Apache OpenOffice Software

Assignment

Secure Software Engineering

M.H. Abeysingha - IT16149090

I.L Basnayake - IT16064164

****

Bachelor of Science Hons. In Information Technology

(Specialization in Cyber Security)

**Table of Content**

1. Product Overview……………………………………………………………………………..3

2. Product Assets……………………………………………………………...............................5

3. Example Attacks………………………………………………………………………………7

4. Vulnerability History…………………………………………………………………………11

5. Architecture Overview………………………………………………………………………..18

6. Threat Model………………………………………………………………………………….28

7. Assets to Threat Model Tracing………………………………………………………………29

8. Code of Inspection……………………………………………………………………………30

9. Summary……………………………………………………………………............................46

**Appendix A**

**Google Drive Link:** [**Apache Open Office**](https://drive.google.com/open?id=17SIjhb6NQfugbiIRl_e0ZSD3mFrh-yVK)

**Video Link YouTube Link**[**: Apache Open Office**](https://youtu.be/MdGEv4hLA3c)

**Domain and Historical Analysis**

**1.** **Product Overview**

Apache OpenOffice is a free and open-source office productivity software suite offered by The Apache Software Foundation for word processing, spreadsheets, presentations, databases, graphics, and more. Apache OpenOffice is both a product and an open-source project. The existence has been since October 13th, 2000. This opensource has lost its initial developer participation. During March 2014 – March 2015 it had only sixteen developers; the main four were IBM employees, and IBM had ceased official participation by the release of 4.1.1. StarDivision, the initial author of the StarOffice suite of software, was founded in Germany within the mid-1980s. On June 1, 2011, OpenOffice.org, the project and merchandise, including ASCII text file, trademarks, domain names, and website, were donated to the Apache Software Foundation. Since its inception as an open source project. Our corporate contributors have included Oracle, RedFlag CH2000, and IBM.

The line control service com.sun.star.awt.UnoControlFixedLine is the behavior of simple lines in a dialog. Mostly the line control is used to visually subdivide a dialog. It may provide horizontal or vertical orientation which is determined by the Orientation property of the model as specified in com.sun.star.awt.UnoControlFixedLineModel. The label of a line control is set by the Label property. Generally when you feel comfortable having others view, and give feedback on, your work. No matter which stage you decide to open source your project, every project should include the documentation such as README, Contributing guidelines, Code of conduct. The components will help you communicate expectations, manage contributions, and protect everyone’s legal rights. Gradually increase your chances of having a positive experience.

Apache OpenOffice could be 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. The office productivity software suite is available in many languages and is compatible with all major operating systems, including Marcos, Microsoft Windows and Linux. It contains four main applications such as Writer, Calc, and Impress which are major to Microsoft Word, Excel, and PowerPoint respectively.

The whole office package is simple to use and also the tools that comprise it work seamlessly, consistently, and harmoniously with one another. Why should a user use Apache OpenOffice main reason it is the leading open-source office software suite for word processing, spreadsheets, presentations, graphics, databases and more. It 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 enables users to simply determine the appliance or tool used for creating a specific file. They’ll also open the file from anywhere and launch the appliance that created it instantly. Furthermore, Apache OpenOffice makes it easy for users to transfer data and data between its tools. The tools also share a typical spell checker that has options which they'll update and synchronize across all the tools. It’s a data processing application that helps users think about writing their ideas while having the chance to create their content and documents look great. Apache OpenOffice provides a spreadsheet application that has the potential to drag in data from various sources. The productivity suite offers an impressive tool which will be used for creating multimedia presentations. Equipped with drawing and diagramming features, tricks, and animations; this tool is helpful for producing impressive materials. With its drawing application, Apache OpenOffice makes the sketching of business processes and creation of diagrams a breeze. additionally, the suite delivers a direction system which will track personal files and enables users to access the information they have for preparing reports and other documents. Last but not least, Apache OpenOffice is provided with a tool used for creating and editing mathematical equations and formulas for text documents.

Great Software requires extraordinary individuals. Apache OpenOffice is that the after effect of quite twenty years top-notch programming building. Structured from the start as a solitary little bit of programming, Apache OpenOffice encompasses a consistency and a top quality that's world-class. Its open-source improvement model methods there aren't any insider facts. Easy to decide on, easy to put in, easy to find out - Apache OpenOffice is that the easy choice for an office software suite. OpenOffice is suitable for complete beginners, but if you've got used the other office software, that's ok too. OpenOffice will change use of what you already know - through familiar screens and menus - and also what you've got - by reading existing files with no retyping. Apache OpenOffice is free software. Which means you're liberated to download it, liberated to install it on as many PCs as you prefer, liberated to pass copies to as many folks as you prefer. You will use OpenOffice for any purpose without restriction: private, educational, public administration, commercial. Free, really free. Open organizations and individuals performing at all degrees of state (neighborhood/bureaucratic/local/national then on) discover Apache OpenOffice is their optimal programming arrangement. The combo of an adaptable word processing system, a ground-breaking spreadsheet, dynamic illustrations, database access and more meets all the normal needs of a commonly occupied office specialist. Not for Profit (NFP) organizations of all sizes are turning to Apache OpenOffice - software that meets their needs and shares their values. The flexible word processing system, powerful spreadsheet, dynamic graphics, database access and more meet all the necessities for an office software package. Free/Open-Source Software (F/OSS) proponents are turning to Apache OpenOffice both as end-users and as contributors. With its flexible word processing system, powerful spreadsheet, dynamic graphics, database access and more, OpenOffice is that the office suite of choice of immeasurable users. OpenOffice is released under the OSI approved Apache License terms, with over 750 contributors and a global community approaching 400,000 individuals.

When considering about the confidentiality of Apache open office, the information about the use of this website is collected using server access logs and a tracking cookie. The collected information consists of the IP address from which you access the website, the type of browser and operating system you use to access our site, the date and time you, access our site, the pages would visit, the addresses of pages from where you followed a link to the site. We do not collect personally identifiable information gathered data with any personally identifying information from other sources.

Different vulnerabilities are found in OpenOffice, which could give self-assertive code execution. OpenOffice is an open-source efficiency programming suite that contains an application, spreadsheet application, introduction application, drawing application, recipe proofreader, and an administration application. Effectively misusing these vulnerabilities could give subjective code execution inside the setting of the influenced application. Relying on the benefits identified with the apparatus, an aggressor could introduce programs; view, change, or erase information; or make new records with full client rights. Failed exploitation could lead to a denial-of-service condition. Apache OpenOffice is the result of over twenty years' software engineering which is designed from the start as a single piece of software.

**2.** **Product Assets**

Sharing in an exceedingly mixed application Environment (Sharing Files with Microsoft Office Users) Most migration difficulties occur to incompatibility issues. No two document production applications will always have 100% fidelity, not whether or not those two applications are just different versions of the identical program. So one of one among one in an exceedingly one amongst one in every of the essential tasks in a successful migration is to spot infidelity risks and thus reduce user frustration. Often the issues that occur aren't actually caused by the migration but are areas that have always been problematic but it had been easier for users to figure around than complaint. Migration support desks will inevitably find yourself fielding calls that don't have anything to try to with the migration but has been some extent of frustration for a few time. By default Apache OpenOffice stores its files in Open Document Format (ODF), ISO/IEC 26300. For details about ODF. Older versions of Microsoft Office files are stored during a proprietary format which isn't human-readable and not publicly documented. Although Apache OpenOffice can open Microsoft Office files. The reverse is *not always* true. Microsoft Office has various degrees of support for ODF, starting in Office 2007, but earlier versions: MSO '97, Office XP etc., are not able to open, read or write ODF documents. Each application has the ability to directly export to PDF (Portable Document Format). This industry-standard format for universal file compression and viewing is ideal for sending the file to anyone else to view using Acrobat Reader or any other PDF display program on any operating system. Cells copied from Excel and pasted special, linked and with RTF format, into Word are imported into Writer as a normal table. This means that when the document is imported, the link to the Excel file is lost. Similarly when a chart is copied from Excel and pasted special, linked into Word, and then the file imported into Writer, the chart is imported as an embedded OLE object but is not converted into a Calc chart. On saving the file in Word format and then opening the file in Word and trying to edit the chart by double-clicking, it displays an error message.

