



Strathmore
UNIVERSITY

Strathmore University
SU+ @ Strathmore
University Library

[Electronic Theses and Dissertations](#)

2017

Forensic analysis of office open XML spreadsheets

David Odhiambo Godiah
Faculty of Information Technology (FIT)
Strathmore University

Follow this and additional works at <http://su-plus.strathmore.edu/handle/11071/5614>

Recommended Citation

Godiah, D. O. (2017). *Forensic analysis of office open XML spreadsheets* (Thesis). Strathmore University. Retrieved from <http://su-plus.strathmore.edu/handle/11071/5614>

This Thesis - Open Access is brought to you for free and open access by DSpace @ Strathmore University. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of DSpace @ Strathmore University. For more information, please contact librarian@strathmore.edu

Forensic Analysis of Office Open XML Spreadsheets

Godiah David Odhiambo

Master of Science in Information Systems Security

2017

Forensic Analysis of Office Open XML Spreadsheets

Godiah David Odhiambo

**Submitted in partial fulfilment of the requirements for the degree of
Master of Science in Information Systems Security at Strathmore University**

**Faculty of Information Technology
Strathmore University
Nairobi, Kenya**

June, 2017

This dissertation is available for Library use on the understanding that it is copyright material and that no quotation from the dissertation may be published without proper acknowledgement.

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the dissertation contains no material previously published or written by another person except where due reference is made in the thesis itself.

© No part of this thesis may be reproduced without the permission of the author and Strathmore University.

Godiah David Odhiambo

Author

15/08/2017

Approval

The dissertation of Godiah David Odhiambo was reviewed and approved by the following:

Dr. Ondrej Rysavy,

Associate Professor, Department of Information Systems,

Brno University of Technology

Dr. Joseph Orero,

Dean, Faculty of Information Technology,

Strathmore University

Professor Ruth Kiraka,

Dean, School of Graduate Studies,

Strathmore University

Abstract

Digital Forensics is the science of acquiring, preserving, analysing and presenting digital evidence from computers, digital devices and networks in a manner that is admissible in a court of law to support an investigation. Microsoft Office, LibreOffice, OpenOffice, NeoOffice and Google documents spreadsheets and presentations are widely used to store and circulate data and information especially within organisations. They are often rich in information deeply embedded in them that can be retrieved by examining metadata or deleted material still present in the files.

OOXML is a standard developed by Microsoft and registered by ECMA (as ECMA-376), and approved by the ISO and IEC (as ISO/IEC 29500:2008) as an open standard for the development of Office documents, spreadsheets and presentations. Documents, spreadsheets and presentations created using this standard consist of zipped file containers, parts and relationships which upon extraction and analysis reveals forensically interesting information. Existing forensic tools have limitations as far as extracting and analysing OOXML spreadsheet metadata is concerned in that most of them can extract only limited and basic metadata.

The objective of this research is to carry out forensic analysis of metadata in OOXML spreadsheets by studying limitations of existing forensic tools in extracting and analysing metadata in OOXML spreadsheets and designing and developing a Proof of Concept (PoC) implementation of a forensic tool that supports automated forensic analysis of OOXML spreadsheets with improved visualization, efficiency and advanced reporting functionality. This research adopts a methodology to review OOXML spreadsheet metadata extraction and analysis capabilities of existing forensic tools using sample spreadsheet datasets, carry out system analysis, design and PoC implementation of a forensic tool. In addition, the research carries out manual, functional, and security tests; quality assurance; and validation of the developed Proof of Concept implementation. The developed tool is able to extract and analyse relevant metadata from OOXML spreadsheets and present results in a forensic report.

Keywords: digital forensics, XML, OOXML, metadata, spreadsheets

Table of Contents

Abstract	iii
List of Figures	vii
List of Tables	xi
List of Abbreviations	xii
Acknowledgements.....	xiii
1. Introduction	1
1.1. Background of the Research	1
1.2. Statement of the Problem	2
1.3. Research Objectives	2
1.4. Research Questions	3
1.5. Research Hypothesis	3
1.6. Scope and Limitations.....	3
1.7. Relevance of the Research	3
2. Literature Review	5
2.1. Digital Forensics and Digital Evidence.....	5
2.2. Metadata	5
2.3. Why is Metadata Useful.....	6
2.4. OOXML Standard	7
2.5. Microsoft Office	9
2.6. LibreOffice.....	10
2.7. Relevant Research	11
2.8. Existing Forensic Tools.....	13
2.9. Conclusions	18
3. Methodology.....	20
3.1. Testing of Existing Forensic Tools	21
3.2. System Analysis, Design and Architecture	21
3.3. System Implementation, Testing and Validation	22
4. Testing of Existing Forensic Tools.....	26
4.1. Controlled Datasets Used	26
4.2. Metadata Extraction Using Existing Tools	26

4.3.	Conclusions	35
5.	System Analysis, Design and Architecture	37
5.1.	System Analysis	37
5.2.	Functional Specifications	38
5.3.	Technical Specifications	38
5.4.	System Design and Architecture	39
6.	System Implementation, Testing and Validation	51
6.1.	Implementation.....	51
6.2.	Testing.....	62
6.3.	Validation	75
7.	Discussion of Results.....	78
7.1.	Limitations of Existing Forensic Tools.....	78
7.2.	Proof of Concept Implementation of Forensic Tool	79
7.3.	Testing and Validation	79
8.	Conclusions, Recommendations and Future Work	81
8.1.	Conclusions	81
8.2.	Recommendations	81
8.3.	Future Work	81
	References.....	83
	Appendix A Metadata Present in OOXML Spreadsheets.....	89
	Appendix B Minimum Parts and Relationships Requirements for a Workbook.....	91
	Appendix B1 Content Types for Relationship Parts, Workbook Part and Sheet Part	91
	Appendix B2 Package-Level Relationship to Workbook Part.....	91
	Appendix B3 Minimum Content for Workbook.....	91
	Appendix B4 Workbook-Level Relationship to a Single Sheet.....	92
	Appendix B5 Minimum Content for a Single Sheet Part.....	92
	Appendix C Testing of Existing Forensic Tools.....	93
	Appendix C1 Extracting Metadata from Workbook.xml Using Metagoofil 2.2	93
	Appendix C2 Extracting Metadata from Workbook.xml.rels Using Metagoofil 2.2.....	93
	Appendix C3 Extracting Metadata from Worksheet.xml Using Metagoofil 2.2	94
	Appendix C4 Extracting Metadata from Worksheet.xml.rels Using Metagoofil 2.2	94

Appendix D System Implementation, Testing and Validation	95
Appendix D1 Implementation of Forensic Report.....	95
Appendix D2 Manual Test Cases Template.....	97

List of Figures

- Figure 2.1: Internal Directory Structure of an OOXML Document
- Figure 2.2: Advanced OOXML Carver Metadata Extraction from Word Document
- Figure 2.3: FOCA Extracting Metadata in Word Document
- Figure 2.4: FTK Extracting Metadata in Word Document
- Figure 4.1: Metadata Extraction Results Using Advanced OOXML Carver
- Figure 4.2: Metadata Extraction Results Using FOCA
- Figure 4.3: Metadata Extraction Results Using FOCA
- Figure 4.4: Metadata Extraction Results Using OfficeDissector 1.0
- Figure 4.5: Metadata Extraction Results Using OfficeDissector 1.0
- Figure 4.6: Metadata Extraction Results using Python-OOXML 0.13
- Figure 4.7: Metadata Extraction Results using Libextractor
- Figure 4.8: Metadata Extraction Results using Libextractor
- Figure 4.9: Metagoofil 2.2 Extracting Metadata from Public Spreadsheets in Microsoft.com
- Figure 4.10: Metadata Extraction Results in docProps/app.xml file Using Metagoofil 2.2
- Figure 4.11: Metadata Extraction Results in docProps/core.xml file Using Metagoofil 2.2
- Figure 4.12: Metadata Extraction Results in /_rels/.rels directory Using Metagoofil 2.2
- Figure 4.13: Metadata Extraction Results Using read_open_xml.pl
- Figure 5.1: Use Case in Conventional Format
- Figure 5.2: Class Diagrams
- Figure 5.3: Class Diagrams
- Figure 5.4: About OOXML Spreadsheet Analyser Wireframe
- Figure 5.5: Extract Metadata Wireframe
- Figure 5.6: Analyse Metadata Wireframe

Figure 5.7: Forensic Report Wireframe

Figure 5.8: System Architecture

Figure 6.1: Metadata Extraction

Figure 6.2: Extracted Package Properties Metadata

Figure 6.3: Extracted Document Properties Metadata

Figure 6.4: Extracted Revision Header Properties Metadata

Figure 6.5: Extracted Users Metadata

Figure 6.6: Extracted Worksheet Metadata

Figure 6.7: Extracted Worksheet Comments

Figure 6.8: Extracted Sheet Revision History Metadata

Figure 6.9: Extracted Cell Revision History Metadata

Figure 6.10: Extracted Comment Revisions Metadata

Figure 6.11: Extracted Row and Column Revisions Metadata

Figure 6.12: Extracted Custom View Metadata

Figure 6.13: Metadata Analysis

Figure 6.14: Unique Identifier rId14

Figure 6.15: Unique Identifiers rId15 and rId16

Figure 6.16: Unique Identifiers rId23 and rId24

Figure 6.17: Forensic Report

Figure 6.18: Test Use Case Results by User

Figure 6.19: Test Use Case Results by User

Figure 6.20: Document Revision Metadata Extraction Testing

Figure 6.21: Sheet Name Revision Metadata Extraction Testing

Figure 6.22: Inserted Sheet Metadata Extraction Testing

Figure 6.23: Inserted Comment Metadata Extraction Testing

Figure 6.24: Inserted Cell Text Metadata Extraction Testing

Figure 6.25: Cell Text Revision Metadata Extraction Testing

Figure 6.26: Last Printing Date Metadata Extraction Testing

Figure 6.27: Analysed Metadata from Spreadsheet Created by LibreOffice Calc

Figure 6.28: Windows 10 Machine IP Address

Figure 6.29: Cyborg Hawk 1.1 IP Address

Figure 6.30: Application Accessed from Cyborg Hawk Virtual Machine

Figure 6.31: Web Application Vulnerabilities

Figure 6.32: Web Application Vulnerabilities after Resolution

Figure 6.33: Code Analysis Feature of Microsoft Visual Studio

Figure 6.34: Code Analysis Results

Figure B1-1: Content_Types.xml Representation

Figure B2-1: Workbook Part Package-Level Relationship

Figure B3-1: Minimum Content for Workbook Part

Figure B4-1: Workbook-Level Relationship to a Single Sheet Part

Figure B5-1: Minimum Content for a Single Sheet Part

Figure C1-1: Metadata Extraction Results in Workbook.xml

Figure C2-1: Metadata Extraction Results in Workbook.xml.rels

Figure C3-1: Metadata Extraction Results in Worksheet.xml

Figure C4-1: Metadata Extraction Results in Worksheet.xml.rels

Figure D1-1: Package Properties Part of Forensic Report

Figure D1-2: Document Properties Part of Forensic Report

Figure D1-3: Users Part of Forensic Report

Figure D1-4: Cell Revisions Part of Forensic Report

List of Tables

Table 2.1:	SpreadsheetML Components
Table 2.2:	Microsoft Excel File Name Extensions
Table 2.3:	LibreOffice ODF File Name Extensions
Table 5.1:	Technical Specifications
Table 5.2:	Functional Specifications
Table 5.3:	Browse OOXML File Use Case
Table 5.4:	Extract Metadata File Use Case
Table 5.5:	Analyse Metadata Use Case
Table 5.6:	Generate Forensic Report Use Case
Table 6.1:	Manual Test Plan
Table 6.2:	Manual Test Cases Template
Table A1-1:	OOXML Spreadsheet Metadata
Table A2-1:	SpreadsheetML Components
Table D2-1:	Manual Test Cases Template

List of Abbreviations

XML:	Extensible Markup Language
OOXML:	Office Open XML
NIST:	National Institute of Standards and Technology
ODF:	Open Document Format
ISO:	International Standards Organisation
IEC:	International Electrotechnical Commission
ECMA:	European Computer Manufacturers Association
DFRWS:	Digital Forensic Research Workshop
CFTT:	Computer Forensics Tool Testing
SDLC:	Software Development Life Cycle
HTML:	Hypertext Markup Language
ASP:	Active Server Pages
OS:	Operating System
IIS:	Internet Information Services
OWASP:	Open Web Application Security Project
PoC:	Proof of Concept
SDK:	Software Development Kit
URL:	Universal Resource Locator
HTTP:	Hyper Text Transfer Protocol

Acknowledgements

I would like to thank my supervisor, Dr. Ondrej Rysavy for the useful feedback throughout the course of this dissertation.

I would also like to thank my classmates especially Caroline Wanjira for their valuable input in discussions, criticism and exchange of ideas.

To the entire Strathmore University and Brno University fraternity, especially my lecturers who supported me in one way or another, thank you very much.

