encapsulate the native widgets or controls of the underlying GUI system. The platform dependent part implements a 2D-graphic drawing canvas which is used by the platform independent parts. This canvas redirect every functionality directly to the underlying GUI system. Currently there exists implementation for the Win32, X-Windows, OS/2 and Mac. The access to the printing functionality, clipboard and Drag & Drop is also realized inside the VCL.

Virtual Operating System layer

To make the usage of system resources like files, threads, sockets,etc. more convenient the virtual operating system layer encapsulates all the functionality of the operating system layer into C++ classes. The C++ classes here offer an easy to use access to all system resources in an object oriented way.

Tools libraries

There are different small libraries building up a set of tool functionality. This includes a common implementation for handling date and time related data. There is in implementation for structured storages available. Other implementation provide a generic registry, typesafe management and persistence of property data.

Universal Network Objects

The so called Universal Network Objects are the component technology used inside the OpenOffice.org products. The component technology does not depend on any graphical subsystem, but is heavily based on multithreading and network communication capabilities.

The system consists of several pieces. An IDL-Compiler, which generates out of the specified definition of an interface a binary representation and the associated C-Header or Java technology files. The binary representation is platform and language independent and is at runtime used to marshall argument for remote function calls or to generate code on the fly for a specific language to access the implementation provided by the interface. This technique reduced the amount of

generated code for the different language binding tremendously. The drawback is that not only for every language binding a specific backend for the code generation is needed, it is that for every specific compiler a bridging module is needed at runtime.

Many parts of the UNO technology are implemented as UNO components. This helps to create a very flexible system and also the extension of the system at runtime. For example by providing new bridges or communication protocols. UNO provides transparent access to components over the network or locally. For the communication over the network IIOP can be used. If the component are realized as shared libraries the component can be loaded by UNO into to the process memory of the application and every access of the component is just like a function call without any marshalling of arguments which is required for remote function call.

Universal Content Broker

The Universal Content Broker allows all upper layers to access different kind of structure content transparently. The UCB consists of a core and several Universal Content Providers which are used to integrate different access protocols. The current implementations provides content providers for the HTTP protocol, FTP protocol, WebDAV protocol and access to the local file system.

The UCB does not only provide access to the content, it also provides the associated meta information to the content. Actually there is synchronous and asynchronous mode for operations supported.

Compound Objects

The Compound Object implementation provide the functionality to build compound documents, where for example a spreadsheet is being embedded in a word-processor document.

The current implementation provides a platform independent implementation of all this functionality for compound documents and for embedding of visual controls like multi media players or different kind of viewers. All content of compound document is stored in a structured storage. The current implementation is compatible to the OLE structure storage format. This allows access to OLE compound documents on every platform where OpenOffice.org is available. On the Windows platform the implementation interact with the OLE services and will so allow a tight integration of all OLE capable applications.

Scripting and Basic library

The scripting functionality coming with the OpenOffice.org suite is a BASIC dialect featuring an interpreter that parses the source statements and generates meta instructions. These instructions can be executed directly by the supplied meta instructions processor or can be made persistent in modules or libraries for later access. All functionality supplied by the upper level application components is accessible via a scripting interface in the component technology. This will help to ensure that new components using the OpenOffice.org component technology can be fully scriptable without spending a huge amount of effort.

The scripting interfaces are also implemented as components which will allow an easy integration of other scripting languages. The interfaces provide functionality like core reflection and introspection similar to the functionality by the Java platform.

4.1.3. Framework Layer

To allow the reuse of implementations in different applications the layer provides the framework or environment for each application and all shared functionality like common dialogs, file access or the configuration management.

Application framework library

The Application framework library provides an environment for all applications. All functionality shared by all application and not provided by any other layer is realized here. For the framework every visual application has to provide a shell and can provide several views. The library provides all basic functionality so only the application specific features have to be added.

The Framework is also responsible for content detection and aggregation. The template management is provided here and the configuration management too. The Framework is in some areas related to the compound documents, because of the functionality for merging or switching menu- and toolbars. Also the capability for customization of all applications is provided by the library.

SVX Library

The SVX library provides shared functionality for all applications which is not related to a framework. So part of the library is a complete object oriented drawing layer which is used by several applications for graphic editing and output. Also a complete 3D-rendering systems is part of the drawing functionality.

The common dialogs for font selection, color chooser, etc. are all part of this library. Also the whole database connectivity is realized here.

4.1.4. Application Layer

All OpenOffice.org applications are part of this layer. The way these applications interact is based on the lower layers The chart shown below was created to depict the architecture of the Star Office suite but it is the same for the OpenOffice.org suite.

4.2. The OpenOffice API database integration

Star Database (SDB) is the highest layer. This layer provides an application-centered view of the databases. Services, such as the database context, data sources, advanced connections, persistent query definitions and command definitions, as well as authentication and row sets are in this layer.

Star Database Connectivity Extension (SDBCX) is the middle layer which introduces abstractions, such as catalogs, tables, views, groups, users, columns, indexes, and keys, as well as the corresponding containers for these objects.

Star Database Connectivity (SDBC) is the lowest layer. This layer contains the basic database functionality used by the higher layers, such as drivers, simple connections, statements and result sets.

4.3. Architecture Overview OpenOffice Address Book Integration

This is an overview of the technologies related to the integration of the address-book feature into OpenOffice.org and Mozilla, a summary of the address-book types supported and their respective underlying API's, and an architecture diagram that shows the shared and unique parts distributed with OpenOffice.org and Mozilla. An appendix lists the Mozilla files installed automatically by OpenOffice.org, including platform variations.

OpenOffice.org has its own open database access API called SDBC (Sun DataBaseConnectivity) which is modeled on the architecture of Java's JDBC. SDBC supports several databases and database-like API's via an extensible provider architecture; ODBC 3.0, JDBC, ADO, dBase and CSV are among the databases and database management systems supported. Each SDBC provider is implemented as a UNO component; UNO is OpenOffice.org's component technology.

Mail applications such as Mozilla Messenger and Microsoft Outlook often have a local address-book (contacts) database, as well as enterprise LDAP-based directory access which is often used as a shared address-book. Mozilla extensions are implemented as XPCOM components; XPCOM is Mozilla's cross-platform ("XP") component (object model) technology.

Table 4 1: Supported Address Books

Mail Client	Address Book Type	API
Any	LDAP	Mozilla's LDAP XPCOM component
Mozilla	Local address book	Mozilla address-book XPCOM component
Outlook	Outlook Personal Address Book (PAB)	MAPI
Outlook Express / MS Mail	Windows Address Book (WAB)	WAB API

4.4. Architecture overview of Address book

As noted above, it was decided to provide open address-book access for both Mozilla's mail client and OpenOffice.org SDBC using a common set of components. Since the integration of new features into Mozilla is achieved using XPCOM, it was decided to develop an adaptor for Outlook and Outlook Express address-books and expose the adaptor using an XPCOM API, and also to support LDAP address-books via an existing XPCOM LDAP component.

OpenOffice.org makes extensive use of it's own UNO object component technology. Similar to the address-book XPCOM component-based provider architecture devised for Mozilla, the SDBC architecture supports providers for many different data sources with each new source having its own adaptor UNO component. To integrate the AbZilla address-book components into OpenOffice.org, an address-book data-source must be exposed via a UNO component.

The architecture chosen was to implement the address-book providers principally using XPCOM to achieve Mozilla integration. These XPCOM interfaces are then wrapped with a private C++ API which is in turn exposed to OpenOffice.org via a UNO interface adaptor. This architecture is summarized in the following diagram.
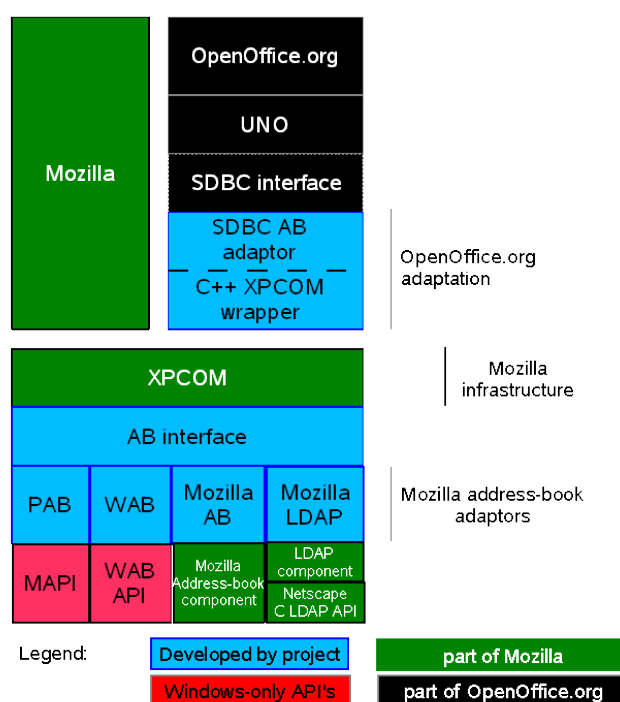
Figure 4. 2: OpenOffice and Mozilla, on a common address-book architecture

OpenOffice.org ships two libraries that expose the Mozilla address-book functionality to OpenOffice.org's SDBC framework:

- libmozab2.so which implements a UNO-based SDBC provider and uses a (private) C++ API provided by libmozabdrv2.so

- libmozabdrv2.so which exposes a (private) C++ API and uses the AbZila address-book XPCOM-based components (and indirectly uses XPCOM and it's dependencies)

Since the PAB, WAB Moz-AB and Moz-LDAP address-book providers expose XPCOM API's, the core XPCOM infrastructure and related dependencies are required to use them.