Apache OpenOffice 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, 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, Adabas D, 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. 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. 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. Database access layer which is a module contains code for accessing databases from applications. It also contains the 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. Forms module is 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. It is self-contained, because exporting UNO components. Next component is svx which is an additional implementations for integrating the form layer into the applications can be found in the SVX module, project graphics. svx/source/form and svx/source/fmcomp belong to Database Access are the directories .Database Drivers which is a main cponent and there are several database drivers available which enable you to access certain database types which OpenOffice.org does not have native support for. Drivers are not a part of the OpenOffice.org distributions it is hosted here. SDBC Driver of postgresql allows to use the postgresql database from OpenOffice.org without any other wrapper layer such as odbc. This SDBC driver is giving OpenOffice.org native access to SQLite databases. The driver uses the MDBTools to provide access which is only read to Microsoft Access files to Linux users of OpenOffice.org.

OpenOffice.org cannot run Microsoft Office macros. Although the macro language is very similar, the underlying objects are quite different. To set whether OpenOffice.org keeps attached macros (so that they are still available for use in Microsoft Office) for Word, Excel and PowerPoint files. Opening a Microsoft Office file infected with a macro virus is quite safe in OpenOffice.org. OpenOffice.org has the ability to record macros. It is better practice to use the application program interface (API). As OpenOffice.org has a different approach to data sources. On importing a Word document that is the main document for a mail merge, there is no connection with the data source. It is not just a matter of making that connection; each field needs to be reinserted. Text form fields import into OpenOffice.org as Input fields and so lose their type. For example, a text form field of type “number” or “date” will accept any text for input. In use a text form field in OpenOffice.org displays a dialog for entering the text. Exporting back to Microsoft Office, text form fields are changed into “Filling” fields. The controls are exported as active controls when there is a Microsoft Office equivalent. As Word does not have the same database functionality of OpenOffice.org and as the macros are not compatible, it would be very difficult to have one form that works in both environments using standard form controls.

Apache OpenOffice recommend the actions be taken: Upgrade to the latest version of OpenOffice immediately, after appropriate testing. Verify no unauthorized system modifications have occurred on system before applying patch. Apply the principle of Least Privilege to all systems and services. Remind users not to open emails, download attachments, or follow links provided by unknown or untrusted sources.

**3.** **Example Attacks**

A security flaw affecting OpenOffice and Apache OpenOffice has been fixed in one in all the 2 open-source office suites. The opposite still appears to be vulnerable. Before attempting to guess which app has yet to be patched, consider that Apache OpenOffice for years has struggled attract more contributors. And though the amount of individuals adding code to the project has grown since last checked, the project missed its recent January report back to the Apache Foundation. The upshot is: security holes aren't being patched, it seems. The issue, identified by security researcher Alex Inführ, is that there is the way to attain remote code execution by triggering a happening embedded in an ODT (OpenDocument Text) file. In a blog post on Friday, Inführ explains how he found the way to abuse the OpenDocument scripting framework by adding an onmouseover event to a link in an ODT file. The event, which fires when a user's mouse pointer moves over the link, can traverse local directories and execute a neighborhood Python script. After trying various approaches to take advantage of the vulnerability, Inführ found that he could rig the event to call a particular function within a Python file included with the Python interpreter that ships with LibreOffice. For the answer I looked into the Python parsing code a bit more exhaustive and discovered that it's not only possible to specify the function you wish to call inside a python script, but it's possible to pass parameters furthermore," he said. The exploit was tested on Windows, and will work on Linux, too. Inführ says he reported the bug on October 18 and it absolutely was fixed in LibreOffice by the tip of the month. RedHat assigned it CVE-2018-16858 in mid-November and gave Inführ a disclosure date of January 31, 2019. When he published on February 1, in conjunction with the LibreOffice fix notification, OpenOffice still had not been patched. Inführ says he reconfirmed that he could move with disclosure while OpenOffice 4.16 has yet to be fixed. His proof-of-concept exploit doesn't work with OpenOffice out-of-the-box because the software doesn't allow parameters to be passed within the same way because the unpatched version of LibreOffice did. However, he says that the trail traversal issue can still be abused to execute a neighborhood Python file and cause further mischief and damage. We're imagining specifically targeted netizens being tricked into opening a zipper file, unpacking an ODT and Python script, so the ODT document attempting to execute the Python script when the victim rolls their mouse over a link, as an example. The Register tried to succeed in two OpenOffice contributors to seek out what is going on. According to Inführ, OpenOffice users can mitigate the chance by removing or renaming the pythonscript.py enter the installation folder.

LibreOffice, an open-source just like Microsoft Office, has patched a bug that allowed attackers to execute commands of their choosing on vulnerable computers. An analogous flaw in Apache OpenOffice remains unfixed. Austrian researcher Alex Inführ publicly reported the vulnerability on Friday, shortly after it absolutely was fixed in LibreOffice. His disclosure included a proof-of-concept exploit that successfully executed commands on computers running what was then a totally patched version of LibreOffice. The sole interaction that was required was that the target user hovers over an invisible link with a mouse. On Wednesday, researcher John Lambert provided additional PoC samples. The chief vulnerability exploited could be a path traversal that allowed the attack code to maneuver out of its current directory and into one that contained a sample Python script that LibreOffice installed by default. That allowed Inführ to invoke the cmd command on the vulnerable computer. The researcher then exploited a separate weakness that allowed him to pass parameters of his option to the command. Inführ chose to open the computer’s calculator, but a malicious attacker could have picked more nefarious things. The researcher privately reported the vulnerability to LibreOffice developers, and that they fixed it in versions 6.1.4.2 and 6.0.7. The identical path-traversal vulnerability remained unpatched in Apache OpenOffice at the time this post went live. In fairness, Inführ’s PoC exploit didn’t work against Apache OpenOffice, because it absolutely was unable to pass malicious parameters. It’s not clear yet if there could be other ways to use the traversal flaw to execute malicious code. Neither Inführ nor Apache OpenOffice developers more matured emails seeking comment for this post. Attackers are exploiting code-execution vulnerabilities in Microsoft Office for quite a decade. Inführ’s work demonstrates that open source clones aren’t likely to be much less liable to determined hackers. Malware bytes has more about the vulnerabilities here.

OOo shouldn't have automatically trusted macro locations. OOo only checks for macros in documents but trusts any macro that's within the “application basic”, delivered with OOo. The reason for this can be that OOo sees them as application code, like every shared library. Malicious code that's executed (with permission from the user) is in a position to govern macros likewise as the other shared library (primo infection). But it's much easier to place some “virus auto-start” code into macros, than in some binary shared library. A solution may well 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 after all, 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 cannot be in hot water 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 very way that users can't add trusted locations or trusted authors. For Sun provided OOo builds it'd be possible within the future that we sign all basic libraries and other shared libraries, must be clarified. An interim solution could even be to warn before any macros execution and to permit administrators and users to configure the situation from installed macros as a trusted source, but don't make that a default trusted source anymore.