Last but not least, special thanks to my wife, Lydia Wambui for her endless and dedicated support and understanding in the course of this dissertation.

1. Introduction

1.1. Background of the Research

Many organisations use Microsoft Office, LibreOffice, OpenOffice, NeoOffice and Google documents, spreadsheets and presentations to store and circulate data and information. Spreadsheets are used to create small file based databases that store data and automate computations. Apart from the surface content of these files, a lot of information can be retrieved by examining metadata using forensic means. It is important for forensic investigators to be able to analyse such files with accuracy and ease in order to extract relevant information and evidence.

The most popular Office suites of today store their files in zipped XML-based containers. Microsoft Office 2007 and subsequent versions store files and related information in a file container format called OOXML, as opposed to previously proprietary binary formats. Other alternatives such as LibreOffice and OpenOffice store files as ODF by default. LibreOffice has read and write support for OOXML while OpenOffice only has read support for OOXML. Google Docs has functionality to import and export OOXML documents (Didriksen, 2014). OOXML files often store a lot of data that can be useful for forensic investigations. This research carried out forensic analysis of OOXML spreadsheets for Microsoft Excel and LibreOffice Calc that have the capability to read and write to OOXML.

OOXML is a standard registered by ECMA (as ECMA-376), and by the ISO and IEC (as ISO/IEC 29500:2008) for representing documents, spreadsheets, charts and presentations. It is an open standard that can be used to develop Office documents, spreadsheets and presentations, though it is today majorly used by Microsoft. Microsoft WordprocessingML is a set of conventions for representing an OOXML document of type Word processing (ECMA International, 2006b, p. 7). Word processing documents are partitioned into sections, paragraphs, and fragments (also called runs). A run is a sequence of characters with identical formatting with unique revision identifiers (RI) related to the editing session when the run was introduced into the document or the session when it was created or changed. This is independent of the activation of the change tracking feature of the application. Looking at RIs it is possible to interpret how a document was modified or composed from source documents. This helps to recreate the change history of a document if one only has access to the document file itself. Microsoft SpreadsheetML is a set of conventions for representing an OOXML document of type Spreadsheet (ECMA International, 2006b, p. 7). It

consists of three main components: ZIP package, parts and relationships. Metadata can be extracted from shared spreadsheets using functionality developed by SpreadsheetML SDK.

OOXML documents store XML data that can support forensic investigations since they contain unique identifiers that could be used for document tracking to for example uncover previously unknown social networks (Garfinkel, 2009). Many forensic tools exist that can extract metadata from these documents but have limited capability to analyse and present the information in appropriate format to forensic investigators. In addition, these forensic tools are limited in extracting metadata from spreadsheets and do not present a good analysis and visualization of evidence from these files.

1.2. Statement of the Problem

There is need to collect and analyse forensic evidence from spreadsheets as these files are widely used to store and manipulate data. Existing forensic tools have limited capability to extract and analyse metadata from OOXML spreadsheets. These existing forensic tools cannot forensically detect revisions made to shared OOXML spreadsheets by different users, including cell value revisions, thus modifications and revisions to cell values in a shared spreadsheet may go undetected. As much as some of these tools being able to extract metadata from OOXML documents, they are only able to extract limited and basic metadata from OOXML spreadsheets.

1.3. Research Objectives

General Objectives

The objective of this research is to carry out forensic analysis of metadata in OOXML spreadsheets by exploring previous work on OOXML, exploring the weaknesses of existing OOXML forensic tools and come up with an improved and automated tool that fully supports forensic analysis of OOXML spreadsheets.

Specific Objectives

The specific objectives of the research are the following:

1. Study the OOXML standard and identify metadata that can be used for forensic analysis of OOXML spreadsheets;
2. Study the weaknesses of existing forensic tools in analysing OOXML spreadsheets;

3. Design, develop and test an improved Proof of Concept (PoC) implementation of OOXML forensic tool which automates the process of extracting and analysing metadata in spreadsheets with improved visualization and efficiency; and
4. Validate the developed OOXML forensic tool.

1.4. Research Questions

The following are the research questions that this research tries to answer:

1. What is the metadata existing in OOXML spreadsheets that is useful for forensic analysis?
2. What are the strengths and weaknesses of available forensic solutions with respect to metadata extraction and analysis of OOXML spreadsheets?
3. What are the areas that the improved OOXML forensic tool is expected to address? and
4. Does the developed tool provide useful and comprehensive metadata extraction?

1.5. Research Hypothesis

The following are the hypothesis that this research seeks to address:

1. Existing forensic tools are limited in adequately extracting and analysing all metadata in OOXML spreadsheets;
2. Previous research on OOXML forensics has not dwelt much on OOXML spreadsheets; and
3. It is not possible to extract metadata from write protected OOXML spreadsheets.

1.6. Scope and Limitations

This research is limited to forensic analysis of OOXML based spreadsheets. It seeks to develop a Proof of Concept (PoC) implementation of OOXML forensic tool which can further be developed at a commercial scale. Most metadata can only be extracted on shared OOXML spreadsheet files thus the developed tool will work best with shared OOXML spreadsheets. The research also aims to establish if there are any limitations on metadata extraction and analysis on write protected OOXML spreadsheet files.

1.7. Relevance of the Research

Systems and information security is increasingly becoming a concern for organisations and individuals due to advancement in technology and increasing affordable accessibility of the internet. Software packages and data installed and used on devices are at risk from unauthorised

access and modification thus the need to enhance their security and continuously monitor and audit them. An overview of existing forensic tools has established that these tools are limited in extracting and analysing metadata in OOXML spreadsheets. Most of these tools were designed to extract and analyse metadata from Office documents, thus the limited capability when used with OOXML spreadsheets. Forensic investigators could benefit from the results of this research by using the tool to accurately analyse file access, modifications and change history of OOXML spreadsheets under investigation, thus be able to verify authenticity of files. Auditors could also use the tool to track unauthorised modifications to spreadsheets especially changes to formulae that could otherwise result in huge discrepancies of figures and potential financial losses to financial institutions using these OOXML spreadsheet files.

2. Literature Review

This chapter describes a general overview of digital forensics and evidence; metadata; the OOXML standard; related previous research on OOXML forensics; and capabilities of existing forensic tools to extract and analyse metadata in OOXML spreadsheets. A lot of research has been carried out on OOXML and its implications in digital forensics, though most of this research has dwelt largely on Office documents. There is very little research that has been done specifically on Office spreadsheets. Many tools have been developed that are capable of extracting metadata from OOXML documents and some of these tools are capable of extracting limited metadata from Office spreadsheets. This research takes a look at the OOXML standard and reviews some selected literature on OOXML forensics and existing forensic tools. The literature review also answers some of the research questions and hypothesis.

2.1. Digital Forensics and Digital Evidence

The report from the first DFRWS presented the following definition of *digital forensic science*:

“The use of scientifically derived and proven methods towards the collection, preservation, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operation” (Pallmer, 2001).

Eoghan (2011) describes forensic as “*a characteristic of evidence that satisfies its suitability for admission as fact and its ability to persuade based upon proof (or high statistical confidence)*”. In addition to digital forensics being used for evidence admissible in a court of law, it can also be applicable in investigating breach of corporate policies by employees. Digital Forensics has evolved over the years from computers to information systems and today includes metadata extraction and analysis of OOXML files.

2.2. Metadata

Metadata is a common term used in digital forensics. It is defined as “*structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource. Metadata is often called data about data or information about information*” (NISO, 2004). In digital forensics, metadata can play a big role in unearthing information about

data itself and help to reconstruct events that have occurred or test hypothesis of a case. Examples of file metadata in forensic investigations include author, origin, data created, date modified, etc.

The SpreadsheetML as part of OOXML standard answers the research question of what are the metadata existing in OOXML spreadsheets that are useful for forensic analysis, and reveals that a lot of metadata can be extracted from OOXML spreadsheets. Table A1 of Appendix A shows the kinds of metadata existing in OOXML spreadsheets (Microsoft, 2017d).

2.3. Why is Metadata Useful

Metadata has a lot of usefulness in forensic investigations. It can aid the detection of changes and revisions in OOXML spreadsheets. Metadata such as creator, creation and modification dates aid the reconstruction of events that took place in the entire lifetime of a spreadsheet.

Unauthorized access and manipulation of spreadsheets may go undetected if not keenly monitored. For example, spreadsheets can be programmed with formulae that automate data manipulation, and even the smallest change in these formulae by methods such as salami attack (slicing) can potentially cause huge damage since these modifications are visually difficult to detect. Salami slicing is defined a series of many minor actions performed in succession that together result in a larger impact that would be difficult or illegal to perform at one go (Barry, 2010). Kabay (2002) gives an example of the old “collect-the-roundoff” salami attack where a programmer modified the arithmetic routines of interest computations such that calculations are carried out to several decimal places beyond the customary 2 or 3 kept for financial records and the roundoffs of the decimal numbers can go up to the nearest whole number. If a programmer collects these differences of actual decimal numbers and the roundoffs in a separate account, a sizable fund can grow with no warning to the financial institution.

In January 1993, four executives of a rental-car franchise in Florida, USA were charged with defrauding at least 47,000 customers using a salami technique by modifying a computer billing program to add five extra gallons to the actual gas tank capacity of their vehicles (Kabay, 2002). These examples confirm that a salami attack on spreadsheets may go undetected at an individual user level but the damage caused may be huge if the discrepancies of the changes are summed up. Metadata about modifications in OOXML spreadsheets is therefore important in order to help detect illegal and potentially harmful modifications early enough and thus save institutions from massive losses.

2.4. OOXML Standard

Microsoft adopted an XML based file standard called Office Open XML (OOXML) in its 2007 release of Office Suite for storing and transporting office files as opposed to the binary format in earlier versions of its Office Suite. OOXML standard is defined by the document (ECMA International, 2006b) ECMA-376 Standard (Microsoft, 2011). This migration to XML format brought numerous benefits including:

- Compact files – Files can be compressed up to 75% thus saving storage space and enhancing overall efficiency (Microsoft, 2006);
- Improved damaged file recovery since not all files will be permanently damaged as there exists different parts related to each other. Therefore, some information could still be recovered;
- Better privacy and more control over personal information - Personal and business-sensitive information such as author names, comments, tracked changes, and file paths can be easily identified and extracted independent of the Office file;
- Better integration and interoperability of business data - Documents, worksheets, presentations, and forms can be saved in an XML file format that is freely available for anyone to use. Information within an Office file can be extracted and utilized by other business applications using a ZIP utility and XML editor;
- Backward compatibility - The 2007 and later Microsoft Office Suite is backward-compatible with these earlier versions: Microsoft Office 2000, Microsoft Office XP, and Microsoft Office 2003; and
- More enhanced security, since macros are not stored inside the content thus it is difficult to damage an entire document using compromised macros.

XML, which OOXML standard is based on is a text based file system used to present structured data and documents and is composed of instructions in the form of tags and markup. In XML, a document is defined by a sequence of elements start and end tags with one or more attributes which basically define the properties and values of the element. XML has the advantage that any text editor can read and modify the file without necessarily being the owner of the file (Castiglione, D'Alessio, De Santis, & Palmieri, 2012). The OOXML standard has undergone many changes since its adoption by ECMA and ISO/IEC, such as ISO/IEC 29500-1:2016, ISO/IEC 29500-

2:2012, ISO/IEC 29500-3:2015 and ISO/IEC 29500-4:2016. At the moment, only Microsoft fully supports this standard while LibreOffice provides read and write functionality to this standard.

OOXML standard is a structure of building blocks and relationships used for composing, packaging, distributing, and rendering document-centred content. These building blocks define a platform-independent framework for document formats that enable software applications to generate, exchange, and display documents reliably and consistently. This standard is based on the simple and parts-based compressed ZIP file format specification and consists of XML reference schemas and a ZIP container (ECMA International, 2006b). Each file is comprised of a collection of any number of parts and has the following components:

- ZIP Package. This contains elements that are shared across Office applications, e.g. document properties, style sheets, charts, hyperlinks, diagrams, and drawings as well as those that are specific to an application e.g. worksheets in spreadsheets and slides in presentations. Each Office created file is saved as a single file in a ZIP container, and remains as single file instance (ECMA International, 2006b).
- Parts. These are the logical components of an OOXML ZIP Package such as thumbnail, metadata, media, and relationships. Each of these parts can be extracted and edited individually and later reassembled back to the original file. Parts used to describe Office applications data are stored as XML and they conform to the XML reference schema that defines the associated Office feature or object. Image content types are stored as binary files (.png, .jpg, etc.) within the document package. The SpreadsheetML package contains a package-relationship item and a content-type item (ECMA International, 2006b). The package-relationship item contains implicit relationships with targets of the parts: *WorkbookPart*, *DigitalSignatureOriginPart*, *CoreFilePropertyPart*, *ThumbnailPart*, *CustomFilePropertyPart* and *ExtendedFilePropertyPart*.
- Relationships. This stores attribute information of the different components of the OOXML file (ECMA International, 2006b). For example, the contents of the Office file, attachments, embedded image and properties are all stored in different files but are linked via relationships with identifiers. When a document is opened, the relationships are read and various components and parts put together to form the whole document. There are two types of relationships in OOXML documents, internal and external (Muhamad, 2011). All

relationships, including the relations associated with the root package are represented as XML files. These XML files are stored inside a package and contain relationships information, for example, the default location for relationships is “/_rels/.rels”. Relationships are composed of four elements: an identifier (Id), an optional source (package or part), relationship type (URI style expression) and a target (URI to another part).