For OpenOffice.org to make use of the AbZilla XPCOM-based address-book components, OpenOffice.org now ships with a total 50 Mozilla files, comprising:

- the essential libraries that implement Mozilla's XPCOM technology
- configuration files
- some other dependent libraries including JavaScript to access XPCOM's configuration
- the library containing the four specific address-book related XPCOM components (PAB, WAB, Mozilla AB, and Mozilla LDAP)

4.5. Apache Open Office Software Architecture
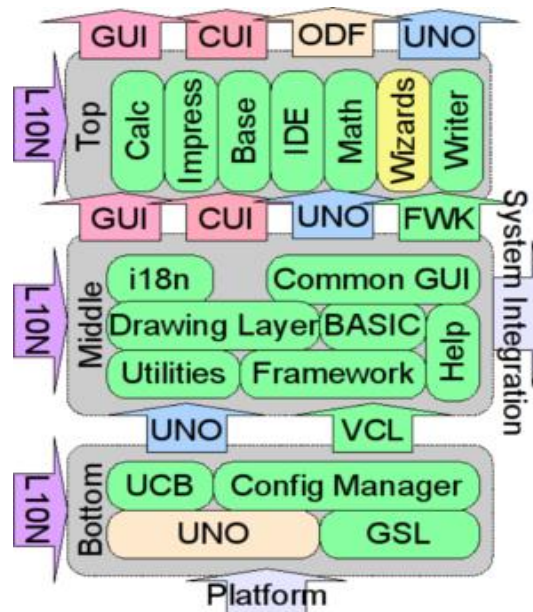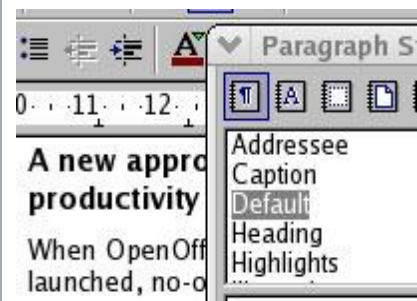
Figure 4. 3: Apache Open Office Software Architecture

As most of you know, OOo is a complex and big software system. Many lines of code are implemented in various programming languages, including BASIC, C, C++, Java, and more (perl, make, ANT, ...), if one counts the build system or tests as being part of OOo. Obviously, it is necessary to follow some kinds of guidance, rules, principles or constraints, to make it work seemingly and to make it understandable.
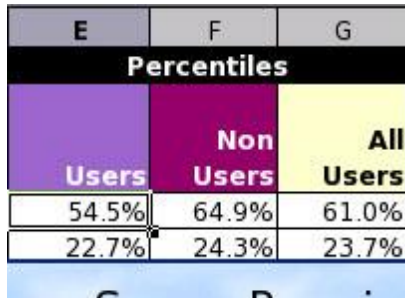
4.6. Apache OpenOffice Components Suite

OpenOffice.org is a unified suite of productivity applications for all common office applications, including such functions as word processing, spreadsheets, drawings, presentations, data charting and formula editing. All components of the suite employ the same user interface concepts and underlying technology. They interoperate closely with one another, supporting features like inter-application copy-and paste and drag-and-drop for creating compound documents. It is straightforward to embed a spreadsheet in a text document, for example. They also interoperate well with other common desktop productivity application suites, including the various office productivity applications produced by Microsoft, for ease of document exchange. A scripting environment called OpenOffice.org Basic is supported in all OpenOffice.org components to automate work or build solutions.

Table 4 2: Apache OpenOffice Component

Word Processor	
	The OpenOffice.org word processor application is a powerful tool for creating professional documents, reports, newsletters, and brochures. It is easy to integrate images and charts in documents, create business letters and extensive text documents with professional layouts, as well as create and publish Web content.

Spreadsheet	
	The OpenOffice.org spreadsheet application features decision making analysis tools for performing advanced spreadsheet functions and data analysis. Charting tools generate presentation application, high-quality 2D and 3D charts.
Presentation Tool	
	The OpenOffice.org presentation application is a tool for creating multimedia presentations. Included are 2D and 3D clipart, special effects animation, and high-impact drawing tools.
Drawings and Diagrams	
	The OpenOffice.org drawing application is a vector-oriented drawing module that enables the creation of dynamic 3D illustrations and special effects.
Database Access	
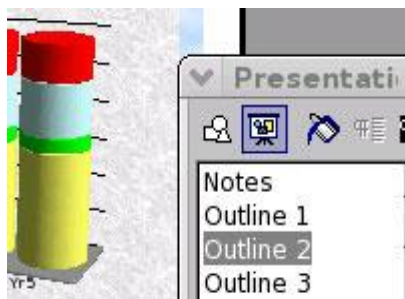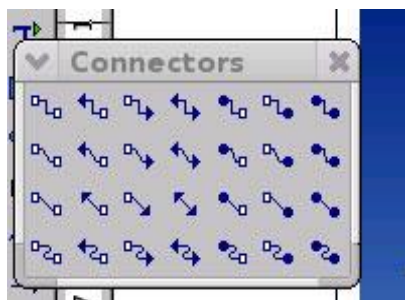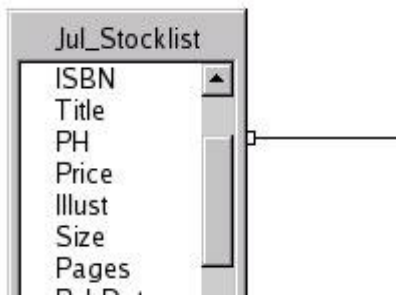
	The OpenOffice.org data charting application presents complex data in visually presentation application ways, from colorful 3D charts to simple pie, bar, and line diagrams.
Microsoft Office Compatibility	
	OpenOffice.org is able to read and write Microsoft Office files. This allows users to open and save Word, Excel and PowerPoint files on their preferred platform incl. Windows, Linux and Solaris.
One-click Export to PDF	
	OpenOffice.org 1.1 introduces the one-click PDF export feature that enables you to easily create PDF files without the need for any additional third party software. This feature makes exchanging documents in a standard "read-only" file format a trivial task. The creation of PDF files normally requires relatively expensive third party add-on tools. With OpenOffice.org this feature comes for free.
Export to Flash (.SWF)	
	OpenOffice.org now can export presentations and drawings to the Macromedia Flash format (.swf). Thus, it's now possible to view presentations in a simple web browser that has the Flash plugin installed. Recipients and users of Flash presentations don't have to install a special viewer anymore in order to view a presentation.
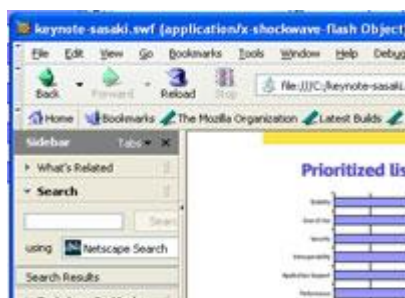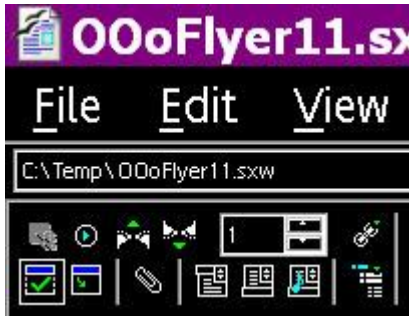
Accessibility	
	OpenOffice.org 1.1 can now be used by handicapped people - for example people with sight problems. OpenOffice.org 1.1 provides a high contrast mode, and together with additional tools (see http://www.sun.com/access/) it's even possible to use special entry devices.
Support for many native languages	
	OpenOffice.org 1.1 introduces functionality like bi-directional and vertical writing that is required for many native languages. This allows OpenOffice.org 1.1 to be translated into Japanese, Hebrew and many other languages that have sophisticated text layout requirements. In addition the OpenOffice.org project has an increasing number of native-language projects where users can access OpenOffice.org information in their native language.
Open XML File Format	
	The default file format in OpenOffice.org is an open XML file format defined in a 500-page specification document. Every OpenOffice.org file is a ZIP archive containing separate XML files for the content, styles, settings and meta data. OASIS is using the OpenOffice.org file format as the basis for the creation of a industry wide standard for an open office document file format. For ordinary end users this means that the content of documents can still be accessed and used even if OpenOffice.org would go away.
API and File Format Compatible with StarOffice 7[tm] Office Suite	
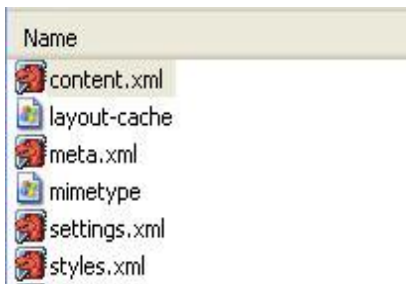
StarOffice™ 7 Copyright © 2003 Sun Microsystems, Inc. Al terms. Third-party software, including font tech suppliers. Sun, Sun Microsystems, the Sun lc trademarks or registered trademarks of Sun M countries. Federal Acquisitions: Commercial S Standard License Term and Conditions. Powered by Software AG This product is based on the OpenOffice.org pri	Since the StarOffice 7[tm] Office Suite is based on OpenOffice.org 1.1 both application suites are API and file format compatible. Sun Microsystems, Inc., the founder and main sponsor of the OpenOffice.org project, is providing support, training, enterprise tools and services for the StarOffice 7 Office Suite. More details about the relationship and the feature differences between OpenOffice.org and the StarOffice product can be found in a comparison document on the OpenOffice.org page at sun.com. Other office suites that are API and file format compatible with OpenOffice.org will soon be listed on the OpenOffice.org website. In order to get your OpenOffice.org based/compatible product included in that list please contact Erwin Tenhumberg or Louis Suarez-Potts.