There are already integrity checks with the zip file format which is employed for ODF files, but this does not 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 possess hash values for all content of the document (zip archive). But this could make manipulations only a touch bit tougher because the calculation of the hash values would be publicly specified (on OOo or in ODF), and even the ASCII text file for this could be publicly available. The hash value approach is barely safe when the general hash value would be encrypted, then again we have already got digital signatures and folks need a certificate containing some private key. Adding this feature without encryption only raises the barrier for document manipulation a touch bit and folks might (wrongly) feel a touch bit more confident. Another problem is that OOo in default configuration wouldn't warn if the hash value doesn't exist, because this can be the case for all existing documents, and doubtless 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 identical we already use for digital signatures. If done when saving the document, the calculation must be fast, so probably no canonization of XML documents, like currently in hot water digital signatures. I understand the advantages of canonization, but is it really useful for XML content stored in ODF files? A standard office installation would only warn if a document contains hash values which don't match this content, but not if no hash values are included. Administrators or users could configure OOo to also warn if no hash values exist.

In computing terminology, a macro virus may be a virus that's written in a very macro language: an artificial language that's embedded inside a software application (e.g., word processors and spreadsheet applications). Some applications, like Microsoft Office, Excel, PowerPoint allow macro programs to be embedded in documents such the macros are run automatically when the document is opened, and this provides a definite mechanism by which malicious computer instructions can spread. Are often one reason it can be dangerous to open unexpected attachments in e-mails. Many antivirus programs can detect macro viruses; however, the macro virus' behavior can still be difficult to detect. A macro may be a series of commands and actions that help to automate some tasks - usually a quite short and straightforward program. However they're created, they have to be executed by some system that interprets the stored commands. Some macro systems are self-contained programs, but others are built into complex applications (for example word processors) to permit users to repeat sequences of commands easily, or to permit developers to tailor the applying to local needs. A large scale infection can be spread through email connections, removable media, systems and the Internet, and is famously hard to recognize. A typical path for a large scale infection to taint a PC is by supplanting ordinary macros with an infection. The full scale infection replaces customary orders with a similar name and runs when the order is chosen. These vindictive macros may begin naturally when a report is opened or shut, without the client's information. When a document containing a large scale infection is opened, the infection can contaminate the framework. When activated, it will start to implant itself in different records and formats. It might degenerate different pieces of the framework, contingent upon what assets a large scale in this application can get to. At the point when the tainted records are imparted to different clients and frameworks, the infection spreads. Full scale infections have been utilized as a strategy for introducing programming on a framework without the client's assent, as they can be utilized to download and introduce programming from the web using mechanized key-presses. Nonetheless, this is remarkable as it is generally not productive for the infection coder since the introduced programming is typically seen and uninstalled by the client. Since a full scale infection relies upon the application instead of the working framework, it can contaminate a PC running any working framework to which the focused on application has been ported. Specifically, since Microsoft Word is accessible on Macintosh PCs, word large scale infections can assault a few Macs notwithstanding Windows stages. A case of a large scale infection is the Melissa infection which showed up in March 1999. At the point when a client opens a Microsoft Word archive containing the Melissa infection, their PC gets contaminated. The infection at that point sends itself by email to the initial 50 individuals in the individual's location book. This caused the infection to duplicate at a quick rate. Not all full scale infections are distinguished by antivirus programming. Alert when opening email connections and different archives decline the opportunity of getting contaminated.

Apache.OpenOffice.Text.Document.Malicious.Macro.Execution

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 / LibreOffice

**Impact:** System Compromise: Remote attackers can gain control of vulnerable systems.

**Recommended Actions**

Currently we are unaware of any vendor supplied patch for this issue.

Please disable Macro in OpenOffice they are not needed.

**Disclosed**: 02/08/2017 **Created**: 05/30/2018

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. The module also works against LibreOffice.

**Development:** Source Code**,** History

**Module Options**

To display the available options, load the module within the Metasploit console and run the commands 'show options' or 'show advanced':

msf > use exploit/multi/misc/openoffice\_document\_macro

msf exploit(openoffice\_document\_macro) > show targets

...targets...

msf exploit(openoffice\_document\_macro) > set TARGET < target-id >

msf exploit(openoffice\_document\_macro) > show options

...show and set options...

msf exploit(openoffice\_document\_macro) > exploit

**4.** **Vulnerability History**

Different vulnerabilities have been found in OpenOffice, which could take into account self-assertive code execution. OpenOffice is an open-source profitability programming suite that contains a word processor, spreadsheet application, introduction application, drawing application, equation manager, and a database the executive’s application. Effectively abusing these vulnerabilities could take into account discretionary code execution with regards to the influenced application. Contingent upon the benefits related with the application, an assailant could introduce programs; view, change, or erase information; or make new records with full client rights. Bombed misuse could bring about a forswearing of-administration condition. Multiple vulnerabilities are discovered in Apache OpenOffice, which could give arbitrary code execution. A vulnerability within the OpenOffice Writer DOC file parser, specifically within the WW8Fonts Constructor, allows attackers to craft malicious documents that cause denial of service (memory corruption and application crash) potentially leading to arbitrary code execution. (CVE-2017-9806) A vulnerability in OpenOffice's PPT file parser, specifically in PPTStyleSheet, allows attackers to craft malicious documents that cause denial of service (memory corruption and application crash) potentially leading to arbitrary code execution. (CVE-2017-12607) A vulnerability in OpenOffice Writer DOC file parser, specifically in ImportOldFormatStyles, allows attackers to craft malicious documents that cause denial of service (memory corruption and application crash) potentially leading to arbitrary code execution. (CVE-2017-12608) Successfully exploiting these vulnerabilities could give arbitrary code execution within the context of the affected application. Counting on the privileges related to the appliance, an attacker could install programs; view, change, or delete data; or create new accounts with full user rights. Failed exploitation could lead to a denial-of-service condition.

Apache OpenOffice is a well-known, free and open-source option in contrast to other office suite items. Vulnerabilities in office suite programming, for example, word processors are extremely valuable to aggressors for customer side assaults. Assailants regularly send pernicious records joined to messages misusing such vulnerabilities to execute malevolent orders when the casualty is fooled into opening the document through some type of social building. OpenOffice isn't the only one with this sort of issue, comparative vulnerabilities have been found by Talos before in other word processor applications and libraries, for instance, LibreOffice or even in the front drivers in the Windows portion. We have observed numerous battles utilizing this assault vector for focused assaults. The as of late broke down assault against South Korean clients is a genuine model. The enemies utilized a weakness in the Hangul Word processor (HWP) to contaminate their casualties. This shows that it is so essential to stay up with the latest and not just the working framework. On the off chance that you are an OpenOffice client we firmly prescribe you introduce the fundamental updates at the earliest opportunity.

Discovered by Marcin's Icewall’ Noga of Cisco Talos. Talos is releasing details of new vulnerabilities discovered within the Apache OpenOffice application. The first vulnerability, TALOS\_2017\_0295 within OpenOffice writer, the second TALOS\_2017\_0300 in the Draw application, and the third TALOS\_2017\_0301 discovered in the writer application. All these vulnerabilities allow arbitrary code execution to be performed.

**1. TALOS\_2017\_0295 – Apache OpenOffice Remote Code Execution Vulnerability in Apache OpenOfficce DOC WW8Fonts Constructor (CVC-2017-9806)**

The vulnerability is in the WW8Fonts class of the OpenOffice word processor application. An attacker can build a malicious .doc (Microsoft Word Binary File Format) file with a specially crafted malicious font, if this font is parsed by the WW8Font::WW8Fonts class constructor it leads to an out of bound write vulnerability which leads to remote code execution. More technical details can be found in the Talos Vulnerability Report. And in the OpenOffice Advisory.