- Non-XML data that may be included within the container, including such parts as binary data representing images or OLE objects embedded in the document.

Figure 2.1 shows the internal structure of OOXML document showing ZIP package, parts and relationships.

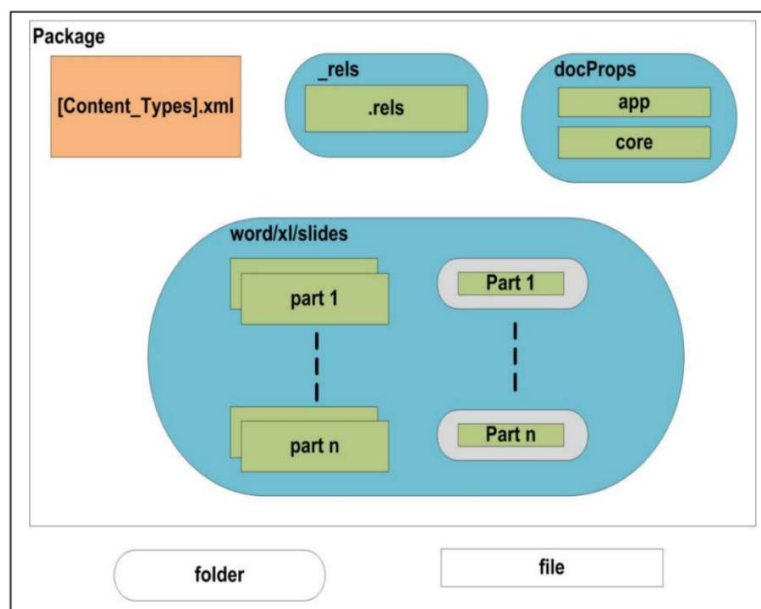


Figure 2.1: Internal Directory Structure of OOXML Document (Muhamad, 2011)

Figures B1-1, B2-1, B3-1, B4-1 and B5-1 in Appendix B shows the minimum requirements for a Workbook in terms of representation of parts and relationships (ECMA International, 2006b, p. 62).

Table A2 in Appendix A shows a summary of various components of SpreadsheetML of the OOXML standard (ECMA International, 2006b, p. 62).

2.5. Microsoft Office

Microsoft Office is a proprietary Office package developed by Microsoft and is the most widely used Office Suite worldwide today (Aghire, 2012). Office 2007 version and later uses an XML

based file system for Word, Excel and PowerPoint that fully supports OOXML standard. Files can be saved in formats supported by different versions of Office with the help of compatibility checkers and file converters which allow file sharing between different versions of Office.

Excel files saved in the XML format have file name extensions “.xlsx” or “.xlsm” by default where the latter contains macros. A file saved as a template will have “.xlst” extension. Table 2.1 shows the full list of file extensions (Microsoft, 2017e).

Table 2.1: Microsoft Excel File Name Extensions (Microsoft, 2017e).

XML File Type	Extension
Workbook	.xlsx
Macro-enabled workbook	.xlsm
Template	.xltx
Macro-enabled template	.xltn
Non-XML binary workbook	.xlsb
Macro-enabled add-in	.xlam

2.6. LibreOffice

LibreOffice Suite is an open source Office package developed by the Document Foundation and is currently used by tens of millions of people around the world. It consists of Writer (word processing), Calc (spreadsheets), Impress (presentations), Draw (vector graphics and flow charts), Base (databases) and Math (formula editing) (LibreOffice, 2016a). By default, LibreOffice uses the ISO standardized Open Document Format (ODF) but also supports the OOXML format. ODF is an XML based standard for sharing files across platforms. LibreOffice version 4.2 and later can read and write to OOXML files. Writing to OOXML format is an option within the LibreOffice software with a warning of possible loss in content or formatting (Document Foundation Wiki, 2016). Output files from LibreOffice saved in OOXML format can be forensically analysed within the scope of this research. LibreOffice Calc is more or less similar in structure and functionality to Microsoft Excel. Table 2.2 shows the file formats supported by LibreOffice (LibreOffice, 2016b).

Table 2.2: LibreOffice ODF File Name Extensions (LibreOffice, 2016b)

XML File Type	Extension
Writer (Document)	.odt
Calc (Spreadsheet)	.ods

XML File Type	Extension
Impress (Presentation)	.odp
Draw (Illustration or Graphics)	.odg

2.7. Relevant Research

Research done by Garfinkel (2009) on “New XML-based Files Implications for Forensics” concluded that unique identifiers stored in OOXML documents are very important in tracking the movements and edits to a document as they are preserved when a document is edited. These identifiers are 32-bit numbers that uniquely identify revisions within a document. The research done by Garfinkel (2009) created 2007 Word documents, made a couple of revisions and saved the files with different file names. It was discovered that the Revision Identifiers (RI) were preserved across all saved files, meaning it is possible to determine and correlate a document’s editing history even if change tracking is not enabled. This means it is possible to prove that one document resulted from another. However, it is possible to manually alter these identifiers and lose track of a document’s editing history or maliciously implicate a user. OOXML documents store timestamps in the ZIP archive of when a document is created or modified (Garfinkel, 2009). These timestamps could be important in correlating document revision history, determine multiple editing sessions or indicate tampering with a document.

Langweg (2012) in his research on OOXML File Analysis of the Terrorist Manual Related to the 22/7 Attacks analysed the Microsoft word “manifest” distributed by the suspected terrorist of the 22nd July 2011 attacks in Oslo, Norway and on Utøya islands. This research studied the Revision Identifiers (RI) in Office Word documents in order to find out how many times the document had been edited and who was the original author. Metadata was extracted from *docProps/app.xml*, *docProps/core.xml* and *settings.xml* files and used to analyse generated document table of contents, document revisions, changes in formatting and language metadata of paragraphs to find evidence of more than one author. Langweg (2012) research was rather manual and was carried out to find out if the OOXML structure of the document was consistent with claims by the suspect apprehended for the terrorist act, and to determine if there had been additional authors on the Microsoft Office document. He manually analysed the generated Word document table of contents, document revisions, changes in formatting and language metadata of paragraphs to find evidence of more than one author. He was able to determine how many times the document had

been revised and saved using extracted revision identifiers and metadata. The findings were that the terrorist manual had been edited (saved) 320 times over the period of creating, composing and editing the document. Examining the pictures in the document using ExifTool revealed that the pictures must have been save on a Windows machine.

Didriksen (2014) in his thesis research on “Forensic Analysis of OOXML Documents” and presented to Department of Computer Science and Media Technology, Gjøvik University College extended the work of (Langweg, 2012). His research was based on finding out the forensic value of OOXML documents and what kind of metadata can be extracted from these documents to support forensic investigations. He also tended to find out the forensic metadata difference between different popular office suites such as Microsoft Office, LibreOffice and OpenOffice. This research employed qualitative research by case studies, experimental research by studying capabilities of different forensic tools that are able to extract OOXML metadata including *read_open_xml.pl*, *DOCXRevisions*, *DSO Tool*, *Encase Forensic*, *Forensic Toolkit* and literature review.

The research experiments proved that OOXML documents metadata and revision identifiers can be trusted to be used for forensic investigations. It specifically found out that even if a document was tampered with and metadata deleted, revision identifiers would still be intact. It was able to determine that manual alterations to documents can be tracked since the timestamps are preserved even after the document has been modified by a Word processor. The research developed a tool called *OOXML Forensic Analysis Tool (OFAT)* which was presented to investigators from the National Authority for Investigation and Prosecution of Economic, and Environmental Crime in Norway (Norwegian: Økokrim) and NCIS Norway (Norwegian: Kripos). This tool was able to validate Word documents by using *OpenXmlValidator* method in the official Microsoft OpenXML SDK, extract document metadata including revision identifiers and correlate the two (Didriksen, 2014). A key finding of this research is that both LibreOffice and Google Docs do not use revision identifiers. LibreOffice strips all existing identifiers from *word/document.xml* and *word/settings.xml* files when it saves an edited document, while Google Docs replaces all existing identifiers with a null sequence. The practical implication of these findings is that OOXML documents created or edited in LibreOffice or Google Docs cannot be used in a revision identifier comparison process thus reducing their forensic usefulness. Thumbnails created in Word 2007 are unreadable, but it is possible to see how the content was structured (Didriksen, 2014). Word 2010,

2013 and 365 produce readable thumbnails, while Word Online, LibreOffice Writer and Google Docs do not support thumbnails. Thumbnails produced in Word 2010, 2013 and 365 are therefore more forensically useful than the other office suites.

Since the research by Didriksen (2014) was limited to Word documents, it suggested that future research should dwell on spreadsheets and presentations and extend this to LibreOffice and OpenOffice documents. The forensic tool developed by this research lacked good visualization and it suggested further improvement to this.

Zhangjie, Xingming, Yuling, & Li (2011) came up with a forensic method based on the unique value of the revision identifier (RI) of OOXML Word to determine the source of suspicious electronic documents and the original author. The RI values are important in determining the source of OOXML documents. If text or characters are copied from one document to another, forensic investigation can reveal that the two documents are from the same source, thus helping to reveal copied documents, templates and information. Timestamp information on dates of document creation and modification were used to reveal the original author of the document and the timeline of revisions and edits.

Muhamad (2011) in his thesis studied data hiding techniques in OOXML files. He categorized these techniques into five, namely: data hiding using OOXML relationship structure; data hiding using XML format feature; data hiding using XML format feature and OOXML relationship structure; data hiding using OOXML file embedded resource architecture; and data hiding using OOXML flexibility of swapping parts using steganographic techniques. The study came up with OOMXQA algorithm that uses XQuery code and can be embedded with any steganalysis and detection tool to query XML metadata of online documents. Muhamad (2011) concluded that methods can be used to hide data within OOXML files that cannot be detected by text editors or document inspector feature of Microsoft Office. Data is normally hidden in the OOXML ZIP archive and it can have other parts or metadata. If the hidden data properly satisfies the relationships order of the OOXML file, then it becomes difficult to recover this data as it will go undetected.

2.8. Existing Forensic Tools

There are existing forensic tools that can analyse OOXML files and these include *Advanced OOXML Carver*, *Encase Forensic*, *read_open_xml.pl*, *DOCXRevisions*, *DSO Tool*, *Forensic*

Toolkit, Python-OOXML, Forensic Toolkit, OfficeDissector, FOCA, Libextractor, and MetaGooful. This research reviews documented information on some of these tools to find out their strengths and weaknesses with respect to metadata extraction and analysis of OOXML spreadsheets. Below is a review of the capabilities of some selected tools.

Advanced OOXML Carver

Schicht (2011) showed that the Advanced OOXML Carver tool was developed from a research on that dwelt on recovering damaged ZIP packages of OOXML documents. It was initially meant for newer Word DOCX, but applies equally well to XLSX, PPTX and ZIP archives generally. The main component of the tool component performs searching, decompressing and generation of logs (Schicht, 2011). This is supported by other modules of the tool that can extract metadata, analyse DOCX files for consistency and repair damaged ZIP files.