Macro Recorder
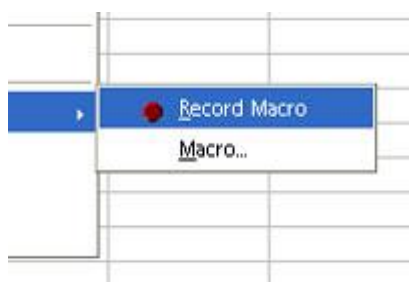
Record Macro Macro…	The new macro recorder records and automates recurring tasks. For more sophisticated programming tasks the OpenOffice.org Software Development Kit (SDK) can be used. The SDK provides libraries, tools and documentation for the Java programming language, C++, Basic, OLE and XML. The SDK is a separate download.

3rd Party Add-ons
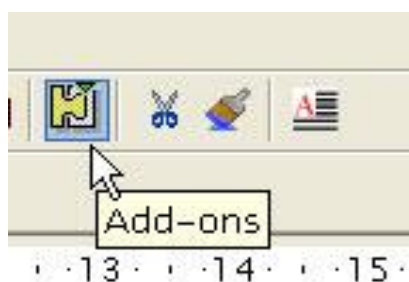
Add-ons ·13· · ·14· · ·15·	The new add-on framework and the new deployment tool (the "pkgchk" utility) allow developers to easily include new components and 3rd party add-ons into an existing OpenOffice.org installation.

XML File Filter Tool
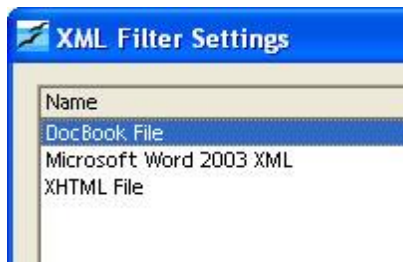
	The new XML file filter tool makes it easy to integrate new import or export file filters that are based on XSL transformations. A popular use case for the XML filter tool is the support for the new Microsoft Office 2003 XML file formats.

ActiveX Control

	OpenOffice.org 1.1 comes with an ActiveX control that allows users to view OpenOffice.org documents within a browser window (Internet Explorer) on the Windows platform. The ActiveX control can also be used within native applications developed in Microsoft Visual Basic or Borland Delphi.

Support for Doc Book and PDA File Formats
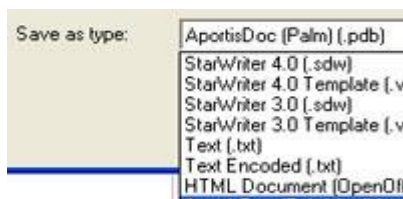
	OpenOffice.org allows users to export OpenOffice.org documents to popular file formats like Doc Book or various small device file formats like AportisDoc. This gives users the possibility to carry their documents with them on their Palm Pilots and Pocket PC PDA's.

MySQL Integration
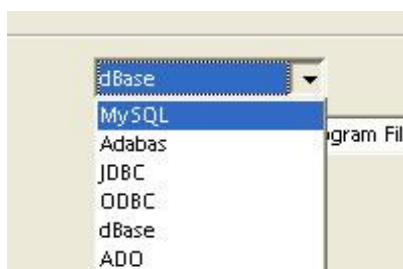
	OpenOffice.org now has support for MySQL databases, i.e. the open source database can be used for mail merge activities and the creation of reports. The combination of OpenOffice.org and MySQL (or another supported database) allows users to do tasks that many people used to solve by using products like Microsoft Access.

4.7. The OpenOffice Component Technology

The OpenOffice suite provides a component technology named Universal Network Objects, which adheres to all these requirements of modern Component Ware and it is formed on the Object Technology level, which is the basis upon which the OpenOffice.org API is set up.

This component technology is:

- Open
  It supports all the popular component standard communication protocols such as CORBA, JavaBeans, OLE Automation (Windows Scripting Host, Visual Basic, Delphi, and so forth), JavaScript, Phyton, Perl, etc. scripting languages, as well as native integration in the C++ and C programming languages.

- Object Oriented
  It is object oriented and therefore supports concepts such as aggregation, inheritance, exception handling and polymorphism.

- Interface Based
  Its functions are integrated into various interfaces. Function areas of similar structures have access to the same interfaces, so that the programmer can easily feel at home in the component world.

- Platform Independent
  It is specified to be platform independent and is available on all platforms that run OpenOffice.org.

- Exception Able
  It offers exception ability, which means that it can allow itself to be mapped onto the inserted development system mechanism - for example onto C++ Exceptions and Java Exceptions.

- Development System Independent
  It can be used with all current development environments and programming languages, including C++, C, Visual Basic, Windows Scripting Host and all systems which support COM, CORBA, JavaBeans components, and OLE Automation.

- Network Able
  Components based on the component technology can communicate on a network and can also delegate functions on a remote server, for example, to offer access to the complete text processing functions on Internet appliances.

4.8. Design principle of the Apache Open Office
4.8.1. Design Principles
Some principles that are important in all our designs are:

- orthogonality
  The API consists of interfaces which can easily be combined to serve special needs of certain objects.

- scalability
  We distinguish between functionality that is commonly needed and that which is required by specialized versions. Developers can always start with a minimum set of interfaces and add more step by step to embody more features.

- reusability
  We avoid creating specialized interfaces when a generic version is possible.

- Remote usability
  Services can be used efficiently from different processes or even different machines.

- multithread enabled
  Services can be used from multiple threads.

4.8.2. Architectural Paradigm

Our architectural decision is Interfaces and Support-Classes instead of Implementation Inheritance.

Interfaces and Support-Classes means that objects communicate only by interfaces. Support classes are used for recurring implementations. This was the choice for our design because components are highly independent of environment, language and version Implementation Inheritance means partly implemented base classes from which specialized classes derive interface and implementation. This paradigm was not our choice, because in larger systems this leads to fat interfaces or deep inheritance hierarchy. In addition, components depend on the environment, mostly via their base class, and are programming language dependent and highly version dependent.

4.8.3. Object Model

The OpenOffice.org API is designed for and implemented using the OpenOffice.org Component Technology. Therefore the OpenOffice.org API is programming language independent and can be used from C/C++, Java, and several scripting languages. For other languages, only a language binding needs to be provided to access the whole OpenOffice.org API. The API is made up by following stereotypes:

- implementation classes
  These are not actually part of the API, but are mentioned here for better understanding. Implementation classes are implementations of services using a real programming language. Normally developers who use services do not have to deal with the implementation itself on the API level. Objects are usually not translated into language concepts for an application that uses them, but rather for the implementer of the class. A good example is an implementation class similar to a concrete Java-class.