Talos Vulnerability Report

CVE Number – CVE-2017-9806

Summary: An exploitable out of bound write vulnerability exists in the WW8::WW8Fonts functionality of Apache OpenOffice 4.1.3. A specially crafted doc file can cause an out of bound write potentially resulting in arbitrary code execution. An attacker can send/provide a malicious doc file to trigger this vulnerability.

Tested versions – Apache OpenOffice 4.1.3

Product URL – http://www.openoffice.org/

CVSS v3 Score - CVSS:3.0/AV:N/AC:H/PR:N/UI:R/S:C/C:H/I:H/A:H

Details: 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’s investigate this vulnerability. After OpenOffice writer opens the malformed doc file we see the following state:

So we see that write access violation appeared in the WW8 Fonts:: WW8Fonts constructor. The definition of this function is located in file

/ storage/ao4.1.3/main/sw/source/filter/ww8/ww8scan.cxx:6571.

Checking the pTmp pointer value we see:

OpenOffice Advisory.

Description : A vulnerability in the OpenOffice Writer DOC file parser, and 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.

Severity: Medium

There are no known exploits of this vulnerability.

A proof-of-concept demonstration exists.

Thanks to Marcin 'Icewall' Noga of Cisco Talos for discovering this issue.

Vendor: The Apache Software Foundation

Versions Affected: All Apache OpenOffice versions 4.1.3 and older are affected. OpenOffice.org versions are also affected.

Mitigation: Install Apache OpenOffice 4.1.4 for the latest maintenance and cumulative security fixes. Use the Apache OpenOffice download page.

**2. TALOS-2017-0300-Apache OpenOffice PPT PPTStyleSheet Level Code Execution Vulnerability (CVE-2017-12607)**

An exploitable out of bound write vulnerability exists in the ‘PPTPPTStyleSheet: PPTPPTStyleSheet’ functionality of Apache OpenOffice. This component is part of the Draw application used to create slideshow presentations. An attacker can create a specifically crafted PPT file which exploits this vulnerability causing an out of bound write and resulting in arbitrary code execution locally on the victim’s machine in the same context of the current running user.

Talos Vulnerability Report

CVC NUMBER – CVE-2017-12607

An exploitable out of bound write vulnerability exists in the PPTStyleSheet:: PPTStyleSheet functionality of Apache OpenOffice. A specially crafted PPT file can cause an out of bound write resulting in arbitrary code execution. An attacker can send/provide a malicious PPT file to trigger this vulnerability.

Tested versions - Apache OpenOffice 4.1.3X64 Apache OpenOffice 4.1.3 x86

Product URLs – https://www.openoffice.org/

CVSSv3 Score – 8.3 – CVSS:3.0/AV:N/AC:H?PR:N/UI:R/S:C/C:H/I:H/A:H

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.

OpenOffice Advisory

CVE-2017-12607: Out-of-Bounds Write in Impress' PPT Filter

Fixed in Apache OpenOffice 4.1.4

Description

A vulnerability in OpenOffice's PPT file parser, and specifically in PPTStyleSheet, allows attackers to craft malicious documents that cause denial of service (memory corruption and application crash) potentially resulting in arbitrary code execution.

Severity: Medium

There are no known exploits of this vulnerability.

A proof-of-concept demonstration exists.

Thanks to Marcin 'Icewall' Noga of Cisco Talos for discovering this issue.

Vendor: The Apache Software Foundation

Versions Affected

All Apache OpenOffice versions 4.1.3 and older are affected.

OpenOffice.org versions are also affected.

Mitigation

Install Apache OpenOffice 4.1.4 for the latest maintenance and cumulative security fixes. Use the Apache OpenOffice download page.

**3. TALOS-2017-0301-Apache OpenOffice DOC ImportOldFormatStyles Code Execution Vulnerability (CVE-2017-12608)**

An exploitable out of bounds write vulnerability exists in the ‘WW8RStyle: ImportOldFormatStyle’ functionality of Apache OpenOffice 4.1.3, specifically within the write application, used for document creation. A specially crafted doc file will cause an out of bound write and result in arbitrary code execution locally on the victim’s machine in the same context of the current running user.

Talos Vulnerability Report

CVE NUMBER - CVE-2017-12608

An exploitable out of bounds write vulnerability exists in the ‘WW8RStyle::ImportOldFormatStyle’ functionality of Apache OpenOffice 4.1.3, specifically crafted doc file can cause a out of bounds write resulting in arbitrary code execution. An attacker can send/provide malicious doc file to trigger this vulnerability.

Tested versions - Apache OpenOffice 4.1.3

Product URLs - https://www.openOffice.org/

CVSSv3 Score - 8.3 – CVSS:3.0/AV:N/AC:H/PR:N/UI:R/S:C/C:H/I:H/A:H

CWE – CWE-787-Out-of-bounds Write

This vulnerability is present in the 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. A specially crafted PPT file can lead to an out of bound write and ultimately to remote code execution.

OpenOffice Advisory

CVE-2017-12608: Out-of-Bounds Write in Writer's ImportOldFormatStyles

Fixed in Apache OpenOffice 4.1.4

Description

A vulnerability in OpenOffice Writer DOC file parser, and specifically in ImportOldFormatStyles, allows attackers to craft malicious documents that cause denial of service (memory corruption and application crash) potentially resulting in arbitrary code execution.

Severity: Medium

There are no known exploits of this vulnerability.

A proof-of-concept demonstration exists.

Thanks to Marcin 'Icewall' Noga of Cisco Talos for discovering this issue.

Vendor: The Apache Software Foundation

Versions Affected

All Apache OpenOffice versions 4.1.3 and older are affected.

OpenOffice.org versions are also affected.

Mitigation

Install Apache OpenOffice 4.1.4 for the latest maintenance and cumulative security fixes. Use the Apache OpenOffice download page.

**4**. **CVE-2017-3157 - Local Information Disclosure Vulnerability**

By abusing the way Apache OpenOffice before 4.1.4 renders installed objects, an aggressor could make a record that permits perusing in a document from the client's file system. Data could be recovered by the aggressor by, e.g., utilizing concealed areas to store the data, fooling the client into sparing the record and persuading the client to send the report back to the assailant. The powerlessness is relieved by the requirement for the assailant to know the exact record way in the objective framework, and the need to fool the client into sparing the archive and sending it back.

Published : Feb 22 2017 12:00AM Updated : May 23 2017 04:27PM

Security – CVSS Version 3.x / CVSS Version 2.0

NIST : NVD

Vector: CVSS:3.0/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N

Weakness Enumeration - CWE-200 – Exposure of sensitive Information to an Unauthorized Actor

Severity: Medium

There are no known exploits of this vulnerability. A proof-of-concept demonstration exists. Ben Hayak was reported this issue.Vendor: The Apache Software Foundation. All Apache OpenOffice versions 4.1.3 and older are affected. OpenOffice.org versions are also affected.

Mitigation - Install Apache OpenOffice 4.1.4 for the latest maintenance and cumulative security fixes.

CVSS Scores & Vulnerability Types

CVSS Score 4.3

Confidentiality Impact Partial (There is considerable informational disclosure.)

Integrity Impact None (There is no impact to the integrity of the system)

Availability Impact None (There is no impact to the availability of the system.)

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) Obtain Information

CWE ID 200

**5**. **CVE-2018-10583**

An information disclosure vulnerability occurs when LibreOffice 6.0.3 and Apache OpenOffice Writer 4.1.5 automatically process and initiate an SMB connection embedded in a malicious file, as demonstrated by xlink:href=file://192.168.0.2/test.jpg within an office:document-content element in a .odt XML document.