This tool has been used by users to extract metadata from Word documents and results documented. Sample results by a user is shown in Figure 2.2. It shows that the tool basically extracts metadata such as document properties, revisions and relationships, but does not do automated analysis. It also shows that the output is in text format.

```

###-- Report on: 1_bad2.docx ---###
2011-06-24 19:21:46 (0001) Program start
2011-06-24 19:21:46 (0001)
2011-06-24 19:21:46 (0002) Total matches on local file header signature (504B0403) found: 10
2011-06-24 19:21:47 (0003) Archive 3 found at offset: 0x000012A9 with internal name: word/theme/theme1.xml
2011-06-24 19:21:47 (0005) CRC32 in header: DCAAE620 do not match CRC32 in decompressed data: 49C9D9F1
2011-06-24 19:21:47 (0008) word/theme/theme1.xml was successfully decompressed, but content may not be 100% correct.
2011-06-24 19:21:47 (0009) WARNING: the internal timestamp for word/settings.xml in the zip structure indicates that the part was
manually modified on: 11 oktober 2006 15:40:56
2011-06-24 19:21:52 (0015)
2011-06-24 19:21:52 (0015) Metadata inside word/settings.xml :
2011-06-24 19:21:52 (0016) rsidRoot: 00B118D5
2011-06-24 19:21:52 (0016) rsids: <w:rsidRoot w:val="00B118D5"/><w:rsid w:val="003518D4"/><w:rsid w:val="003B5634"/><w:rsid
w:val="003C6E95"/><w:rsid w:val="00425F6D"/><w:rsid w:val="004F685E"/><w:rsid w:val="0057773C"/><w:rsid w:val="006B03F3"/><w:rsid
w:val="00745E82"/><w:rsid w:val="007F0C3C"/><w:rsid w:val="00830CE9"/><w:rsid w:val="00863A33"/><w:rsid w:val="00865294"/><w:rsid
w:val="008F4350"/><w:rsid w:val="00942AF5"/><w:rsid w:val="00946DAC"/><w:rsid w:val="00B118D5"/><w:rsid w:val="00CD526B"/><w:rsid
w:val="00D10B06"/><w:rsid w:val="00DA169D"/><w:rsid w:val="00E01208"/><w:rsid w:val="00ED5AC6"/>
2011-06-24 19:21:52 (0017)
2011-06-24 19:21:52 (0017) Metadata inside word/_rels/settings.xml.rels :
2011-06-24 19:21:52 (0018) TemplatePath: file:///C:/Documents%20and%20Settings/Jonny/Application%20Data/Microsoft/Maler/NORMAL.DOTM
2011-06-24 19:21:52 (0011)
2011-06-24 19:21:52 (0011) Metadata inside docProps/core.xml :
2011-06-24 19:21:52 (0012) title: TOP SECRET
2011-06-24 19:21:52 (0012) creator: Jonny
2011-06-24 19:21:52 (0012) lastModifiedby: Benny
2011-06-24 19:21:52 (0012) revision: 5
2011-06-24 19:21:52 (0012) lastPrinted: 2002-10-15T10:27:00Z
2011-06-24 19:21:52 (0012) created: 2010-10-01T07:45:00Z
2011-06-24 19:21:52 (0012) modified: 2010-10-25T10:27:00Z
2011-06-24 19:21:52 (0013)
2011-06-24 19:21:52 (0013) Metadata inside docProps/app.xml :
2011-06-24 19:21:52 (0014) Template: Normal
2011-06-24 19:21:52 (0014) TotalTime: 5
2011-06-24 19:21:52 (0014) Pages: 1
2011-06-24 19:21:52 (0014) Words: 214
2011-06-24 19:21:52 (0014) Characters: 1138
2011-06-24 19:21:52 (0014) Application: Microsoft office word
2011-06-24 19:21:52 (0014) DocSecurity: 0
2011-06-24 19:21:52 (0014) Lines: 9
2011-06-24 19:21:52 (0014) Paragraphs: 2
2011-06-24 19:21:52 (0014) ScaleCrop: false
2011-06-24 19:21:52 (0014) company: My Company
2011-06-24 19:21:52 (0014) LinksUpToDate: false
2011-06-24 19:21:52 (0014) CharactersWithSpaces: 1350
2011-06-24 19:21:52 (0014) SharedDoc: false
2011-06-24 19:21:52 (0014) HyperlinksChanged: false
2011-06-24 19:21:52 (0014) AppVersion: 12.0000
2011-06-24 19:21:52 (0099)
2011-06-24 19:21:52 (0099) Program exit

```

Figure 2.2: Advanced OOXML Carver Metadata Extraction from Word Document (Schicht, 2011)

Fingerprinting Organisation with Collected Archives (FOCA)

FOCA is an automated Windows based tool developed by Eleven Pathways that can download published documents from website using Google, Microsoft Bing and Exalead Search engines, extract and analyse metadata from these documents as well as offline documents (Bajpai, 2014). It supports DOC, PPT, PPS, XLS, DOCX, PPTX, PPSX, XLSX, SWX SXC, SXI, ODT, ODS, ODP, PDF, WPD, SVG, SVGZ, INDD, RDP and ICA file formats and can extract metadata related to users, folders, printers, software, emails, operating systems, passwords and servers and matches information in an attempt to identify which documents have been created by the same team and what servers and clients may be inferred from them (Eleven Paths, 2015). FOCA can also be used to perform penetration testing and it can map a network, a feature which is useful for penetration testers. It provides two view tree and timeline, which show events related to files organised by date which enables quick view of events of a certain date (Kumar, 2012). Figure 2.3 shows sample Word metadata being extracted by FOCA.

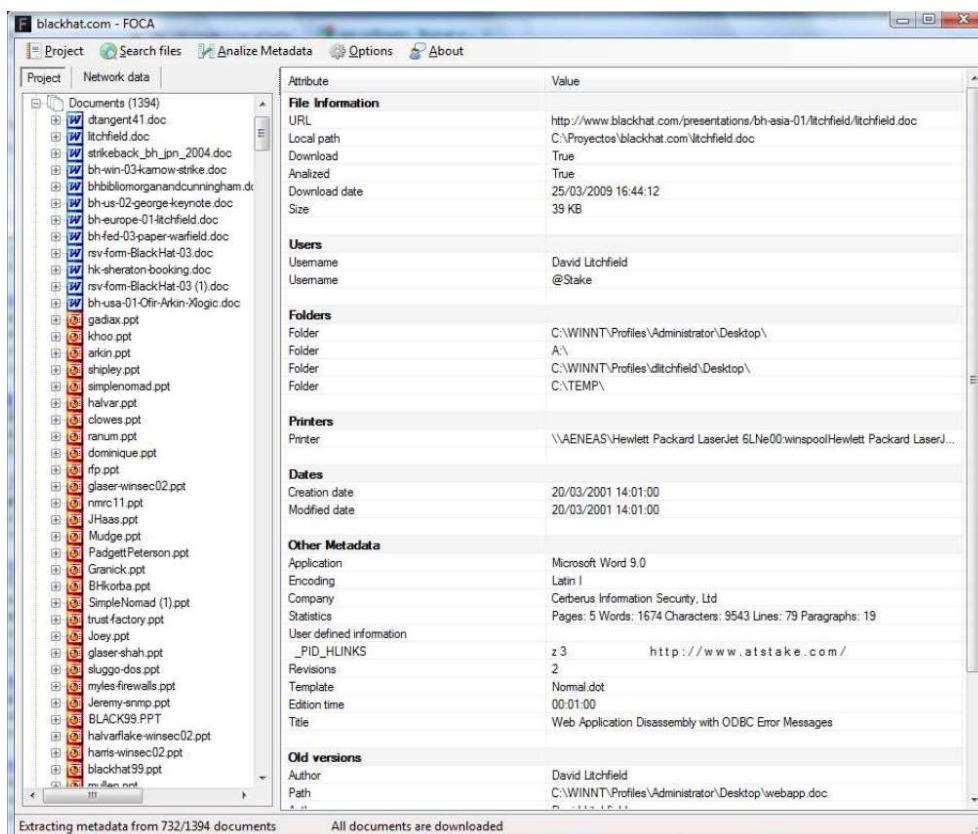


Figure 2.3: FOCA Extracting Metadata Stored in Word Document (Published at Blackhat.com) (Chema, Rando, Oca, & Guzman, 2008)

OfficeDissector 1.0

OfficeDissector is a Python parser library created by Grier Forensics for the Cyber System Assessments Group used for static security analysis of OOXML documents (Grier Forensics, 2015). It parses document properties, parts, content-type, relationships, embedded objects, multimedia, and comments, and exposes metadata via a Python interface. It also provides full JSON export, and a MASTIFF based plugin architecture (Grier Forensics, 2015). This command based tool works on Linux/UNIX OS and fully supports Office Word document metadata extraction. However, it's not proven to fully support spreadsheets.

Python-OOXML 0.13

Python-OOXML is a Python library for parsing Office Open XML files. At the moment, it only supports HTML as output format. Strong emphasis is put on easy customization of the output. The library comes with an importer which is capable of splitting a document into separate chapters. It works both with documents which use Word styles and those that do not (Python Software Foundation, 2016). This tool only supports Office Word documents and output of results in HTML format, therefore, it cannot be fully relied on for forensic analysis of spreadsheets (Python Software Foundation, 2016).

Libextractor

Libre extractor is an open source tool for metadata extraction developed by GNU Operating System (Free Software Foundation, 2016). It supports the following formats: HTML, MAN, PS, DVI, OLE2 (DOC, XLS, PPT), OpenOffice (sxw), StarOffice (sdw), FLAC, MP3 (ID3v1 and ID3v2), OGG, WAV, S3M (Scream Tracker 3), XM (eXtended Module), IT (Impulse Tracker), NSF(E) (NES music), SID (C64 music), EXIV2, JPEG, GIF, PNG, TIFF, DEB, RPM, TAR(.GZ), LZH, LHA, RAR, ZIP, CAB, 7-ZIP, AR, MTREE, PAX, CPIO, ISO9660, SHAR, RAW, XAR, FLV, REAL, RIFF (AVI), MPEG, QT and ASF (Free Software Foundation, 2016). It comes with possibility to write and install additional plugins to enhance its functionality (GNU, 2008).

MetaGooful 2.2

MetaGoofil is an information gathering Python library that extracts metadata from public documents hosted online (Bechtsoudis, 2011). It supports many file types including PDF, DOC, XLS, PPT, DOCX, XLSX and PPTX. This tool traces published documents in a targeted website

using Google search engine, downloads the documents and extracts metadata using libextractor, Hachoir and PdfMiner. It is also able to extract usernames and software version metadata from the documents (Bechtsoudis, 2011).

Read_open_xml.pl

This is a tool written in Perl and it can extract metadata from OOXML documents (Gudjonsson, 2009). It takes an OOXML document as input, extracts it and reads the data stored in *docProps/app.xml* and *docProps/core.xml*, which contains document metadata such as the title, author, number of revisions, number of pages, last printed timestamp, created timestamp, modified timestamp, total editing time, name and version of Word processor. However, it is limited to documents core and application properties and thus not all metadata as summarised in Table A1 of Appendix A could be extracted.

Forensic Tool Kit (FTK)

Forensic Toolkit (FTK) is a commercial forensic tool for creating forensic images, browsing seized file systems, viewing individual seized files, visualizing evidence and performing various evidence analysis (AccessData, 2017). Didriksen (2014) tested the functionality of FTK version 3.4.1.34295, and observed that it seemed to by default have more functionality than EnCase Forensic for handling OOXML documents. Unlike EnCase, OOXML documents loaded in FTK were extracted automatically. However, the FTK only extracted metadata related to core, custom and extended file properties. It did not extract all metadata including revisions needed for forensic investigations. Figure 2.4 shows sample Word metadata being extracted by FTK.

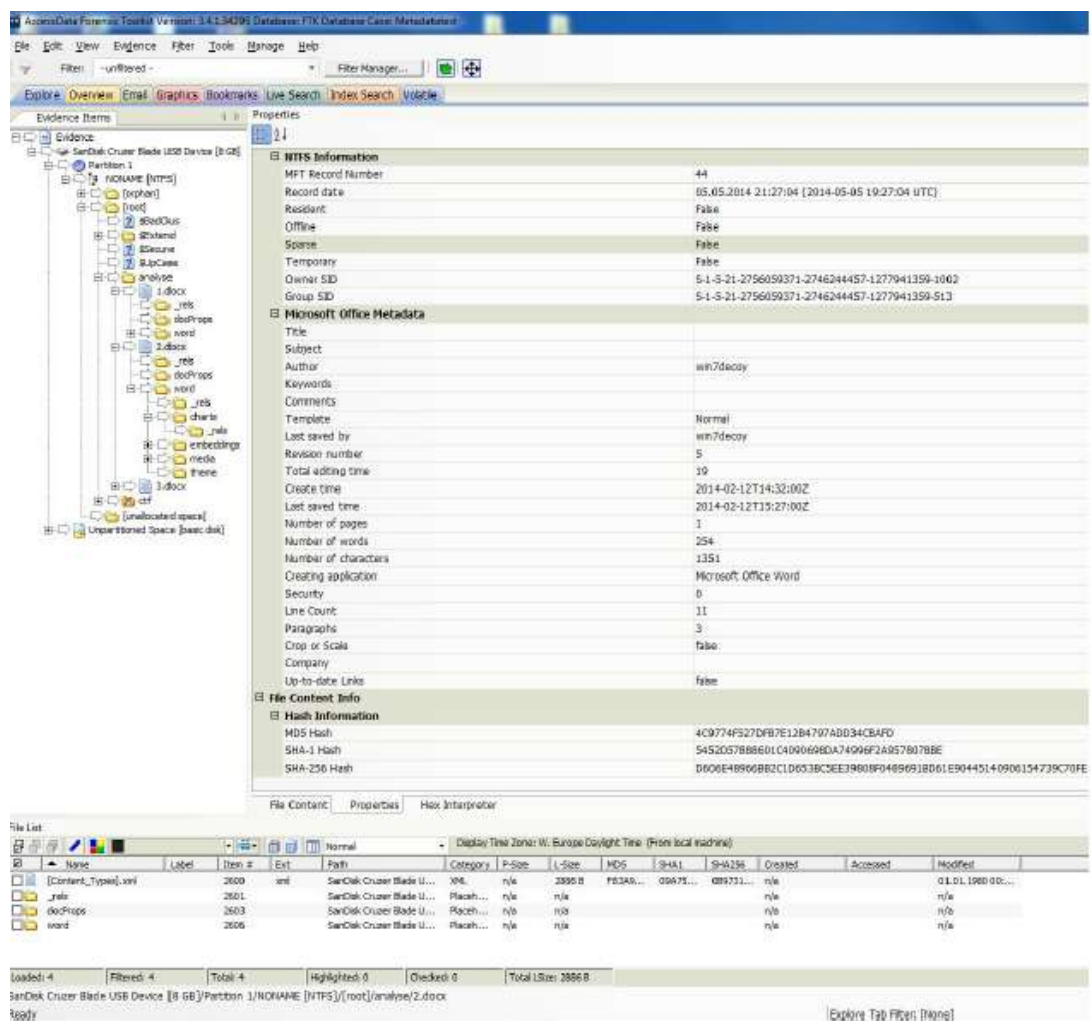


Figure 2.4: FTK Extracting Metadata Stored in Word Document (Didriksen, 2014)

2.9. Conclusions

OOXML spreadsheets are some of the potential sources of forensic evidence. The OOXML standard is fully supported by Microsoft, although other Office products support this standard with limitations. LibreOffice offers read and write while OpenOffice offers only read capabilities of the standard. Research has been carried out on the relevance of OOXML in digital forensics with a conclusion that OOXML metadata can be trusted as source of forensic evidence. However, most of this previous research dwelt on OOXML Office documents thus leaving a gap in research on OOXML spreadsheets. This positively confirms the research hypothesis that previous research on OOXML forensics has not dwelt much on OOXML spreadsheets.