- services
  Specifications of objects are called services. One can think of a service as a contract which is beneficial to both sides: the implementation class that supports certain services and the application that uses this component for those services. Services usually describe the

interfaces which they implement and a set of their properties. Although services are normally not translated into language concepts, a service may be considered to be similar to an abstract Java class.

- interfaces
Specifications of a single aspect on an API level are called interfaces. A service can be considered to be a legally proven text module for contracts. Services can be combined to create contracts. Interfaces are very much like Java technology interfaces.

- structs
Plain data blocks are specified as structs . Structs, therefore, do not have methods. The advantage of structs is that they can transferred as they are to a different process or even a different machine, which increases efficiency of interprocess and remote calls. With Java technology, structs are represented as a class which consist only of data members and get and set methods.

- exceptions
Exceptions are extraordinary results from method calls. Exceptions are used for error handling just as in the Java technology.

- constants/constant groups/enums
Constant values are split into two categories: constants - which can be grouped and can have numeric or character string values, and enums - which contain a fixed set of numeric values. In the Java technology, both are represented as classes with constant data members.

4.8.4. Common Design Patterns

The OpenOffice.org API uses heavily designed patterns, which provide a very consistent overall design. Some examples of application domain unspecific design patterns are:

- Factory environment/container
New instances of services are created using factories. Factories of contents can emanate from the container object or from the environment. For efficiency, factories for iterators of all kinds, including cursors, are provided by the container.

- Property Sets/-Access etc.
A set of interfaces makes the properties specified in the services accessible. Properties can be viewed as non-structural data members of objects, such as color or font. The variety of access interfaces covers the spectrum from convenient local access to fast remote access.

- Collections/Containers
A collection in our terminology gives access to a set of similar sub-objects. A container allows replacement, insertion and removal of these sub-objects. A wide variety of predefined interfaces is available for this design pattern.

- Enumerators/Iterators/Cursors
Enumerators, iterators and cursors are used for efficiently listing contained objects. Cursors in text play an especially major role, because indexing of text is not reasonable.

- X...Supplier
  To access structural but optional data members, we frequently offer supplier interfaces which in many cases only have one get method.

- Events
  Where it is of interest to get notification about certain status changes, there are services that offer methods to register and unregister listener interfaces. When the event occurs, a method at the listener interface is called and the event is given as an argument. This is the same as in the event concept in JavaBeans.

- Exceptions for Error Handling
  Exceptions constitute our principal error handling concept. For asynchronous calls, exceptions are transferred in callback events.

4.8.5. Module Categories

The OpenOffice.org API is organized in a hierarchical module concept which follows Java technology package or CORBA conventions.

- office specific interfaces
  e.g. For text documents, spreadsheet documents, drawing and presentation documents

- integration framework
  These interfaces make it possible to integrate new components into OpenOffice.org, e.g.: configuration management , Universal Content Broker

- application domain independent
  This very important category contains interfaces for property access, collections and contains, or streaming operations, as well as for attaching scripting engines and many more interfaces.

- component system
  The base of the suite is the handful of interfaces which are necessary to deal with object model topics such as lifetime control, querying interfaces, building bridges and instantiating remote objects.

5.THREAT MODELING FOR OPEN SOURCE SOFTWARE

Figure 5. 1: Threat Modeling Process

Threat modelling works to identify, communicate, and understand threats and mitigations within the context of protecting something of value.

Threat modelling can be applied to a wide range of things, including software, applications, systems, networks, distributed systems, things in the internet of things, business processes, etc. There are very few technical products which cannot be threat modelled; rewarding, depending on how much it communicates, or interacts, with the world. Threat modelling can be done at any stage of development, preferably early - so that the findings can inform the design.

Step 1: Identify security objectives

Understand security requirements and identify possible threats in business flows to achieve objectives. You should also consider if there are any specific compliance or security-related requirements that are a part of the business objectives. For example, during auditing, sensitive information (e.g. SSN number, age etc.) should not get logged and the log file should be accessible to a specific set of users only.

Step 2: Identify assets and external dependencies

Unauthorized access to assets such as data, code, and system information are the reason why threats occur. The security architect needs to identify a list of assets to be protected from potential attackers. They must also identify external dependencies which are not part of code but may pose a threat to

the system. Consider how the application will be accessed on the production environment or the web server. Or how database communication will take place on a private or public network.

Step 3: Identify trust zones

Architects must identify a trust zone and corresponding entry-exit points. This information should be documented and used to develop data flow diagrams with privilege boundaries. This helps define the approach to user authentication, input data validation, and error handling. In the e-commerce website example we talked about earlier, the order processing system can be identified as a trust zone that will need a price validation check against the ordered item ID.

Step 4: Identify potential threats and vulnerabilities

Besides running a wide search for threats under a predefined approach like STRIDE, consider threats that would generally impact your system. Some examples could be - SQL injections, broken authentication, and session management vulnerabilities. Identify risk-prone areas like poor input validations, over privilege accounts, weak password policies, custom encryption, inadequate auditing or logging, displaying error or exception messages to end user.

Step 5: Document threat model

Threat modelling is an iterative process and documentation forms an important aspect of the team's responsibilities. Architects can use documentation to create secure design/architecture and mitigate architecture related security threats. Developers can use documentation as security guidelines to mitigate security risks and testers to drive test cases to find vulnerabilities in the system. It helps the tester create security related test cases and validation test cases for trust zone

Threat modelling should start with design phase and run parallel with architectural design. Moreover, it is important to remember that there is no single approach to threat modeling.
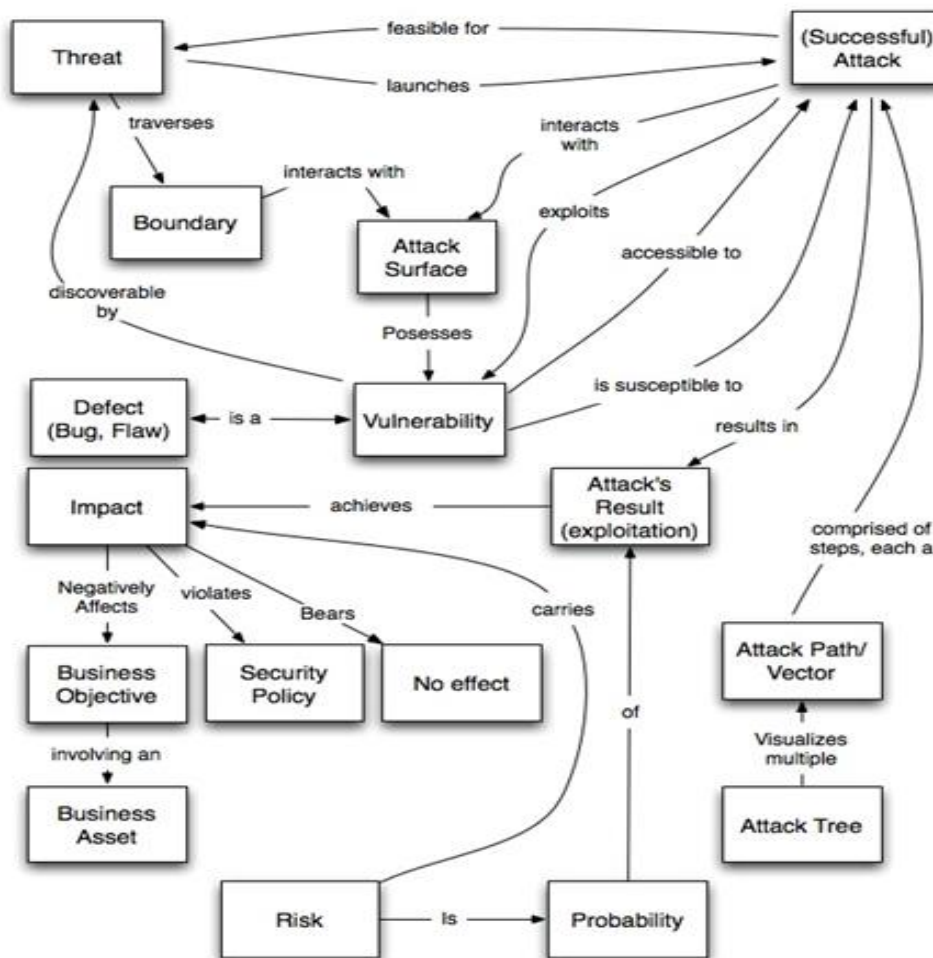
Figure 5. 2: Placement of threat

5.1. Security Concepts for Open Office

5.1.1. Encryption

Encrypted documents should not have any non-encrypted content.