CVSS Scores & Vulnerability Types

CVSS Score 5.0

Confidentiality Impact Partial (There is considerable informational disclosure.)

Integrity Impact None (There is no impact to the integrity of the system)

Availability Impact None (There is no impact to the availability of the system.)

Access Complexity Low (Specialized access conditions or extenuating circumstances do not exist. Very little knowledge or skill is required to exploit. )

Authentication Not required (Authentication is not required to exploit the vulnerability.)

Gained Access None

Vulnerability Type(s) Obtain Information

CWE ID 200

LibreOffice is an open source, community-developed office productivity suite. It includes key desktop applications, such as a word processor, a spreadsheet, a presentation manager, a formula editor, and a drawing program. LibreOffice replaces OpenOffice and provides a similar but enhanced and extended office suite.

Security Fix(es):

\* libreoffice: Use-after-free in sdstor/stgstrms.cxx:StgSmallStrm class allows for denial of service with crafted document (CVE-2018-10119)

\* libreoffice: Out of bounds write in filter/ww8/ww8toolbar.cxx:SwCTBWrapper class allows for denial of service with crafted document (CVE-2018-10120)

\* libreoffice: Information disclosure via SMB connection embedded in malicious file (CVE-2018-10583)

For more details about the security issue(s), including the impact, a CVSS score, and other related information, refer to the CVE page(s) listed in the References section.

Additional Changes:

For detailed information on changes in this release, see the Red Hat Enterprise Linux 7.6 Release Notes linked from the References section.

**Appendix B**

**Design Analysis**

**5.** **Architecture Overview**

The OpenOffice.org source project is based on an architecture that can provide comprehensive personal productivity to different UNIX-based systems and maybe 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 us to port 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 to deliver a fully object-oriented architecture. The following information will give just a rough overview of 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.

**Layered architecture**

The whole architecture is based on a layered approach. There are four defined layers where each covers a special area of the functionality.

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.

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.

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 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 StarOffice suite but it is the same for the OpenOffice.org suite:

System abstraction layer

The layered approach of the system architecture is one of the important facts to allow the easy porting of the technology to wide range of different system platforms. For this the architecture defines a virtual layer witch is called the System Abstraction Layer (SAL). All platform depended implementation take place below this layer or are part of some optional modules. In an ideal world an implementation of the SAL specific functionality and recompiling the upper layer module will allow you to run the applications. To provide the whole set of functionality the optional platform specific modules, like telephony support or speech recognition, have to be ported, too. To reduce the porting effort the set of functionality provide by the SAL is reduced to a minima set available on every platform. Also for some system the layer includes some implementations to emulate some functionality or behavior. For example on systems where no native multi-threading is supported, the layer can support so called “user land” threads.

At this time the implementation of the platform dependent and independent parts of the graphical library is linked into one dynamically loaded shared library. So there is no well-defined set of libraries which build up the SAL.

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.

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.

**Infrastructure layer**

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, type safe 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.

OpenOffice.org 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 multimedia 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.

OpenOffice.org 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.

**Framework layer**

OpenOffice.org 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.

Application layer

All applications like the wordprocessor application, spreadsheet application, presentation application, charting application, etc. build up this layer. All these applications are realized as shared libraries, which are loaded by the application framework at runtime. The framework provides the environment for all these applications and also provides the functionality for how these applications can interact.

**Encrypted documents shouldn't 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)

**Improving the Encryption Implementation in OOo**

For OOo, 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**

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. This is the general part of the latest draft of the proposal that we will send to the OASIS OpenDocument TC:

Encryption:

The encryption process takes place in the 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 start key is generated. The byte sequence representing the password in UTF-8 is used to generate a 20-byte SHA1 digest (see [RFC3174]). 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 (see [RFC2898]) 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 (see [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.

**Comply with the updated ODF 1.2 Encryption Specification in OOo**

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. (Issue #XXXXX)

**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.

**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 dialog should be removed, but don't allow empty passwords. (Issue #i103783)

**OpenOffice.org Single Sign-On API**

Users of a client application that can communicate with a variety of services on a network may need to enter several passwords during a single session to access different services. This situation can be further exacerbated if the client application also requires the user to enter a password each time a particular network service is accessed during a session. As most network users must authenticate to an OS at login time, it would make sense to access some of the required network services at this time as well. A solution to this problem is provided by the Single Sign-On (SSO) methodology, which is the ability to login in once and access several protected network services. The best known SSO is the Kerberos network authentication protocol (see rfc1510). Kerberos functionality is commonly accessed through the Generic Security Service Application Program Interface (GSS-API, see rfc2743). Central to GSS-API is the concept of a security context, which is the "state of trust" that is initiated when a client (also known as source or initiator) identifies itself to a network service (also known as target or acceptor). If mutual authentication is supported, then the service can also authenticate itself to the client. To establish a security context, security tokens are exchanged, processed, and verified between the client and the service. The client always initiates this exchange. Once established, a security context can be used to encrypt or decrypt subsequent client-service communications.

The OpenOffice.org SSO API is based on GSS-API. The SSO API supports the creation of security contexts on the client and the service side as well as the generation of the security tokens that are required for the exchange to complete the security context based authentication. The SSO API does not support the actual exchange of security tokens or the encryption or decryption of client-service communications in an established security context. OpenOffice.org implements SSO in two different ways to authenticate with an LDAP server for configuration purposes. The first is Kerberos based and the second is a simple non-standard "cached username/password" SSO. The latter is provided as a fallback to support scenarios where no Kerberos server is available.

**Implementing the OpenOffice.org SSO API**

Implementing the OpenOffice.org SSO API involves creating security context instances (see XSSOInitiatorContext and XSSOAcceptorContext below) and using these instances to create and process security tokens. All of the OpenOffice.org SSO interfaces are available from the ::com::sun::star::auth namespace. The major interfaces are shown in Illustration 7.22 and described below.

XSSOManagerFactory

Represents the starting point for interaction with the SSO API. This interface is responsible for providing XSSOManager (described below) instances based on the user's configured security mechanism e.g. "KERBEROS".

XSSOManager

This interface is responsible for the creation of unestablished security contexts for clients (XSSOInitiatorContext) and services (XSSOAcceptorContext). An XSSOManager instance "supports" a single security mechanism, that is, the context instances that are created by an XSSOManager instance only interact with a single security mechanism implementation.

XSSOInitiatorContext

This interface represents a client-side security context that is unestablished when it is created. A single method, init(), is provided so that you can create an initial client-side security token that can be delivered to the relevant service and for processing or validating returned service-side security tokens (if mutual authentication is supported). The expected sequence of events for this client-side security context is: The client calls init(), passes NULL as the parameter, receives an appropriate client-side security token in return. The client sends the security token to the relevant service. If the service successfully processes this token, the client is authenticated. If mutual authentication is not supported, the client-side authentication sequence is now complete. If mutual authentication is supported, the service sends a service-side security token to the client. The client calls init() a second time and passes the returned service-side security token as a parameter. If the token is successfully passed, the service is authenticated.

XSSOAcceptorContext

This interface represents a service-side security context that is not established when it is created. A single method, accept (), is provided and is responsible for processing an initial client-side security token. If mutual authentication is supported, the method also generates a service-side security token for the client. The expected sequence of events for this service-side security context is: The service receives the client-side security token. The service calls accept (), passes the client-side security token as a parameter, and if successful, the client is authenticated. If mutual authentication is not supported, the service-side authentication sequence is now complete. If mutual authentication is supported, accept () returns a non-zero length service-side security token. The service sends the service-side security token to the client to authenticate the service.