From the review of existing forensic tools, it is evident that these tools are inadequate in extracting and analysing OOXML metadata in spreadsheets. Most of the tools reviewed by this research can extract limited metadata but none has the capability to extract and analyse all the metadata as summarised in Table A1 of Appendix A, including revision metadata from shared OOXML spreadsheets. The forensic tools that can extract some metadata from OOXML spreadsheets output the results in a non-user friendly text format which is difficult to analyse in an automated way. These answers the research question on strengths and weaknesses of available forensic solutions with respect to metadata extraction and analysis of OOXML spreadsheets and positively confirms the research hypothesis that existing forensic tools are limited in adequately extracting and analysing all metadata in OOXML spreadsheets. Therefore, there is need to further analyse OOXML spreadsheet metadata and develop a forensic tool that can extract and analyse metadata in these spreadsheets.

3. Methodology

This chapter describes the methodology employed by this research for forensic analysis of metadata in OOXML spreadsheets. The methodology consists of three main steps: testing of existing forensic tools with regard to OOXML metadata extraction and analysis; system analysis, design and architecture; and Proof of Concept (PoC) implementation of OOXML forensic tool, testing and validation to address the gaps identified with existing tools.

The testing of existing forensic tools is carried out using selected reviewed forensic tools to extract and analyse OOXML metadata from sample OOXML spreadsheet datasets, noting the amount and type of metadata they can extract and analyse against the expected metadata as in Table A1 of Appendix A.

Analysis is carried out by examining the weaknesses of the reviewed and tested forensic tools so as to come up with specifications and requirements of the Proof of Concept implementation of OOXML forensic tool. The design of the tool is carried out using a modified Agile Software Development methodology to address the gaps identified in “Literature Review” and “Testing of Existing Forensic Tools” chapters. Since this research aims at coming up with a Proof of Concept implementation of OOXML forensic tool, applying all aspects of Agile Software Development methodology is not feasible for a single developer and thus only specific aspects of Agile Software Development methodology are applied. It is not expected to have in mind the all expectations and specifications from final users at this point of Proof of Concept implementation of OOXML tool analysis and design, and prototyping is limited to the first working version of the tool. The modified Agile Software Development methodology is suited for these situations where primary focus is developing the application without comprehensively knowing all specifications and requirements (Ambler, 2014a). The research is not initially expected to come up with a comprehensive and exhaustive specifications and design requirements arising from system analysis. In addition, there is likelihood that these specifications and requirements will change as development progresses given that this research is limited in time frame and thus more emphasis is pegged on tool development rather than comprehensive system analysis and documentation.

Testing and validation is done using sound scientific methods to ensure that the test results are repeatable when the same results are obtained using the same methods in the same testing environment, and also reproducible when the same test results are obtained using the same method

in a different testing environment (NIST, 2015). This includes developing a test plan, developing controlled datasets, conducting tests in a controlled environment and validating the test results against known specifications and expectations.

3.1. Testing of Existing Forensic Tools

This research uses the existing forensic tools reviewed in “Literature Review” chapter to extract and analyse metadata from OOXML spreadsheet datasets selected from a controlled pool so as to establish the strengths and weaknesses of these tools. The controlled datasets consist of about 20 different OOXML spreadsheet files of sizes ranging from a few KB to 50 MB since it may not be realistic to have very big spreadsheet datasets. These datasets are manually created using different Microsoft Office and LibreOffice applications and their metadata changes noted including basic properties, rows, cells, sheets, comments, revisions; comments, hyperlinks and embedded images. A few of these sample datasets are intentionally corrupted to investigate the level at which the tool can extract and analyse metadata from these corrupted files. Some of the sample datasets are write protected by passwords to determine performance of the tool on write protected files and also seek to confirm or reject the research hypothesis that it is not possible to extract metadata from write protected OOXML spreadsheets. It is to note that the same spreadsheet files are used for all the tools except for scenarios where these datasets are analysed directly on the cloud, and also in testing and validation.

The research makes conclusions on the strengths and weaknesses of these tools with regard to OOXML metadata extraction and analysis from these spreadsheet datasets, and this guides the system analysis and design of the proposed tool.

3.2. System Analysis, Design and Architecture

System Analysis

Appropriate analysis is done to determine the relationship between the expected specifications and the actual tool to be developed and to identify the goals and purpose of the proposed tool so as to enable designing and developing the tool in an efficient and effective manner in order to determine if it will be economically, socially, technologically and organisationally viable to develop the tool (Dennis, Wixom, & Roth, 2012). In particular, the system analysis comes up with detailed functional and technical specifications as explained below.

- Functional specifications. These specifications contain the proposed functionality of the tool arising from the weaknesses and gaps identified on existing forensic tools summarised in the conclusions of “Literature Review” and “Testing of Existing Forensic Tools” chapters.
- Technical specifications. Using the functional specifications, technical specifications are developed and these details how the functional specifications are implemented.

System Design

This research employs an object-oriented design methodology consisting of:

- Backend design. Use Cases are developed to model the interactions between entities and the system using StarUML software (Jacobson, Spence, & Bittner, 2011). The system model also incorporates Class Diagrams (Ambler, 2014b). These emulate the tool specifications and functionality; and
- Frontend design. Wireframes are developed for front end display and manipulation of metadata. It is intended that the frontend be a web based interface.

System Architecture

System Architecture consists of:

- Web server running in the cloud;
- Web application hosted in a web server. It consists of business processes, reporting module and other services;
- Client interface consisting of standard web browsers through which users interact with the system;
- Internet acting as communication media between web server and client; and
- Cross cutting components such as security and operational management.

3.3. System Implementation, Testing and Validation

System Implementation

In order to implement the system, the chosen development methodology focuses on object-oriented development on a Microsoft platform and follows OWASP guidelines of secure coding standards (OWASP, 2016). The implementation covers the following phases:

- Development and hosting. The tool is developed and hosted locally on Microsoft Internet Information Services (IIS) which is a scalable, reliable and secure development and hosting platform (Microsoft, 2017a). IIS is chosen because the OOXML SDK to be used is a product of Microsoft and therefore highly compatible with this platform (Microsoft, 2011). The developed prototype is then hosted at Microsoft Azure cloud hosting platform that is a secure and scalable cloud hosting platform for ASP.NET web applications (Microsoft, 2017b);
- Backend development. Technical specifications and the developed Use Cases and Class Diagrams from design phase are used to guide the backend development of the tool to implement backend functionality;
- Frontend development. Technical specifications and the developed wireframes from design phase are used to guide web based frontend tool development;
- Languages. Backend development is implemented primarily using C# programming language that is a product of Microsoft within its .NET library and approved by ECMA (ECMA-334) and ISO (ISO/IEC 23270:2006) (ECMA International, 2006a). Frontend development is implemented using ASP.NET which is an open-source server side web application framework (Neudesic, LLC, 2015), SQL Server Reporting Services for generating forensic report, HTML5 (WHATWG Community, 2017), JavaScript (Wiley, 2016) and JQuery (jQuery Foundation, 2017); and
- Frameworks and libraries. OpenXMLSDKTool version 2.5 which is the Microsoft library for OOXML development forms the primary library to develop the tool (Microsoft, 2017c).

Testing

Testing of the tool is conducted including manual and functional tests on all modules of the tool to ascertain that they are working correctly and satisfies the specifications and requirements. Sample spreadsheets datasets created earlier are used as sample datasets for testing. Users to carry out tests are randomly drawn from Strathmore University and work colleagues. They are provided with sample OOXML datasets and each individually carries out the tests and documents findings which are partially used for validation. The tests carried out are detailed below:

- Manual Tests. The research carries out thorough manual tests for each module developed to make sure the tool is working correctly in terms of metadata extraction, analysis and reporting. A manual test plan is created that details the scope of the testing including functions that are

tested and those that are not tested. Manual test cases are developed for metadata extraction, analysis and reporting functionality. It is expected that the tool is able to extract all metadata as summarised in Table A1 of Appendix A if this metadata is available in the OOXML files;

- Functional Tests. Functional tests are done on the overall functionality and stability of the tool to determine if the tool is able to handle large spreadsheets without crashing;
- Security Tests. These tests are carried out by employing penetration tests using OWASP guidelines for penetration testing to ensure the software and platform comply with security standards (OWASP, 2017); and
- Quality Assurance. Results of the tests are compared against functional requirements to determine if the application meets the specifications and requirements.

Validation

Validation of the developed tool is done to determine if the tool serves its intended purpose. The validation will determine if the specifications and requirements were correct and verify that the tool meets these specifications and requirements. This will include validating that developed tool performs all intended functionality as outlined in the specifications, and validating that the results output by the tool are accurate and correct.

1. Validation of the developed tool is achieved by comparing the functionality of the tool against each of the specified requirements, and noting if the tool performs the functions correctly. Three approaches are used for this validation namely:
 - Documentation checks. This is to ascertain the completeness in technical specifications, design, end user and technical manuals. The specifications are cross-checked to make sure they are aligned with user requirements as derived through system analysis;
 - Functional completeness of the tool. Functional completeness of the tool is carried out to ascertain that the tool implements all functionality as per the specifications. Most of the testing is done under manual and functional testing; and
 - Source code. Review of the tool design and data flow analysis to detect poor and potentially incorrect program structures by scrutinizing the source code.
2. Validation of the accuracy and correctness of results obtained by the developed tool is carried out using the following approach:

- Validation plan. This outlines the steps and requirements for the validation and how many times a specification is tested. It also defines the error rates and confidence levels to be achieved by the validation; and
- The tool is then validated using the controlled datasets and results compared against expected results registered when manual changes are made to the files. The metadata extracted by the tool is also be compared with the expected results as in Table A1 of Appendix A to evaluate the success of metadata extraction. Each requirement is validated at least three times to ascertain that they are repeatable and reproducible (Brunty, 2011).

4. Testing of Existing Forensic Tools

This chapter explains in detail the tests that were practically carried out to extract metadata using different forensic tools reviewed in “Literature Review” chapter in order to determine the strengths and weaknesses of the tools as far as OOXML metadata extraction and analysis on spreadsheets is concerned. It also elaborates the types and formats datasets that were used to carry out the tests. The results of these tests are documented in detail and a conclusion at the end of the chapter highlights the strengths and weaknesses of these tools.

4.1. Controlled Datasets Used

OOXML supported spreadsheet datasets of .XLSX format of varying file size were used to perform tests on selected forensic tools. Some of the datasets were created manually and edits done on them in shared mode, while some were downloaded from the internet. One file was write protected by password.

4.2. Metadata Extraction Using Existing Tools

Advanced OOXML Carver

This research performed tests using Advanced OOXML Carver version 4. Test results as shown in Figure 4.1 established that this tool can extract metadata related to document properties from OOXML spreadsheets. This is metadata in *docProps/app.xml*: application name and version, document security, shared document, links and if hyperlinks have been changed; and metadata in *docProps/core.xml*: creator, creation and modification dates and user who last modified the spreadsheet. The output of the extracted metadata is in a plain text file.