Currently it is possible to add any non-encrypted content into encrypted documents, simply by putting new streams into the zip archive.

This way, someone could for example add macro code into documents without the need to know the password.

The possibility to add content to an encrypted document is quite bad, and it becomes worse with macros, because passwords increase the level of trust the user has to a document, when it was encrypted by a person he knows.

OOo will detect the macros and ask the user whether or not to execute them. The user might trust the author of the document, and because of the encryption normally nobody should have been able to manipulate it, so he will probably allow OOo to execute the macros. At least, it's not possible to bind the macros to some event, so it would get executed automatically.

In general, an encrypted document shouldn't have any non encrypted content. Exceptions are some files in the META-INF folder: manifest.xml, which is needed to get the encryption information, and the digital signature files also might not be encrypted, depending on the signature implementation.

OOo should show a warning when an encrypted document contains streams which should be encrypted, but aren't. Additionally, macros should be disabled for the document. (issue #i103927)

The encryption process takes place in the following multiple stages:

- The start key is generated and is provided to the package component.
- The derived key is generated by the component based on the start key.
- The files are encrypted based on the derived key and the encryption algorithm.
- The implementation must support at least the described below default way for the mentioned steps.

In addition it might support encryption, where algorithms different from the default are used for the steps. The information regarding used for each step algorithms has to be provided in the manifest.xml accordingly.

The default way is:

The start key is generated. The byte sequence representing the password in UTF-8 is used to generate a 20-byte SHA1 digest. The result star key is passed to the package component.

The derived key is generated by the package component from the start key. The PBKDF2 algorithm based on HMAC-SHA-1 function is used for the key derivation. The random number generator initialized with the current time is used to generate 16-byte salt for each file. The salt is used together with the start key to derive a unique 128-bit key for each file. The default iteration count for the algorithm is 1024.

The files are encrypted. The random number generator is used to generate the 8-byte initialization vector for the algorithm. The derived key is used together with the initialization vector to encrypt the file using the Blowfish algorithm in cipher feedback (CFB) mode (blowfish).

Each file that is encrypted is compressed before being encrypted. To allow the contents of the package file to be verified, it is necessary that encrypted files are flagged as 'STORED' rather than 'DEFLATED'. As entries which are 'STORED' must have their size equal to the compressed size, it is necessary to store the uncompressed size in the manifest. The compressed size is stored in both the local file header and central directory record of the Zip file.

Improving the Encryption Implementation in OOo 3.2

For OOo 3.2, we plan to improve the document encryption, or better, the handling of documents which have issues like mentioned above (issue #i103927).

When OOo opens an encrypted document, it will check via the manifest if all files are encrypted. Some encrypted files in a ODF zip archive still might use the encryption key, different from the document encryption key.

When detecting not encrypted files, OOo will show a warning to the user, and will disable the execution of macros in the document.

Enhancing the Encryption Specification for ODF 1.2

Currently the ODF specification specifies one way for encrypting ODF documents, and ODF applications can't chose different algorithms.

Choosing other encryption algorithms can be important, to comply with encryption rules from a company or Government using ODF.

The information about the used algorithms (for each step) need to be documented in the manifest.xml.

Comply to the updated ODF 1.2 Encryption Specification in OOo 3.2

OOo 3.2 needs to comply with the updated encryption specification above. This mainly means that all algorithms used in the different steps needs to be documented in the manifest.xml. It does not mean that OOo would implement other algorithms now.

Missing Encryption of manifest.xml

To some people it looks surprising that the manifest.xml is not encrypted.

Actually, the manifest.xml contains the information that any ODF application needs to open the document, including the encryption and hash algorithms which have been used to encrypt the document...

Protecting the not encrypted manifest.xml in an encrypted document

Since the manifest.xml can't be encrypted, someone could make modifications to manifest.xml in encrypted documents.

It needs to be discussed if there needs to be some protection against this.

For example, we could generate some signature/hash for encrypted version of manifest.xml (with the common document encryption key), and store it in a separate stream in the META-INF folder.

This could be some OOo-only solution, and OOo would only warn when the signature is missing in documents written with OOo, or it could become part of some future version of the ODF specification.

Public Key Document Encryption

It would be a nice feature to allow public key encryption for encrypting documents. People could encrypt the documents with their own and/or with other people's public key certificates, so only the owners of the corresponding private keys would be able to open the document. No need to exchange or remember passwords.

5.1.2. Digital Rights Management

OpenOffice.org could support DRM. The part "user can/can't open a document" makes sense, can be implemented reliable. Enhanced rights/restrictions like "can copy/print/..." can't be guaranteed, because they could be removed easily in an open source application.

Password length and password pattern

The ancient minimal password length in the password dialogue should be removed, but don't allow empty passwords. (issue #i103783)

Ooo-password-properties: MinPasswordLength (default=1 if not set), MinNumbercharacters, MinSpecialCharacters, ForbiddenCharacters.

We would then also need some dialogue which explains the password rule to the user.

Thumbnails in encrypted documents

ODF documents can contain a thumbnail, to display a small picture of the first page in a file explorer.

Currently, encrypted documents contain encrypted thumbnails, which is not good for anything.

Encrypted documents shouldn't store thumbnails.

The reason for this is that encrypted documents mustn't contain any non encrypted streams (with some exceptions). And since the encrypted document contains a standard bitmap as a thumbnail it makes no sense to store it in the document anyway. The bitmap should be part of the system integration.

Document Integrity

The best way to ensure document integrity is to digitally sign the documents.

But most people won't do it, or even don't have the infrastructure for this, so there should be some light weight mechanism for checking the document integrity

ODF conformance clause

The ODF specification should have some conformance clause, that all files, except package meta data in META-INF folder (signature streams for example) and the mimetype stream, must be registered in manifest.xml.

With the current ODF specification, the META-INF folder should only contain manifest.xml, and might contain some signature streams (*signatures.xml).

When this is defined, an ODF application should not load any files which are not registered in manifest, or show a warning to the user.

Make sure OOo 3.2 would adhere to the ODF conformance clause

Independently from whether or not the conformance clause above will make it into the ODF specification, OpenOffice.org should create documents adhering to the definition above. That should be already the case in OOo3.1 except the signature stream that is still registered in manifest.xml. That will be changed for OOo3.2

Check ODF integrity in OOo 3.2

Assuming that the conformance clause above will be added to the ODF specification, OOo should warn when loading documents containing streams not registered in manifest.xml.

The check/warning is not necessary when the document has a broken signature.

This shouldn't be a problem for older documents (written with OOo), because OOo is already registering all files in manifest.xml

Maybe the check should only be done for ODF 1.2 (and newer) documents, to avoid interop problems with documents written by other applications.

5.1.3. Digital Signatures
Document files and package files, that is the files which carry meta information for the package, such as manifest.xml,

Digital signatures are stored in one or more files within the META-INF folder. The names of these files shall contain the term "signatures". Each of these files contains a <dsig:document-signatures> root element that serves as a container for an arbitrary <Signature> element as defined by the [xml-dsig] specification. If the <dsig:document-signatures> element contains multiple <Signature> elements, then there should be a relation between the digital signatures they define, for instance, they may all apply to the same set of files.

Applications may require that a digital signature includes a certain set of files. That is, they may consider a digital signature to be valid if, and only if,

the digital signature itself is valid, and

if the <Reference> child elements of the <Signature> element reference a certain set of files.

In particular, application may require that a digital signature references all files contained in a package.

The schema for digital signatures is To distinguish between the "old signature" and the one produced by OOo 3.2, the signature component will check if the manifest.xml is part of the signature. If not, then the validation algorithm of OOo 3.0. is used. However, even if the signature validates correctly, it will be indicated, that this is an "old" signature. The wording will be something like "Partial signature", because the old signature did not comprise all files of the

document (the ZIP archive). Signatures created by OOo 3.2 will be displayed as invalid in previous versions of OOo.

Improving the Digital Signature Implementation in OOo 3.2

The implementation of the digital signatures need to be changed to match the updated Digital Signature Specification for ODF 1.2. The manifest.xml needs to be included in the document signature, as well as other streams in META-INF. The document signature stream itself might be excluded.

As a side effect, it will not be possible anymore to sign the macros after signing the document, because the macro signature stream must be included in the document signature. The corresponding menu items should be disabled, or a information needs to be displayed to the user.

For ODF 1.0/1.1 documents, some special handling for not signed macro streams in a signed document is needed (see below), because the implementation was different than what is specified in ODF 1.2 now

Older versions of OOo with a document signature only check for not signed files when these are not located in the META-INF folder. With OOo 3.2, this check will be enhanced to also check for not signed content in the META-INF folder.