When you implement the SSO API, you may require access to user passwords, especially if you are relying on a preexisting underlying security mechanism. If you do not know how to gain such access, you can use the OpenOffice.org SSO password cache. This cache provides basic support for maintaining a list of user name or password entries. Individual entries have a default lifetime corresponding to a single user session, but can optionally exist for multiple sessions. Support is provided for adding, retrieving, and deleting cache entries. Only one entry per user name can exist in the cache at any time. If you add an entry for an existing user name, the new entry replaces the original entry. The SSO password cache is represented by a single interface, namely the XSSOPasswordCache interface, available in the ::com::sun::star::auth namespace.

**Consequent implementation of the Model-View-Controller (MVC) paradigm**

When software applications contain a mixture of data access code, business logic code, presentation code and control logic they appear to be increasingly difficult to maintain. With the increased size of an application the complexity gets fast out of control because of interdependencies between all of the components. Changes become increasingly difficult, reuse of classes are nearly impossible because of their heavy dependency on too many other classes. Changes or additions to the functionality of the application in terms of new views on data too often require re-implementation or doubling of existing business logic as well as data access code for adaption. This in turn leads to an again increased maintenance effort because changes have to be made at different code locations and needs to be hold in sync. In large software systems the spiral of tightly coupled code, poor re-usage of classes/functionality and copy-pasted code, often leads to extreme maintenance efforts and significant longer time to market cycles.

A Model-View-Controller ( MVC ) architectural design pattern offers here a way of decoupling data access, business logic, data presentation and user interaction by defining three categories of objects, each specialized on either data access and business logic (Model), the presentation of data (View) or the handling of interaction events (Controller). The separation of this three categories, known as MVC triad (which origins from Smalltalk80), allows for e.g. different views on the same, shared data model. Consequently implemented in a software system with strictly separated, specialized object categories the MVC like architectural pattern leads to increased re-usage opportunities for classes, better modularity, better scalability, less dependencies between classes and therefor better maintainability of the entire system. This can only apply when the MVC triad like pattern is transparent to the contributing developers and implemented according to an overarching strategy throughout the entire software system. The challenge in the scope of large software systems like OpenOffice.org is to analyze if and how the MVC paradigm is followed and how contributing developers can be lead to a unified understanding, handling and implementation of this architectural design pattern. It should be emphasized that MVC here is called a paradigm and should be seen more as an architectural strategy which aims for clear functional separation of classes. What flavor of the MVC paradigm to prefer, is left to imagination and creativity. Is it Presentation-Abstraction-Control (PAC) pattern, Interface-Model Control (ICM), or a subset or variation of PAC like Hierarchical Model-View-Controller pattern (HMVC). All of these seem to offer compelling advantages wrt. User experience especially startup time experience due to the possibility of separately multithread the model and the view.

**Service-Oriented Architecture (SOA)**

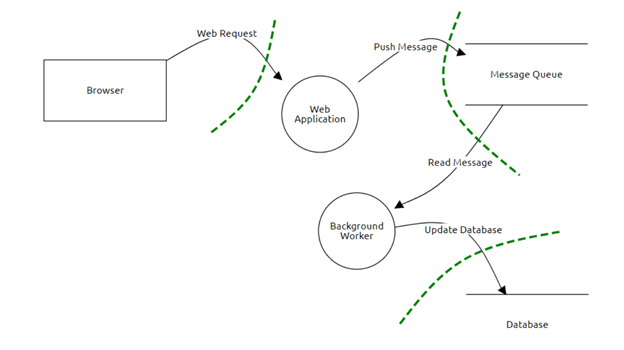
Service-orientation describes an architecture that uses loosely coupled independent services to provide resources that can be accessed without knowledge of their underlying platform implementation. The key is independent services with defined interfaces that can be called to perform their tasks in a standard way, without the service having foreknowledge of the calling application, and without the application having or needing knowledge of how the service actually performs its tasks. SOA is a style of information systems architecture that enables the creation of applications built by combining loosely coupled and interoperable services based on a formal interface definition that is independent of the underlying platform and programming language. The interface definition hides the implementation of the language-specific service. SOA-based systems can therefore be independent of development technologies and platforms.

**Service Component Architecture (SCA)**

Applications designed with Service Component Architecture (SCA) should have the following properties: Decoupling of service implementation and of service assembly from the details of infrastructure capabilities. Ability to work with multitude of programming languages Ability to work with various messaging constructs including One-Way, Asynchronous, Call-Return, and Notification. Infrastructure capabilities, such as Security, Transactions and the use of Reliable Messaging should be applied to code through metadata. Data should be represented in Service Data Objects. Components designed in SCA should be easily reused. Local calls to services should be more tightly coupled, reducing the overhead of creating and parsing messages intended for transport over a network.

**6.** **Threat Model**

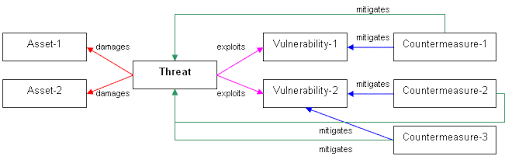
This Threat Model Diagram is an example of the connection from the user to the database. We will see where the possible threat security may occur and therefore the solution we will try and prevent it. A threat model may be a process that points out the potential threats towards an application. These are often structural vulnerabilities that require to be identified, enumerated, and prioritized. The threat modeling team asks questions on the component parts of the appliance and the interactions or data flows between them to guess how someone might attempt to attack it. Identify the threat is very important. Once some threats are identified the team designs ways to dam, avoid or minimize the threats. Some threats may have quite one mitigation. Some mitigations may be preventative and a few may be the detective. The team could favor to accept some low-risk threats without mitigations. Of course, some mitigations imply design changes, therefore the threat model diagram may need to be revisited.



**7.** **Assets to Threat Model Tracing**

The OpenOffice.org source project is based on an architecture that can provide comprehensive personal productivity to different UNIX-based systems and maybe 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 us to port 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 to deliver a fully object-oriented architecture. The following information will give just a rough overview of 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.

Sharing Files with Microsoft Office Users, Most migration difficulties occur to incompatibility issues. No two document production applications will always have 100% fidelity, not whether or not those two applications are just different versions of the identical program. So one of one among one in an exceedingly one amongst one in every of the essential tasks in a successful migration is to spot infidelity risks and thus reduce user frustration. Often the issues that occur aren't actually caused by the migration but are areas that have always been problematic but it had been easier for users to figure around than complaint. Migration support desks will inevitably find yourself fielding calls that don't have anything to try to with the migration but has been some extent of frustration for a few time.



**Appendix C**

**8.** **Code of Inspection**

**1. Testing equality to None - Pythonloader.py**



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

Query pack:[com.lgtm/python-queries](https://lgtm.com/search?q=pack%3Acom.lgtm%2Fpython-queries&t=rules)

Query ID: py/test-equals-none

Language: Python

Severity: Recommendation

Tags:[efficiency](https://lgtm.com/search?q=tag%3Aefficiency&t=rules)[maintainability](https://lgtm.com/search?q=tag%3Amaintainability&t=rules)

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.

## **Recommendation**

Replace == with is.

## **Example**

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

**2.** **Variable defined multiple times - pythonscript.py**

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)]

Assignment to a variable occurs multiple times without any intermediate use of that variable.



Query pack : com.lgtm / python-queries

Query ID : py/multiple-definition

Language : Python

Severity : Warning

Tags : maintainabilityuseless-codeexternal/cwe-563

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).