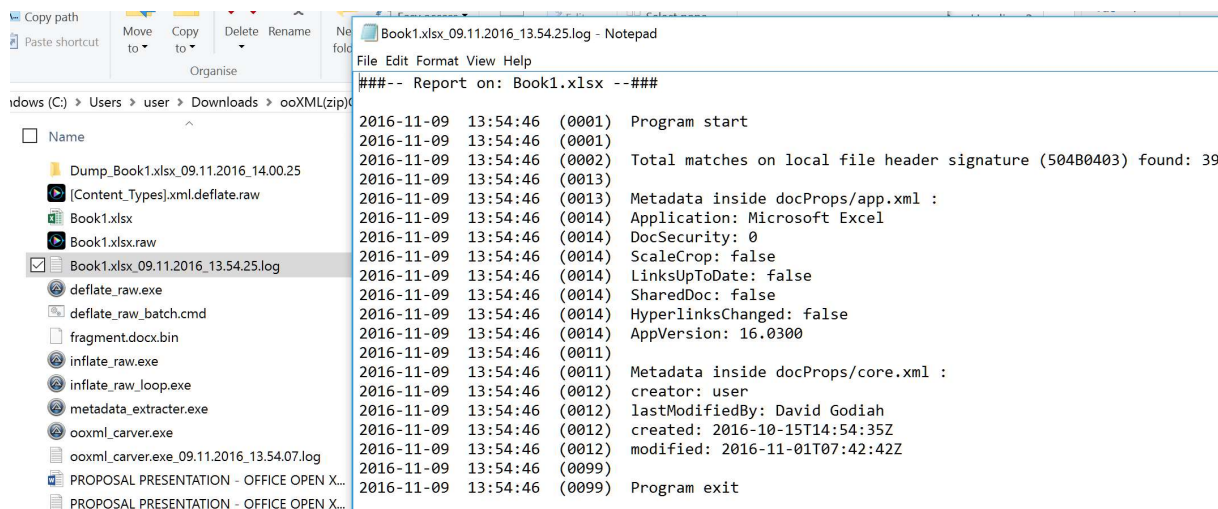


Figure 4.1: Metadata Extraction Results Using Advanced OOXML Carver

The tool was used on write protected files and results were same as in Figure 4.1.

Fingerprinting Organisation with Collected Archives (FOCA)

This research performed tests on FOCA using offline OOXML spreadsheet files to extract metadata and the results are shown in Figures 4.2 and 4.3. It is evident from the results that FOCA is capable of extracting basic metadata related to document properties including application name and version, document security, links, shared document, creator, creation and modification dates. The same results were achieved when write protected datasets were used.

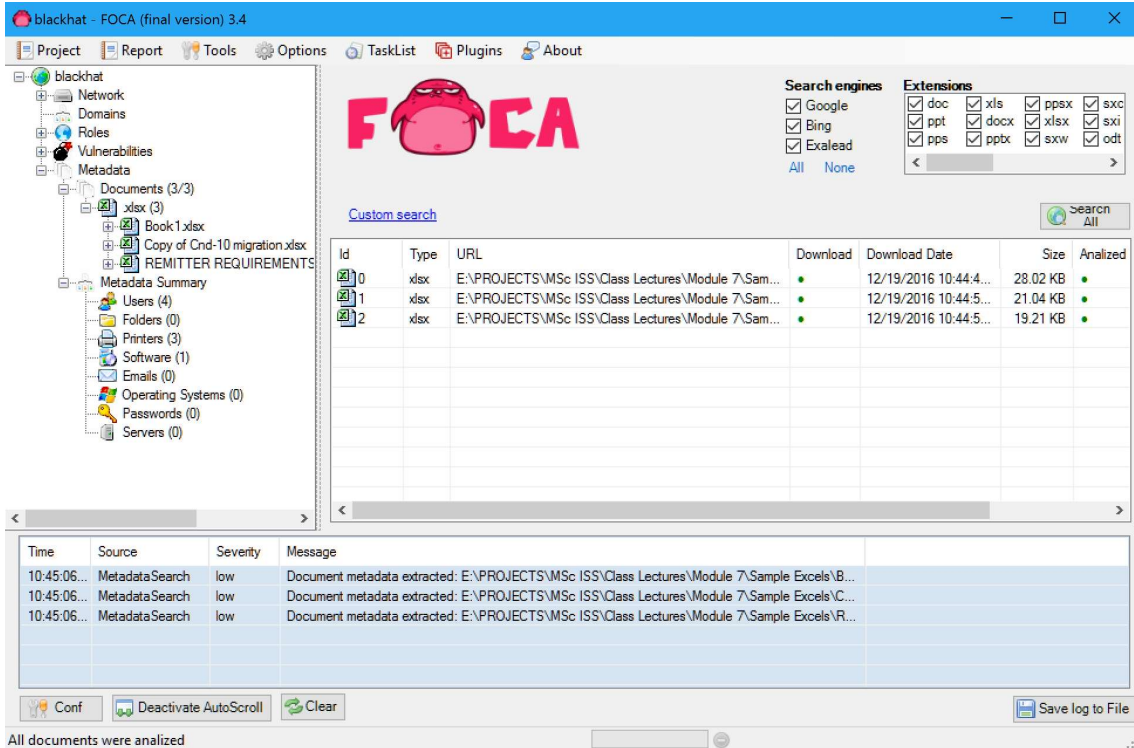


Figure 4.2: Metadata Extraction Results Using FOCA

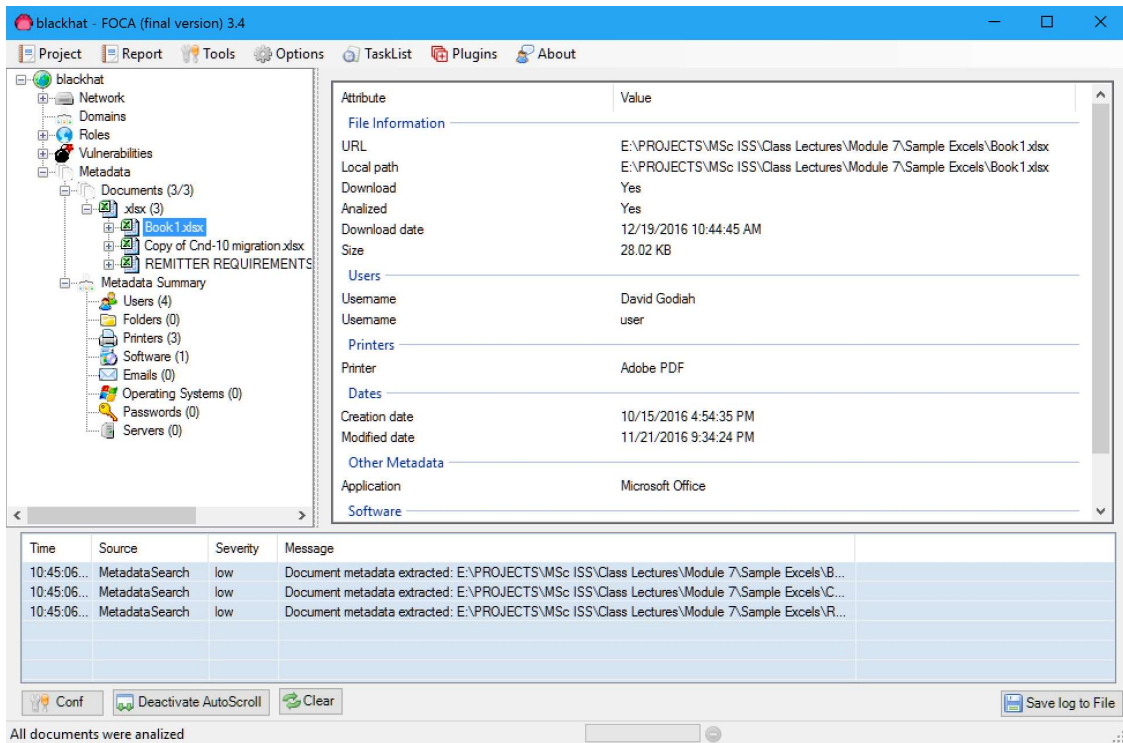
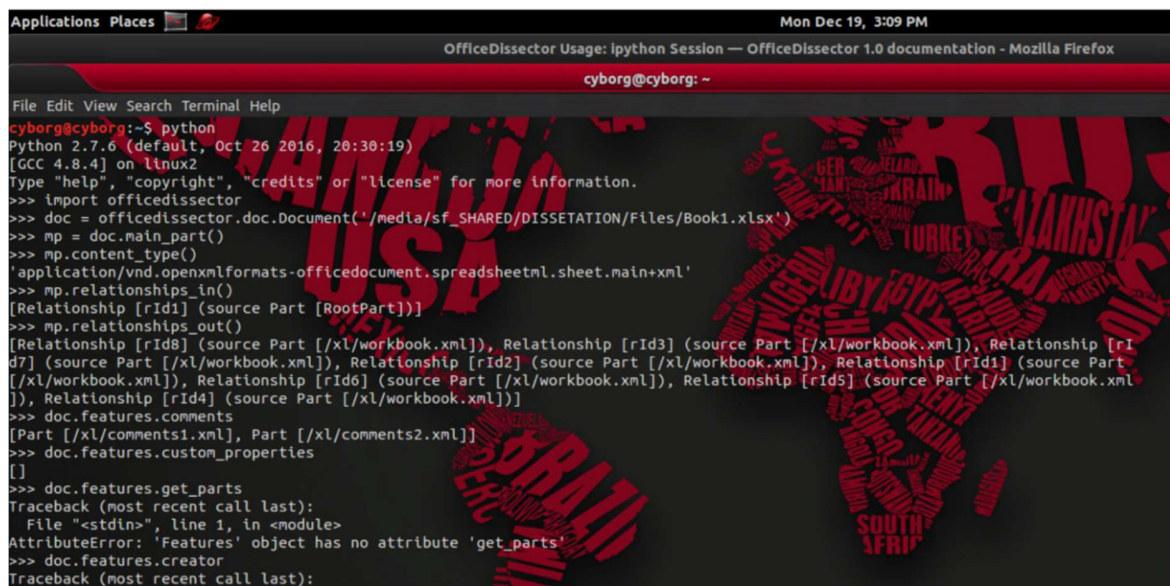


Figure 4.3: Metadata Extraction Results Using FOCA

OfficeDissector

OfficeDissector version 1.0 was installed in Linux Ubuntu 12 with Python version 2.7 for tests. Test results showed that the tool could extract limited metadata from spreadsheets including core properties such as create and modification dates, document name; parts metadata such as comments; relationships and embedded images as shown in Figures 4.4. and 4.5. The tool is command based and poses a challenge to users who do not have appropriate skills. Same results were achieved when the tool was used on write protected OOXML files.



```
Applications Places Mon Dec 19, 3:09 PM
OfficeDissector Usage: ipython Session — OfficeDissector 1.0 documentation - Mozilla Firefox
cyborg@cyborg: ~
File Edit View Search Terminal Help
cyborg@cyborg:~$ python
Python 2.7.6 (default, Oct 26 2016, 20:30:19)
[GCC 4.8.4] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import officedissector
>>> doc = officedissector.doc.Document('/media/sf_SHARED/DISSETATION/Files/Book1.xlsx')
>>> mp = doc.main_part()
>>> mp.content_type()
'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet.main+xml'
>>> mp.relationships_in()
[Relationship [rId1] (source Part [RootPart])]
>>> mp.relationships_out()
[Relationship [rId8] (source Part [/xl/workbook.xml]), Relationship [rId3] (source Part [/xl/workbook.xml]), Relationship [rId7] (source Part [/xl/workbook.xml]), Relationship [rId2] (source Part [/xl/workbook.xml]), Relationship [rId1] (source Part [/xl/workbook.xml]), Relationship [rId6] (source Part [/xl/workbook.xml]), Relationship [rId5] (source Part [/xl/workbook.xml]), Relationship [rId4] (source Part [/xl/workbook.xml])]
>>> doc.features.comments
[Part [/xl/comments1.xml], Part [/xl/comments2.xml]]
>>> doc.features.custom_properties
[]
>>> doc.features.get_parts
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Features' object has no attribute 'get_parts'
>>> doc.features.creator
Traceback (most recent call last):
```

Figure 4.4: Metadata Extraction Results Using OfficeDissector 1.0

```

Applications Places Mon Dec 19, 3:11 PM
Files
cyborg@cyborg: ~
File Edit View Search Terminal Help
d7] (source Part [/xl/workbook.xml]), Relationship [rId2] (source Part [/xl/workbook.xml]), Relationship [rId1] (source Part
[/xl/workbook.xml]), Relationship [rId6] (source Part [/xl/workbook.xml]), Relationship [rId5] (source Part [/xl/workbook.xml
]), Relationship [rId4] (source Part [/xl/workbook.xml])
>>> doc.features.comments
[Part [/xl/comments1.xml], Part [/xl/comments2.xml]]
>>> doc.features.custom_properties
[]
>>> doc.features.get_parts
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Features' object has no attribute 'get_parts'
>>> doc.features.creator
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Features' object has no attribute 'creator'
>>> doc.features.name
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Features' object has no attribute 'name'
>>> doc.core_properties.name
'/docProps/core.xml'
>>> doc.core_properties.created
'2016-10-15T14:54:35Z'
>>> doc.core_properties.modified
'2016-11-21T19:34:24Z'
>>> doc.core_properties.version
'1'