Include existing document signatures in newer document signatures

The document signature could include all existing signatures. This way it wouldn't be possible that someone removes old signatures, which where existent in the moment an other person signed the document. Use case: Different people/groups need to sign the document in a certain order. It's important to see who had signed the document before, and that older signatures can't be removed afterwords without breaking the signature.

Encryption of Digital Signatures in Encrypted Documents

Currently OOo doesn't encrypt digital signatures in encrypted documents. This doesn't make the signatures less reliable, and there are arguments for encrypting the signatures as well as arguments for not doing so.

No encrypting the signatures can be a privacy issue. because someone could see who has signed a document. But on the other side, some automatic processes can't verify the signature when it's encrypted.

For now, we don't have any plans to change the current implementation. In the future, additional signature implementations might handle it differently.

The signatures are part of the package, contained in META-INF, and dont need to be listed in manifest.xml, so encryption is possible even when signing manifest.xml (w/o storing information for the encryption in the manifest.xml, but in the signature file itself).

Show a warning for not signed macro streams in documents signed with OpenOffice.org 2.x

The first implementations of digital signatures in OpenOffice.org 2.x completely separated signing document content from signing scripting content. Macros have not been included in the document signatures, so they could be manipulated in a signed document. This behavior changed in OpenOffice.org 3, where the macros are now also signed when the document is signed (without having the same status like explicitly signed macros).

For compatibility reasons, OOo should show a warning (and not a broken signature) when a signed document contains macros which are not signed, but only for documents created with OpenOffice.org 2.x.

The warning will not contain any extra explanation text that the issue can have happened for legacy reasons - signed ODF documents with macros are probably not used that much.

Inform the user that OOo's Digital Signatures have no legal value

While the digital signatures help for author authentication and for ensuring document integrity, the implementation in OOo is not a certified solution, so the signatures won't have any legal value in most countries.

OOo should inform the user about this when he starts to add a digital signature to a document. There should be a configuration item to disable the warning.

Digital Signatures Framework

There are some ideas for a digital signature framework, so other (3rd party) implementations can be used in OpenOffice.org. See the description in the Wiki and Jochen's presentation

- Digital Signatures for Extensions
  It would be good if OpenOffice.org extensions could be digitally signed, for publisher authentication as well as for integrity verification.

- Skip support for self signed certificates?
  Self signed certificates are not good for establishing trust, because the user would need to verify the certificate's fingerprint somehow.

It's very easy to fool people with signing a document with a self signed certificate, using the name and other data from another person. Most users don't have a deeper knowledge about signatures, certificates and PKI, and they simply look at the name listed in the certificate.

Certificate revocation

OOo needs to support certification revocation.

Currently revocation works on Windows, using MS Crypto API, but it doesn't work with our current implementation on other platforms based on Mozilla profiles and NSS.

5.1.4. Hardened Office Installation

The idea about a "Hardened Office Installation" came up in discussions with Eric Filiol & Jean-Paul Fizaine.

In a special installation/configuration, more security features and more security checks might be available.

This list include things like

- Digital Signatures also for application macros installed with OOo (currently digital signatures are only used for macros in documents)
- Do not install additional scripting engines, like python for example
- Lock certain user configuration, like macro security level
- Add hash values in all ODF documents, check on loading
- Digital Signatures also for application macros
- This is mainly to stop easy to achieve viral effects.

Somebody who can manipulate the macros in the office installation already has access to the system, and could manipulate anything, not only the OOo macros. The "primo infection" has already happened, and the system is compromised. But of course, manipulating macros is much easier than manipulating system binaries or configurations, and with OOo running on many different platforms, the malicious code also can be multi platform quite easily.

Security Enhancements

Become more robust against viral threats

We can't hinder people from executing unsafe stuff, but at least we can make it harder for that stuff to get deeper into OOo.

Macro location, trusted sources

OOo shouldn't have automatically trusted macro locations.

OOo only checks for macros in documents, but trusts any macro that is in the "application basic", delivered with OOo.

The reason for this is that OOo sees them as application code, like any shared library. Malicious code that is executed (with permission from the user) is able to manipulate macros as well as any other shared library (primo infection). But it's much easier to put some "virus auto start" code into macros, than in some binary shared library.

A solution could be to warn/hinder macro execution from any location, not only from documents. No path would be implicitly trusted, the default should be to execute only signed macros.

Signing all macros is difficult in an open source project, because everybody can build OpenOffice.org installation sets, but of course can't have "the" certificate for singing the macros. So administrators need some tooling to sign all macros with their own certificate and to configure

that certificate as trusted. Alternatively, if signing the macros can't be done for some reason, the administrator could also put all macros in some special folder and configure that folder to be trusted.

Additionally, the administrator can configure OOo in a way that users can't add trusted locations or trusted authors. For Sun provided OOo builds it might be possible in the future that we sign all basic libraries and other shared libraries, must be clarified. An interim solution could also be to warn before any macros execution and to allow administrators and users to configure the location from installed macros as a trusted source, but don't make that a default trusted source anymore.

Basic integrity checking for not signed documents

Most people don't sign documents, but it would be nice to have some confidence that the document was not manipulated.

Integrity checks for documents

There are already integrity checks with the zip file format which is used for ODF files, but this doesn't help if some user or some malicious code manipulates document content or macros, or adds /removes something to/from the zip archive.

A solution would be to have hash values for all content of the document (zip archive). But this would make manipulations only a little bit more difficult, because the calculation of the hash values would be publicly specified (on OOo or in ODF), and even the source code for this would be publicly available.

The hash value approach is only safe when the overall hash value would be encrypted, but then we already have digital signatures and people need a certificate containing some private key. Adding this feature without encryption only raises the barrier for document manipulation a little bit and people might (wrongly) feel a little bit more confident.

An other problem is that OOo in default configuration wouldn't warn if the hash value doesn't exist, because this is the case for all existing documents, and probably still for future documents created with other applications or by some automatic document creation tools. Comments

In case people think this feature is useful: Hash creation algorithms should be the same we already use for digital signatures If done each time when saving the document, calculation must be fast, so probably no canonization of XML documents, like currently done for digital signatures

I understand the benefits of canonization, but is it useful for XML content stored in ODF files? A normal office installation will only warn if a document contains hash values which don't match the current content, but not if no hash values are included Administrators or users could configure OOo to also warn if no hash values exist.

6.CODE INSPECTION

6.1. Code Review and Inspection - I

File name: code.js

Link to Source file :
Sourceroot/main/ooxml/source/framework/SchemaParser/src/org/apache/openoffice/ooxml/schema/generator/html/code.js

Reason to select the file: A variable declaration has been missed. If a variable is not declared as a local variable, it becomes a global variable by default, which may be unintentional and could lead to unexpected behavior and this can be causes to reliability and maintainability of the product.

Function: The OOXML framework targets the import of Office open XML documents and tries to do as much of this work automatically. By reading the OOXML schema files and producing a parser skeleton it removes the necessity of managing XML contexts manually.

The design of the new OOXML import framework adds a layer of abstraction on top of a push parser. The events for start tags, end tags or text are handled by the new import framework. Rules for this are derived directly from the specifications. These introduce the concept of complex types and simple types. Very simplified a complex type describes parent-child relationships between elements while simple types describe the types of attribute values. It is the task of the framework to do the translation from elements to complex types and preprocess attribute values according to their simple types. This has several advantages over a classical bare bones push parser:

- There are a few cases where the same element is mapped to different complex types. This disambiguation is now done automatically by the framework.
- The callbacks are more readable because they are directly tied to a complex type which can be looked up in the spec.
- Much of the low-level processing is now done automatically and therefore
  - does not obfuscate the import code
  - is less error prone
  - requires the developer to write less code
  - is potentially faster

The connection between OOXML parser and importer callbacks is done via a domain specific language (DSL). This, together with the automatic preprocessing of the specifications, allows the development of automatic analysis and processing programs. These allow us to

- analyze how much of the specification is handled by import callbacks and thus
- tell us which complex types and attributes still need more work
- compile documentation contained in the document code
- track progress of the development

- improve the development process by providing means to e.g. search for the implementation of a certain element or complex type
- add logging and debugging functionality on demand

Language: JavaScript

Severity: Warning

Tags: reliability and maintainability

In JavaScript, if a variable is used in a function but not declared as a local variable, it becomes a global variable by default. This can have unintended consequences: unlike local variables, global variables can be read and modified by all functions. If different functions use the same global variable, they may end up overwriting each others values, leading to subtle and difficult to diagnose bugs.