## **Example**

In this example, x is assigned the value of 42 but then the value is changed to 12 before x is used. This makes the first assignment useless.

x = 42

x = 12

print x

**2. Unnecessary pass – pythonscript.py**



Unnecessary 'pass' statement

Query pack:com.lgtm/python-queries

Query ID:py/unnecessary-pass

Language:Python

Severity:Warning

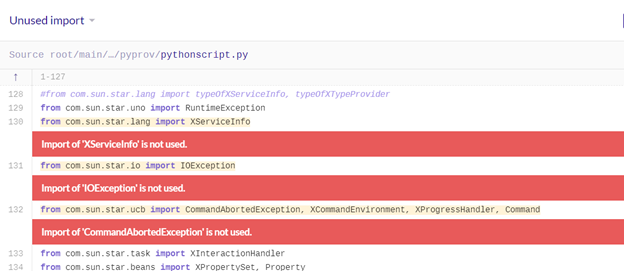
Tags:maintainabilityuseless-code

A pass statement is only necessary when it is the only statement in a block (the list of statements forming part of a compound statement). This is because the purpose of the pass statement is to allow empty blocks where they would otherwise be syntactically invalid. If the block already contains other statements then the pass statement is unnecessary.

## **Recommendation**

Remove the pass statement.

Unused import – pythonscript.py





Import is not required as it is not used

Query pack:com.lgtm/python-queries

Query ID:py/unused-import

Language:Python

Severity:Recommendation

Tags:maintainabilityuseless-code

A module is imported (using the import statement) but that module is never used. This creates a dependency that does not need to exist and makes the code more difficult to read.

## **Recommendation**

Delete the import statement.

**3. Expression has no effect – template.html**



An expression that has no effect and is used in a void context is most likely redundant and may indicate a bug.

Query pack:com.lgtm/javascript-queries

Query ID:js/useless-expression

Language:JavaScript

Severity:Warning

Tags:maintainabilitycorrectnessexternal/cwe/cwe-480external/cwe/cwe-561

An expression that has no effects (such as changing variable values or producing output) and occurs in a context where its value is ignored possibly indicates missing code or a latent bug.

## **Recommendation**

Carefully inspect the expression to ensure it is not a symptom of a bug. To document that the value of an expression is deliberately ignored, wrap it into a void expression.

## **Example**

The following code snippet accesses the selectedIndex property of a DOM node to trigger additional processing in certain versions of Safari. This, however, is not clear from the code itself, which looks like a property read whose value is discarded immediately.

elem.parentNode.selectedIndex;

To document the fact that the property read has a hidden side effect and its value is deliberately ignored, it should be wrapped into a void expression like this:

void(elem.parentNode.selectedIndex);

A common source of warnings are constructor functions that "declare" a property of the newly constructed object without initializing it, by simply referring to it in an expression statement like this:

function Graph(nodes, edges) {

this.nodes = nodes;

this.edges = edges;

// cache minimum distance between pairs of nodes

this.distance;

Semantically, this is unnecessary, since the property will be created upon first assignment. If the aim is to document the existence of the property, it would be better to explicitly assign it an initial value, which also serves to document its expected type:

function Graph(nodes, edges) {

this.nodes = nodes;

this.edges = edges;

// cache minimum distance between pairs of nodes

this.distance = {};

}

**4. Except block handles 'BaseException – mailmerge.py**

****

****

****

Handling 'BaseException' means that system exits and keyboard interrupts may be mis-handled.

Query pack:com.lgtm/python-queries

Query ID:py/catch-base-exception

Language:Python

Severity:Recommendation

Tags:reliabilityreadabilityconventionexternal/cwe/cwe-396

Displayed by default?Yes. Alerts for this query are visible by default, but can be hidden on a per-project basis. Learn how.

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.

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()

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

**05) selected file : pythonscript.py / external/cwe/cwe-563**

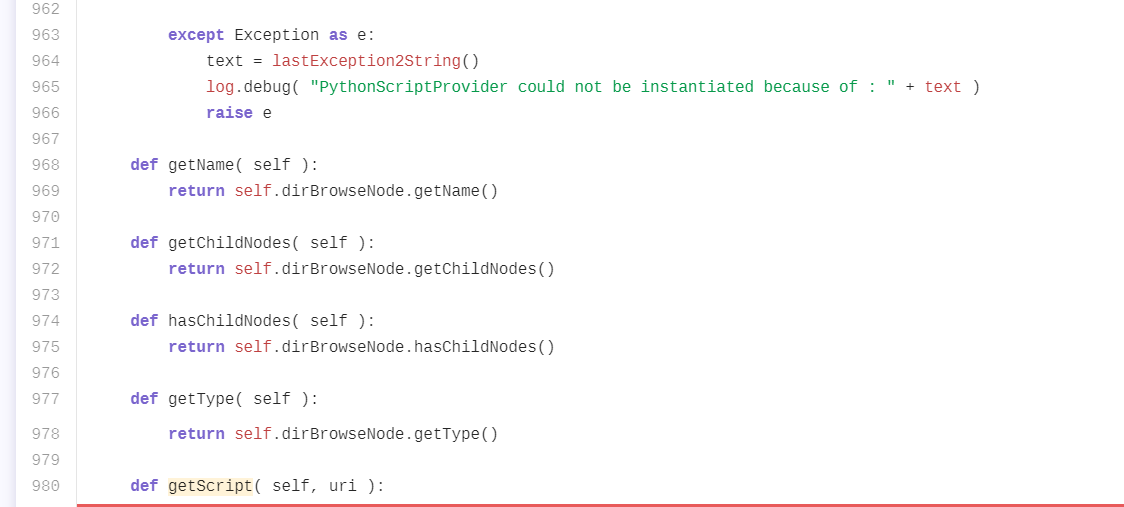
Query pack:com.lgtm/python-queries

Query ID:py/multiple-definition

why u select this file : According to my opinion the analysis done in Apache openoffice there are several weaknesses . This vulnerabilities are present in Apache OpenOffice in this mentioned category are Useless assignment to field , Useless assignment to local variable, Return statement assigns local variable,Variable defined multiple times,Unused local variable,Unused global variable, Assignment to exports variable,Redundant assignment,Duplicate property,Overwritten property,Conflicting function declarations, Useless assignment to global variable,Useless assignment to local variable,Conflicting variable initialization,Useless assignment to local variable,Unused local variable etc.

There are some imports which has not been used such as 'XServiceInfo' ,'IOException' 'CommandAbortedException' 'ROOT' etc.

This assignment is to ‘getScript’ is unnecessary as it is redefined before the value is used.

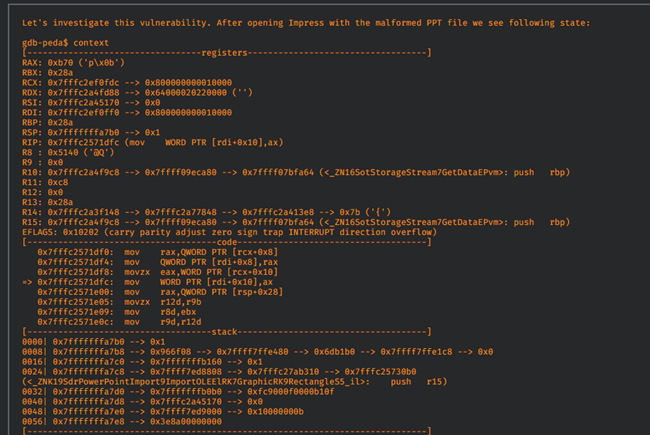
****

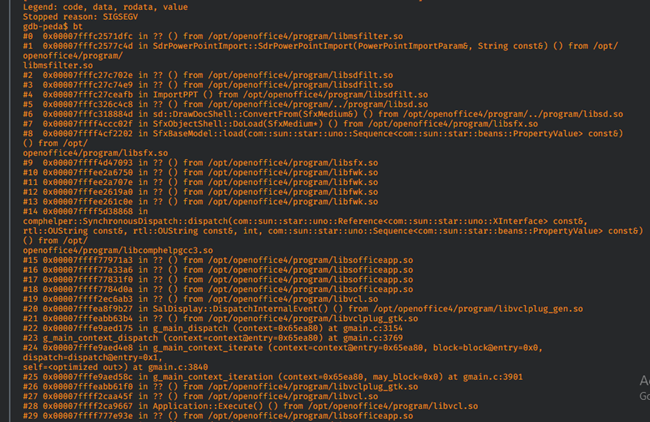
Assignment to a variable occurs multiple times without any intermediate use of that variable. Multiple assignments to a single variable without an intervening usage makes the first assignment redundant. Its value is lost.