```

Figure 4.5: Metadata Extraction Results Using OfficeDissector 1.0

Python-OOXML 0.13

Python-OOXML version 0.13 was installed in Linux Ubuntu 12 with Python version 2.7 for tests. An attempt to extract metadata from OOXML Excel file failed as shown in Figure 4.6, indicating that this tool does not support OOXML spreadsheets.

```

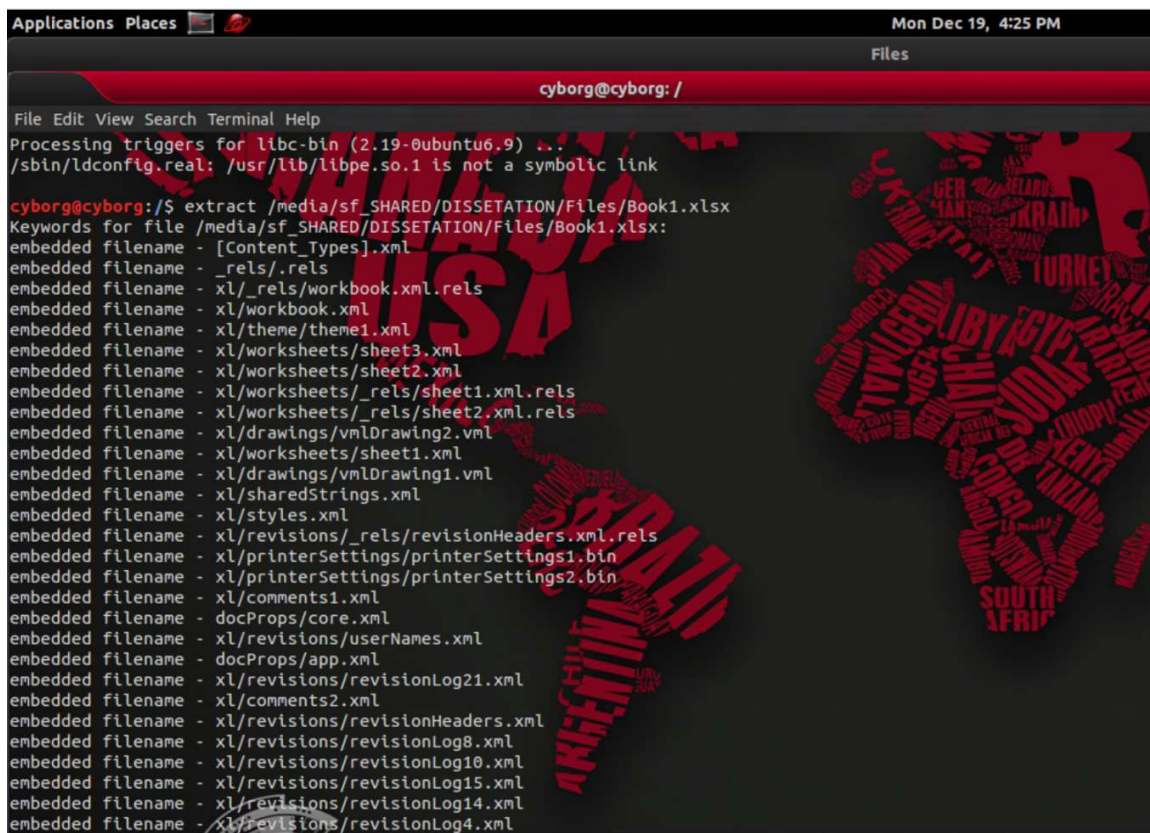
cyborg@cyborg: ~
File Edit View Search Terminal Help
>>> from ooxml import parse, serialize, importer
>>> logging.basicConfig(filename='ooxml.log', level=logging.INFO)
>>> dfile = ooxml.read_from_file('/media/sf_SHARED/DISSETATION/Files/Book1.xlsx')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/lib/python2.7/dist-packages/ooxml/__init__.py", line 52, in r
ead_from_file
    dfile.parse()
  File "/usr/local/lib/python2.7/dist-packages/ooxml/docxfile.py", line 46, in p
arse
    self._doc = parse_from_file(self)
  File "/usr/local/lib/python2.7/dist-packages/ooxml/parse.py", line 652, in par
se_from_file
    doc_content = file_object.read_file('document.xml')
  File "/usr/local/lib/python2.7/dist-packages/ooxml/docxfile.py", line 49, in r
ead_file
    return self.zf.open('word/{}'.format(file_name)).read()
  File "/usr/lib/python2.7/zipfile.py", line 961, in open
    zinfo = self.getinfo(name)
  File "/usr/lib/python2.7/zipfile.py", line 909, in getinfo
    'There is no item named %r in the archive' % name)
KeyError: "There is no item named 'word/document.xml' in the archive"
>>>


```

Figure 4.6: Metadata Extraction Results Using Python-OOXML 0.13

Libextractor

Figure 4.7 and Figure 4.8 show sample tests carried out on Excel file in a Linux Ubuntu 12 and Python version 2.7. The test results show that the tool is able to extract lots of metadata information from shared spreadsheet files including Workbook, Worksheet names, revisions, styles, printers, users and comments as in Figure 4.7 and 4.8. It however gives output in a command line interface which is not user friendly and is difficult to analyse. A test on a write protected file revealed the same results.



```
Applications Places  Mon Dec 19, 4:25 PM
Files
cyborg@cyborg: /

File Edit View Search Terminal Help
Processing triggers for libc-bin (2.19-0ubuntu6.9) ...
/sbin/ldconfig.real: /usr/lib/libpe.so.1 is not a symbolic link

cyborg@cyborg:/$ extract /media/sf_SHARED/DISSETATION/Files/Book1.xlsx
Keywords for file /media/sf_SHARED/DISSETATION/Files/Book1.xlsx:
embedded filename - [Content_Types].xml
embedded filename - _rels/.rels
embedded filename - xl/_rels/workbook.xml.rels
embedded filename - xl/workbook.xml
embedded filename - xl/theme/theme1.xml
embedded filename - xl/worksheets/sheet3.xml
embedded filename - xl/worksheets/sheet2.xml
embedded filename - xl/worksheets/_rels/sheet1.xml.rels
embedded filename - xl/worksheets/_rels/sheet2.xml.rels
embedded filename - xl/drawings/vmlDrawing2.vml
embedded filename - xl/worksheets/sheet1.xml
embedded filename - xl/drawings/vmlDrawing1.vml
embedded filename - xl/sharedStrings.xml
embedded filename - xl/styles.xml
embedded filename - xl/revisions/_rels/revisionHeaders.xml.rels
embedded filename - xl/printerSettings/printerSettings1.bin
embedded filename - xl/printerSettings/printerSettings2.bin
embedded filename - xl/comments1.xml
embedded filename - docProps/core.xml
embedded filename - xl/revisions/userNames.xml
embedded filename - docProps/app.xml
embedded filename - xl/revisions/revisionLog21.xml
embedded filename - xl/comments2.xml
embedded filename - xl/revisions/revisionHeaders.xml
embedded filename - xl/revisions/revisionLog8.xml
embedded filename - xl/revisions/revisionLog10.xml
embedded filename - xl/revisions/revisionLog15.xml
embedded filename - xl/revisions/revisionLog14.xml
embedded filename - xl/revisions/revisionLog4.xml
```

Figure 4.7: Metadata Extraction Results Using Libextractor

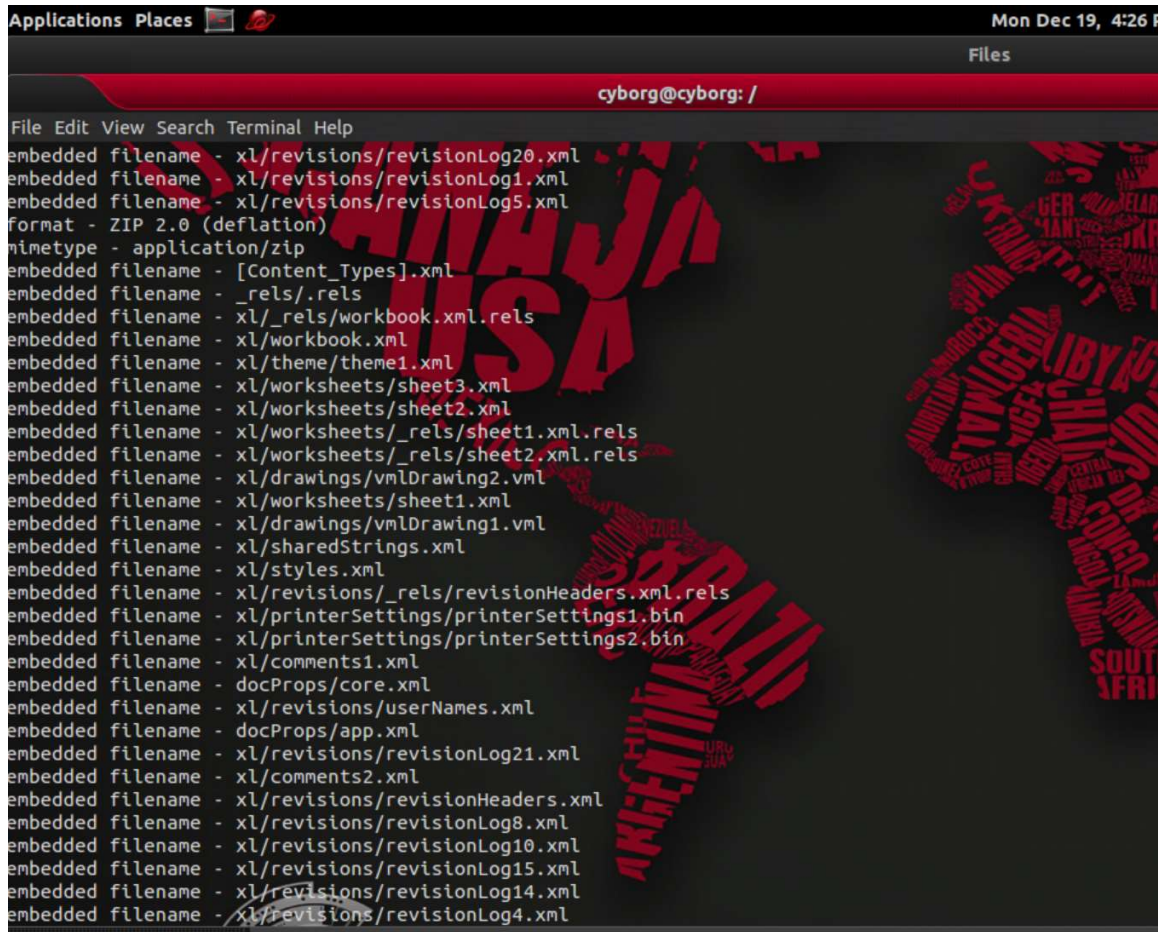
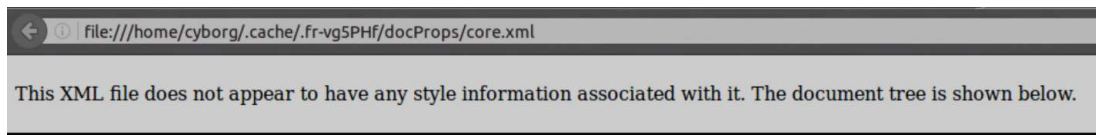


Figure 4.8: Metadata Extraction Results Using Libextractor

Metagoofil

Metagoofil version 2.2 was able to extract metadata related to document properties in *docProps/app.xml* and *docProps/core.xml* from Microsoft.com domain and these were saved into different files in the selected output directory. It could also extract metadata for drawings, images, printer settings, themes and information in Worksheet and Workbook. Figures 4.10, 4.11 and 4.12 show metadata extracted from *docProps/app.xml*, *docProps/core.xml* and */.rels* directory respectively. Appendices C1, C2, C3 and C4 contains additional metadata extracted by Metagoofil 2.2.