Recommendation

Check whether the variable in question was meant to be local; if so, declare it by means of a var declaration. If the variable is really meant to be global, it is best to document this fact by inserting a global var declaration at the beginning of the source file.

```
Source root/main/.../html/code.js                                                    10 alerts
↑   1-12
13  function LoadData ()
14  {
15      gTypeNames = [];

    Variable gTypeNames is used like a local variable, but is missing a declaration.

16      ComplexTypeNames = [];

    Variable ComplexTypeNames is used like a local variable, but is missing a declaration.

17      SimpleTypeNames = [];

    Variable SimpleTypeNames is used like a local variable, but is missing a declaration.

18      for (var sName in Data)
19          {
```

Figure 6. 1: missing variable declaration error - I

```
74
75  function GetMatches (sPattern)
76  {
77      var aLcPatterns = sPattern.toLowerCase().split(/\s+/);
78
79      var nTypeCount = gTypeNames.length;
80
81      var aMatches = [];
82      for (index=0; index<nTypeCount; ++index)

    Variable index is used like a local variable, but is missing a declaration.

83          {
84          var sTypeName = gTypeNames[index];
85          var aParts = new Array(sTypeName);
86          var sLcTypeName = sTypeName.toLowerCase();
87          var bIsMatch = true;
88          var nSearchStart = 0;
89          for (nPatternIndex=0; nPatternIndex<aLcPatterns.length && bIsMatch; ++nPatternIndex)

    Variable nPatternIndex is used like a local variable, but is missing a declaration.
```
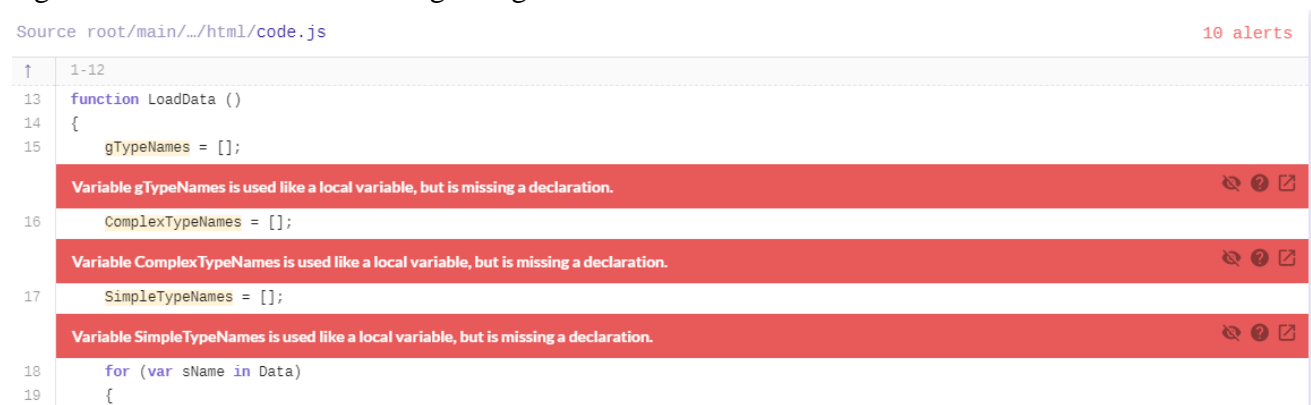
Figure 6. 2: missing variable declaration error - II

Example

In the following example, both f and g use a loop counter variable i. Since neither of them declares i to be a local variable, they end up accessing the same global variable, so every time f invokes g inside the loop, g overwrites f's value for i.

```
function f(a) {

    var sum = 0;

    for (i=0; i<a.length; ++i)

        sum += g(a[i]);

    return sum;

}

function g(b) {

    var prod = 1;

    for (i=0; i<b.length; ++i)

        prod *= b[i];

    return prod;

}
```

The example should be fixed by declaring i to be a local variable in f and g.

6.2. Code Review and Inspection - II

File name : example.html

Link to source file : Source root/main/bean/test/applet/oooapplet/example.html

Reason to select the file : The file include a Cross-site scripting vulnerability due to user-provided value.

Language: JavaScript

Severity: Recommendation

Tags: maintainability external/cwe/cwe-676

DOM functions that act like 'eval' and execute strings as code are dangerous and impede program analysis and understanding. Consequently, they should not be used. Several DOM functions allow evaluating strings as code without using eval explicitly. They should be avoided for the same reason as eval itself.

Recommendation

When calling setTimeout or setInterval, do not pass it a string to evaluate but a function.

Instead of using document.write to insert raw HTML into the DOM, use a framework such as jQuery.

Figure 6. 3: Cross-site scripting vulnerability

Example

In the following example, setTimeout is used to register a callback. The code to execute once the timeout expires is given as a string; this is bad practice.

setTimeout("notifyUser();", 1000);

Instead, directly pass the function to be invoked to setTimeout like this:

setTimeout(notifyUser, 1000);

CWE details for the vulnerability

CWE-676: Use of Potentially Dangerous Function

Description

The program invokes a potentially dangerous function that could introduce a vulnerability if it is used incorrectly, but the function can also be used safely.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOf and give insight to similar items

that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that the user may want to explore.

Table 6 1: Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type ID	Name
ChildOf	Class 1177	Use of Prohibited Code
ParentOf	Variant 785	Use of Path Manipulation Function without Maximum-sized Buffer

Table 6 2: Relevant to the view "Software Development" (CWE-699)

Nature	Type ID	Name
MemberOf	Category 1228	API / Function Errors

Modes of Introduction

The different Modes of Introduction provide information about how and when this weakness may be introduced. The Phase identifies a point in the life cycle at which introduction may occur, while the Note provides a typical scenario related to introduction during the given phase.

Phase

Architecture and Design

Implementation

Likelihood Of Exploit

High

6.3. Code Review and Inspection - III
File name : pythonloader.py

Link to source file : Source root/main/pyuno/source/loader/pythonloader.py

Reason to select the file : the file include a bug which makes the code more difficult to understand and may slow down loading of modules.

Language: Python

Severity: Warning

Tags: maintainability ,useless-code, external/cwe/cwe-561

Recommendation

Deleting the unreachable code will make the code clearer and preserve the meaning of the code. However, it is possible that the original intention was that the code should execute and that it is unreachable signifies some other error.

Figure 6. 4: Warning - Code is unreachable

Example

In this example the assignment to remainder is never reached because there is a return statement on the previous line.

```
import math
def my_div(x, y):

    return math.floor(x / y)

    remainder = x - math.floor(x / y)
* y
```

CWE-561: Dead Code – Code is Unreachable

Description

The software contains dead code, which can never be executed.

Extended Description

Dead code is source code that can never be executed in a running program. The surrounding code makes it impossible for a section of code to ever be executed.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOf and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and Can also be are defined to show similar weaknesses that the user may want to explore.

Table 6 3: Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name
ChildOf Class	Base	571	Irrelevant Code
CanFollow	Base	570	Expression is Always False
CanFollow	Base	571	Expression is Always False

Table 6 4: Relevant to the view "Software Development" (CWE-699)

Nature	Type	ID	Name
MemberOf	Category	1006	Bad Coding Practices

Modes of Introduction

The different Modes of Introduction provide information about how and when this weakness may be introduced. The Phase identifies a point in the life cycle at which introduction may occur, while the Note provides a typical scenario related to introduction during the given phase.

Phase

Implementation

Applicable Platforms

The listings below show possible areas for which the given weakness could appear. These may be for specific named Languages, Operating Systems, Architectures, Paradigms, Technologies, or a class of such platforms. The platform is listed along with how frequently the given weakness appears for that instance**.**

Languages

Class: Language-Independent (Undetermined Prevalence)

Common Consequences

The table below specifies different individual consequences associated with the weakness. The Scope identifies the application security area that is violated, while the Impact describes the negative technical **impact that arises if an adversary succeeds in exploiting this weakness. The Likelihood provides information about how likely the specific consequence is expected to be seen relative to the other consequences in the list. For example, there may be high likelihood that a weakness will be exploited to achieve a certain impact, but a low likelihood that it will be exploited to achieve a different impact.**

6.4. Code Review and Inspection - IV

File name: pythonloader.py

Link to source code : <u>Source root/**main**/**pyuno**/**source**/**loader**/**pythonloader.py**</u>

Language: Python

Severity: Recommendation

Tags: efficiency maintainability

Testing whether an object is 'None' using the == operator is inefficient and potentially incorrect.

When you compare an object to None, use is rather than ==. None is a singleton object, comparing using == invokes the __eq__ method on the object in question, which may be slower than identity comparison. Comparing to None using the is operator is also easier for other programmers to read.

Figure 6. 5: Using the == operator is inefficient error

Recommendation

Replace == with is.