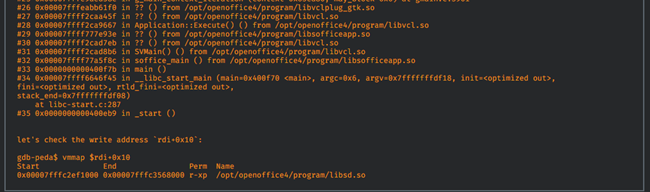
Recomendation: Ensure that the second assignment is in fact correct. Then delete the first assignment (taking care not to delete right hand side if it has side effects).

**6) (CVE-2017-12607)**

CVSS Score is 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 is Partial (There is reduced performance or interruptions in resource availability.) Access Complexity is Medium (The access conditions are somewhat specialized. Some preconditions must be satisfied to exploit) Authentication is Not required (Authentication is not required to exploit the vulnerability.) Gained Access None. Vulnerability Type(s) are Denial Of ServiceExecute CodeMemory corruption. CWE ID is 787.

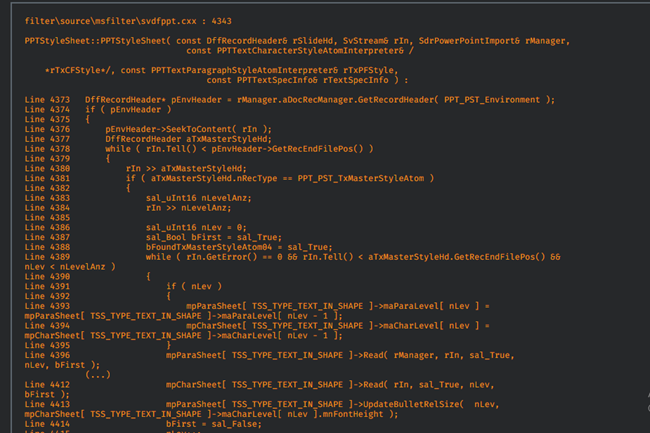
****

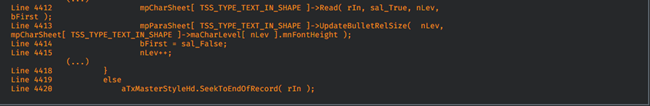
****

****

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.

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:

****

****

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.

****

Next, we see that nLevelAnz is read at line 4384. According to documentation:

****

but in our case its value is equal:

****

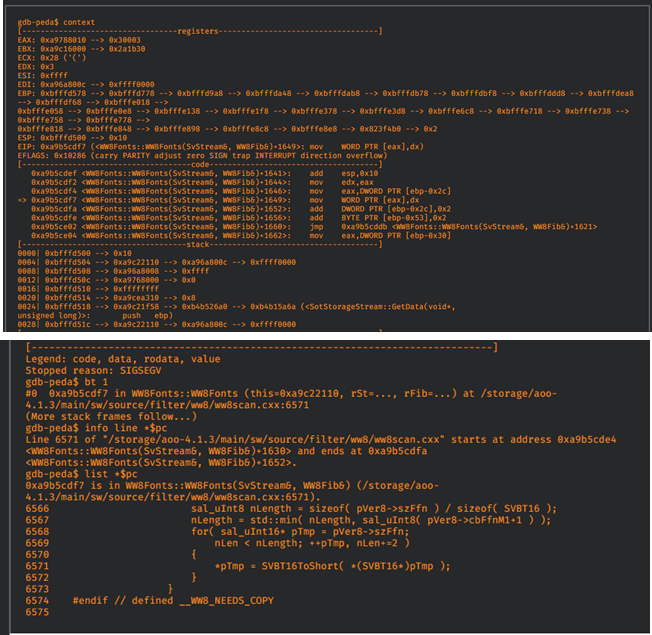
We also see the following:

****

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. 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.

CVE-2017-9806

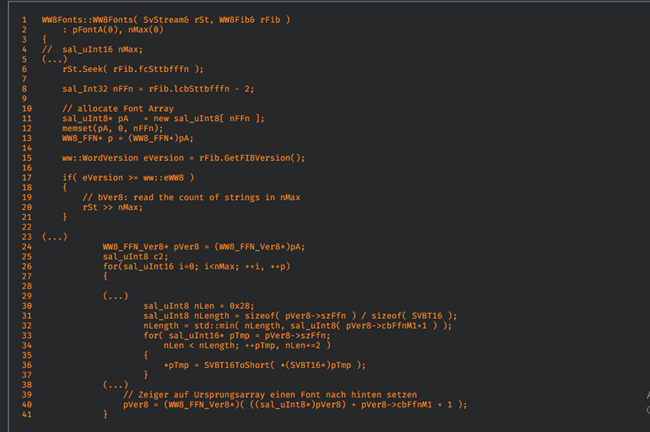
This weakness is occured to the availability of systems. CVSS score is 6.8 and confidentiality impact is partial and there is considerable information disclosure. So modification of the system file is possible. Even though the attacker does not have control over what can be modified is the integrity impact. If we consider the availability impact. There is reduced interruption in resource availability. And the access conditions are specialist must be satisfied to exploit. The denial of service and execution code memory questions are the vulnerability type. This vulnerability is present in Apache OpenOffice, a free open source office suite. A specially crafted file can lead to an out of bound write and ultimately to remote code execution.

****

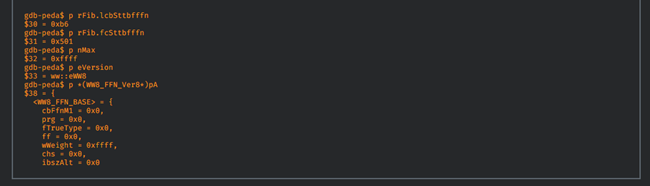
****

WW8Fonts::WW8 Fonts constructor. The definition of this function is located in file /storage/aoo-4.1.3/main/sw/source/filter/ww8/ww8scan.cxx:6571. 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. Looking at source code we see (the lines here have different numbers than the lines in the original source code)

****

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.

****

**9. Summary**

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 It is free, it can save you money that you would need to buy costly office productivity suites in the market. Apache OpenOffice is the leading open-source office software suite for word processing, spreadsheets, presentations, graphics, databases and more. It 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. The result: Apache OpenOffice does everything you want your office software to do, the way you want it to. Tong with a set of predefined table definitions for tracking Assets, Customers, Sales Orders, Invoices and much more. 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.Multiple vulnerabilities have been discovered in Apache OpenOffice, which could allow for arbitrary code execution. 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 “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 Rascagneres. Using .odt files is not common, the report showed, but if proven successful could lead to wider spread use in the future

Vedeo Link: <https://youtu.be/MdGEv4hLA3c>