Example

The filter2 function is likely to be more efficient than the filter1 function because it uses an identity comparison.

```
def filter1(function, iterable=None)

    if iterable == None:    # Comparison using '__eq__'

        return [item for item in iterable if item]

    else:

        return [item for item in iterable if function(item)]

def filter2(function, iterable=None)

    if iterable is None:    # Comparison using identity

        return [item for item in iterable if item]

else        return [item for item in iterable if function(item)]
```

6.5. Code Review and Inspection - V

File name: pythonscript.py

Link to source file: Source root/main/scripting/source/pyprov/pythonscript.py

Language: Python

Severity: Warning

Tags: maintainability, useless-code, external/cwe/cwe-563

Assignment to a variable occurs multiple times without any intermediate use of that variable. Multiple assignments to a single variable without an intervening usage makes the first assignment redundant. Its value is lost.

Recommendation

Ensure that the second assignment is in fact correct. Then delete the first assignment (taking care not to delete right hand side if it has side effects).

```
972         return self.dirBrowseNode.getChildNodes()
973
974     def hasChildNodes( self ):
975         return self.dirBrowseNode.hasChildNodes()
976
977     def getType( self ):
978         return self.dirBrowseNode.getType()
979
980     def getScript( self, uri ):
```

This assignment to 'getScript' is unnecessary as it is redefined here before this value is used.

```
981         log.debug( "DirBrowseNode getScript " + uri + " invoked" )
982
983         raise IllegalArgumentException( "DirBrowseNode couldn't instantiate script " + uri , self , 0 )
984
985     def getScript( self, scriptUri ):
986         try:
987             log.debug( "getScript " + scriptUri + " invoked")
988
989             storageUri = self.provCtx.getStorageUrlFromPersistentUrl(
990                 self.provCtx.uriHelper.getStorageURI(scriptUri) );
991             log.debug( "getScript: storageUri = " + storageUri)
992             fileUri = storageUri[0:storageUri.find( "$" )]
993             funcName = storageUri[storageUri.find( "$" )+1:len(storageUri)]
```

Figure 6. 6: variable occurs multiple times without any intermediate

6.6. Code Review and Inspection - VI

File name: mailmerge.py

Link to source file : Source root/main/scripting/source/pyprov/mailmerge.py

Language: Python

Severity: Recommendation

Tags: reliability readability convention external/cwe/cwe-396

Handling 'BaseException' means that system exits and keyboard interrupts may be mis-handled. All exception classes in Python derive from BaseException. BaseException has three important subclasses, Exception from which all errors and normal exceptions derive, KeyboardInterrupt which is raised when the user interrupts the program from the keyboard and SystemExit which is raised by the sys.exit() function to terminate the program.

Since KeyboardInterrupt and SystemExit are special they should not be grouped together with other Exception classes.

Catching BaseException, rather than its subclasses may prevent proper handling of KeyboardInterrupt or SystemExit. It is easy to catch BaseException accidentally as it is caught implicitly by an empty except: statement.

Recommendation

Handle Exception, KeyboardInterrupt and SystemExit separately. Do not use the plain except: form.

```
175         if dbg:
176             out.write("PyMailSMTPService flavors len %d\n" % len(flavors))
177
178         #Use first flavor that's sane for an email body
179         for flavor in flavors:
180             if flavor.MimeType.find('text/html') != -1 or flavor.MimeType.find('text/plain') != -1:
181                 if dbg:
182                     out.write("PyMailSMTPService mimetype is %s\n" % flavor.MimeType)
183                 textbody = content.getTransferData(flavor)
184                 try:
185                     textbody = textbody.value
186                 except:
```

Except block directly handles BaseException.

```
187                     pass
188                 textbody = textbody.encode('utf-8')
189
190                 if len(textbody):
191                     mimeEncoding = re.sub("charset=.*", "charset=UTF-8", flavor.MimeType)
192                     if mimeEncoding.find('charset=UTF-8') == -1:
193                         mimeEncoding = mimeEncoding + "; charset=UTF-8"
194                     textmsg['Content-Type'] = mimeEncoding
195                     textmsg['MIME-Version'] = '1.0'
196                     textmsg.set_payload(textbody)
```

Activate Windows
Go to Settings to activate W

Figure 6. 7: keyboard interrupts mis-handled

Example

In these examples, a function application.main() is called that might raise SystemExit. In the first two functions, BaseException is caught, but this will discard KeyboardInterrupt. In the third function, call_main_program_fixed only SystemExit is caught, leaving KeyboardInterrupt to propagate.

In these examples KeyboardInterrupt is accidentally ignored.

```
def call_main_program_implicit_handle_base_exception():
  try:
    #application.main calls sys.exit() when done.
    application.main()
  except Exception as ex:
    log(ex)
  except:
    pass
def call_main_program_explicit_handle_base_exception():
  try:
    #application.main calls sys.exit() when done.
    application.main()
```

```python
    except Exception as ex:

        log(ex)

    except BaseException:

        pass

def call_main_program_fixed():

    try:

        #application.main calls sys.exit() when done.

        application.main()

    except Exception as ex:

        log(ex)

    except SystemExit:

        pass
```

Summary

Apache OpenOffice is a free and open source suite of productivity tools that assist companies, businesses, and professionals in managing their writing projects, streamlining their document management tasks, and collecting and manipulating valuable data efficiently and is the leading open-source office software suite for word processing,

Equipped with drawing and diagramming features, special effects, and animations to summarize in a nutshell they are easy to learn, flexible, productive and is free of charge for any purpose. The OpenOffice.org source project is based on an architecture that can provide comprehensive personal productivity to different UNIX-based systems and may be ported many other platforms as well.

Multiple vulnerabilities have been discovered in OpenOffice Successfully exploiting these vulnerabilities could allow for arbitrary code execution in the context of the affected application. Failed exploitation could result in a denial-of-service condition. After carrying out a series of trial and error for code inspection it is possible to arrive to the conclusion that depending on the privileges associated with the application, an attacker could install programs; view, change, or delete data; or create new accounts with full user rights. Failed exploitation could result in a denial-of-service condition.

REFERENCE

[1]"Why Apache OpenOffice", Openoffice.org, 2020. [Online]. Available: http://www.openoffice.org/why/index.html. [Accessed: 08- May- 2020].

[2]"What is Open Source Software and How Can You Use it For Business? - Small Business Trends", Small Business Trends, 2020. [Online]. Available: https://smallbiztrends.com/2017/01/what-is-open-source-software.html. [Accessed: 08- May- 2020].

[3]"Compliance Costs and the Apache License", Openoffice.org, 2020. [Online]. Available: https://www.openoffice.org/why/why_compliance.html. [Accessed: 08- May- 2020].

[4]"Product Reviews", Openoffice.org, 2020. [Online]. Available: http://www.openoffice.org/product/reviews.html. [Accessed: 08- May- 2020].

[5]"TALOS-2017-0300 || Cisco Talos Intelligence Group - Comprehensive Threat Intelligence", Talosintelligence.com, 2020. [Online]. Available: https://talosintelligence.com/reports/TALOS-2017-0300. [Accessed: 08- May- 2020].

[6]H. Unterbrink, "Vulnerability Spotlight: Apache OpenOffice Vulnerabilities", Blog.talosintelligence.com, 2020. [Online]. Available: https://blog.talosintelligence.com/2017/10/vulnerability-spotlight-apache.html. [Accessed: 08- May- 2020].

[7]"Framework/Article/General Architecture Of The Framework User Interface Implementation - Apache OpenOffice Wiki", Wiki.openoffice.org, 2020. [Online]. Available: https://wiki.openoffice.org/wiki/Framework/Article/General_Architecture_Of_The_Framework_User_Interface_Implementation. [Accessed: 08- May- 2020].

[8]"Learning about Threat Modeling", Medium, 2020. [Online]. Available: https://medium.com/@roberthurlbut/learning-about-threat-modeling-3f6811e7520c. [Accessed: 08- May- 2020].

[9]"Open Source Threat Modeling - Core Infrastructure Initiative", Core Infrastructure Initiative, 2020. [Online]. Available: https://www.coreinfrastructure.org/blogs/open-source-threat-modeling/. [Accessed: 08- May- 2020].

[10]"LGTM - Code Analysis Platform to Find and Prevent Vulnerabilities", Lgtm.com, 2020. [Online]. Available: https://lgtm.com/. [Accessed: 08- May- 2020].

[11]"OpenOffice.org for Developers", Openoffice.org, 2020. [Online]. Available: http://www.openoffice.org/development/indexNew.html. [Accessed: 08- May- 2020].

[12]"Architecture - Apache OpenOffice Wiki", Wiki.openoffice.org, 2020. [Online]. Available: https://wiki.openoffice.org/wiki/Documentation/DevGuide/Database/Architecture. [Accessed: 08- May- 2020].