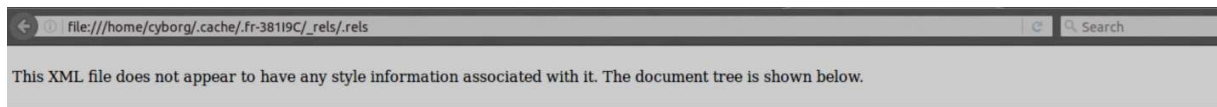


```

- <cp:coreProperties>
  <dc:creator>Beckett Thomsen</dc:creator>
  <cp:lastModifiedBy>v-LoriP</cp:lastModifiedBy>
  <cp:lastPrinted>2007-10-12T06:59:38Z</cp:lastPrinted>
  <dcterms:created xsi:type="dcterms:W3CDTF">2007-10-08T03:44:48Z</dcterms:created>
  <dcterms:modified xsi:type="dcterms:W3CDTF">2009-11-11T19:53:22Z</dcterms:modified>
</cp:coreProperties>

```

Figure 4.11: Metadata Extraction Results in docProps/core.xml File Using Metagoofil 2.2



```

- <Relationships>
  <Relationship Id="rId3" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/extended-properties" Target="docProps/app.xml"/>
  <Relationship Id="rId2" Type="http://schemas.openxmlformats.org/package/2006/relationships/metadata/core-properties" Target="docProps/core.xml"/>
  <Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument" Target="xl/workbook.xml"/>
</Relationships>

```

Figure 4.12: Metadata Extraction Results in /_rels/.rels Directory Using Metagoofil 2.2

Read_open_xml.pl

Read_open_xml.pl was installed in Linux and sample .XLSX files used to extract metadata. The results in Figure 4.13 shows that this tool was able to extract application metadata including type of application, application version, number and names of Worksheets; and file metadata including creator, user who last modified the file, creation and last modification date. The tool was able to extract metadata from write protected OOXML files and give the same output as in Figure 4.13.

```
cyborg@cyborg: /media/sf_SHARED/DISSETATION/Tools
File Edit View Search Terminal Help
cyborg@cyborg:~$ cd /media/sf_SHARED/DISSETATION/Tools
cyborg@cyborg:/media/sf_SHARED/DISSETATION/Tools$ read_open_xml.pl /media/sf_SHARED/DISSETATION/Files/Book1.xlsx
read_open_xml.pl: command not found
cyborg@cyborg:/media/sf_SHARED/DISSETATION/Tools$ ./read_open_xml.pl /media/sf_SHARED/DISSETATION/Files/Book1.xlsx
=====
cmd line: ./read_open_xml.pl /media/sf_SHARED/DISSETATION/Files/Book1.xlsx
=====
Document name: /media/sf_SHARED/DISSETATION/Files/Book1.xlsx
Current Date: Tue Mar 21 12:20:15 CAT 2017
This is a Excel document
-----
Application Metadata
-----
Application = Microsoft Excel
DocSecurity = 0
ScaleCrop = false
HeadingPairs = Worksheets, 3
TitlesOfParts = Sheet1, Godiah, Sheet2
Company =
LinksUpToDate = false
SharedDoc = false
HyperlinksChanged = false
AppVersion = 16.0300
-----
File Metadata
-----
creator = user
lastModifiedBy = David Godiah
created (xsi:type = dcterms:W3CDTF) = 2016-10-15T14:54:35Z
modified (xsi:type = dcterms:W3CDTF) = 2016-11-21T19:34:24Z
cyborg@cyborg:/media/sf_SHARED/DISSETATION/Tools$
```

Figure 4.13: Metadata Extraction Results Using read_open_xml.pl

4.3. Conclusions

OOXML spreadsheets contain lots of metadata as summarised in Table A1 in Appendix A. Most of the tested forensic tools supporting OOXML are not capable of extracting all metadata required to aid effective and complete investigations of spreadsheets, but can only extract basic metadata relating to document properties. However, Metagoofil, Libextrator and OfficeDisector are capable of extracting more metadata related to individual package parts such as comments, Workbook, Worksheet; relationships between different package parts and revisions done on the parts. However, their output is not in a user-friendly manner thus making analysis manual and difficult. For instance, Libextrator outputs a plain text format report that displays the corresponding XML files without giving much details on the content of these files. OfficeDisector requires one to have Python command line skills and an understanding of the library in order to extract metadata, which can be challenging for most users. The results further reveal that write protection by passwords of OOXML spreadsheets does not prevent metadata extraction, thus negating the research hypothesis that it is not possible to extract metadata from write protected files. There is therefore need to have

an automated tool that extracts all metadata in a user friendly and can be used by low skilled users while at the same time give a detailed forensic report.

5. System Analysis, Design and Architecture

This chapter covers in detail analysis done in order to understand the challenges of existing forensic tools and specifications and requirements identified. It also contains details of the design and architecture of the system.

5.1. System Analysis

System analysis was carried out in order to investigate the strengths and weaknesses of the existing forensic tools as covered in conclusions of “Literature Review” and “Testing of Existing Forensic Tools” chapters so as to come up with requirements that will aid the design and consequently development of the Proof of Concept (PoC) implementation of OOXML forensic tool.

Most of the forensic tools reviewed and tested are only capable of extracting basic metadata from OOXML supported spreadsheets without performing any analysis of the metadata. The few that can extract much more metadata are tedious to work with in that they are command line based and the user needs to have sufficient command of the tool’s API in order to extract metadata. Further these tools output results as unanalysed in HTML or plain text format that is difficult to comprehend at a glance especially by low skilled users. These limitations pose a challenge to a forensic investigator in that not all required metadata can be extracted to support a forensic investigation, thus crucial evidence may be uncollected therefore possibly watering down a case. In addition, most of the forensic investigators usually may not be very skilled in command line language that most of the tools use therefore limiting their use. Furthermore, most of the users usually have limited time and requiring them to install and use these tools especially in new platforms is time consuming and may not be productive. It is also notable that none of the tools tested outputs a forensic report in proper standard with good visualization.

All the forensic tools reviewed and studied required the end user to install prior to using them, and more so that these tools are specific to certain platforms and Operating System. This poses a challenge to the user especially if the user does not run the OS required or the user is travelling without his or her computer and needs to extract metadata.

The Proof of Concept implementation of the tool consists of various components including: user interface where users interact with the system with various interactive sections such as uploading

OOXML datasets, metadata extraction, analysis and reporting; backend application logic that process user requests from frontend and return feedback.

5.2. Functional Specifications

Functional specifications are derived as a result of system analysis including functionality, usability, reliability, performance, supportability and security. Table 5.1 summarises the functional specifications identified for the tool.

Table 5.1: Functional Specifications

Category	Functional Specifications
Functionality	Spreadsheet datasets should not be altered by the system
	Highly interactive visualisation for end users
	Extract all metadata as summarised as summarised in Table 1A of Appendix A from OOXML spreadsheets including write protected files
	Analyse the extracted metadata and correlate time series events in a flow format
	Output a detailed and highly visualised standard forensic report
	Compatibility with major browsers for client access
Usability	No installation required for end user
	Independent on end user platform and Operating System
	Easy to use even by low skilled users
Reliability	Online and accessible to users on demand
Performance	Fast and accurate metadata extraction, analysis and reporting with speed more than 500KB of data per second
Supportability	Fully maintained online by hosting entity
	Open to future development to commercial version
Security	Application developed with security considerations following OWASP secure coding standards
	Web application is secured by signed server certificate. Datasets are uploaded via a web page and transferred to the application using secure HTTPS (if affordable)
	Data not stored within the system, metadata is extracted and analysed on the fly

5.3. Technical Specifications

Technical specifications explain how the functional specifications will be implemented. Table 5.2 summarizes the technical specifications.

Table 5.2: Technical Specifications

Item	Technical Specifications
Platform	Web based system installed on the cloud
Web Server	Microsoft IIS
Frontend Development	ASP.NET, HTML5, JavaScript, JQuery
Backend Deployment	C# and associated class libraries (Open XML SDK version 2.5)
Reporting	SQL Server Reporting Services
Security	OWASP secure coding standards

5.4. System Design and Architecture

The design of the tool is composed of use cases, class diagrams and wireframes in order to implement the functional and technical specifications. From the specifications, it is evident that the proposed tool should be able to extract and analyse metadata, and present a forensic report on the analysed metadata. The tool is therefore designed to have three main modules – Metadata Extraction, Metadata Analysis and Forensic Report; and a sub module for end user explanation of what the tool is all about.

Use Cases

Figure 5.1 shows use cases in conventional format.

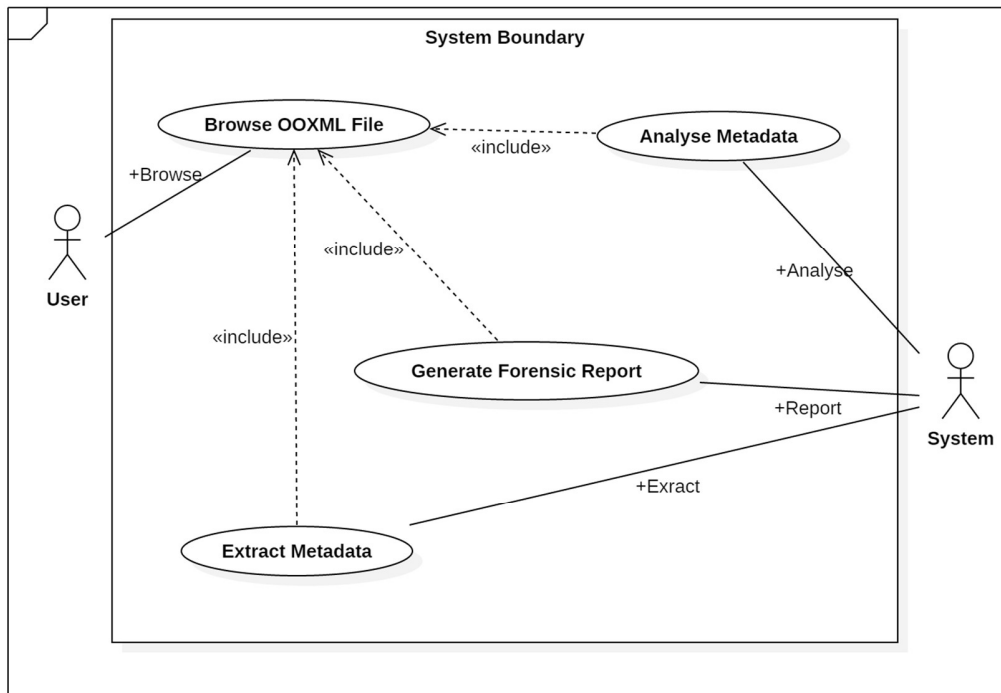


Figure 5.1: Use Case in Conventional Format

The Use Cases in Figure 5.1 are broken down to individual use cases in fully dressed format which expounds on the diagrammatic representation of the Use Case into a more detailed tabular format that makes the specifications clearer and maximizes the flexibility of the design and implementations. Table 5.3 details the Browse OOXML File Use Case in fully dressed format. This Use Case illustrates how a user browses, selects and uploads an OOXML file to extract metadata. The user must have accessed the application on a web browser and the Use Case is successful if the name of the selected file is displayed on the web page, otherwise an error message is displayed.

Table 5.3: Browse OOXML File Use Case

System: OOXML Application	Group ID:
Use Case Name: Browse OOXML File	Use Case ID: UC 1
Primary Actor: User	Priority:
Supporting Actor: System	Use Case Points:
Goal: Browse and uploads OOXML file for metadata extraction	
Trigger: User clicks the file browse button	
Relationships:	
<ul style="list-style-type: none"> ▪ Association: +Browse ▪ Includes: 	

<ul style="list-style-type: none"> ▪ Extends: ▪ Generalization: ▪ Extension Points: 	
Input:	
Preconditions:	
<ul style="list-style-type: none"> ▪ User access application via web browser 	
Normal Flow of Events (Main Success Scenario Steps):	
Actor	System
1. User clicks file browse button on OOXML application 3. User selects file	2. File select window is launched 4. File select window is closed 5. Selected file name is displayed
Alternative and Exceptional Flows:	
Post-conditions on success:	
<ul style="list-style-type: none"> ▪ Selected file name is displayed 	
Post-conditions on failure:	
<ul style="list-style-type: none"> ▪ Error message displayed or selected file name is not displayed 	

Table 5.4 details the Extract Metadata Use Case in fully dressed format. This Use Case illustrates the process to extract and display OOXML metadata. A user must have successfully selected a valid OOXML file and the Use Case is successful when metadata is extracted and displayed in tabular format on the web page, otherwise an error message is displayed.

Table 5.4: Extract Metadata File Use Case

System: OOXML Application	Group ID:
Use Case Name: Extract Metadata	Use Case ID: UC 2
Primary Actor: System	Priority:
Supporting Actor: User	Use Case Points:
Goal: Extract and display OOXML metadata	
Trigger: User clicks the Extract button	
Relationships:	
<ul style="list-style-type: none"> ▪ Association: +Extract ▪ Includes: Browse OOXML File ▪ Extends: ▪ Generalization: ▪ Extension Points: 	
Input: OOXML file contents	
Preconditions:	
<ul style="list-style-type: none"> ▪ Browse OOXML File Use Case is successful 	
Normal Flow of Events (Main Success Scenario Steps):	
Actor	System
1. User clicks Extract button on OOXML application	2. System analyses selected file to make sure it is OOXML supported 3. System extract metadata

	4. System displays extracted metadata in tabular format
Alternative and Exceptional Flows:	
2.1. File is not OOXML supported	
a. System displays error message and requests user to select correct file	
Post-conditions on success:	
▪ Metadata is extracted and displayed in tabular format	
Post-conditions on failure:	
▪ Error message displayed	

Table 5.5 details the Analyse Metadata Use Case in fully dressed format. This Use Case illustrates the process to analyse and present metadata. A user must have successfully selected a valid OOXML file and the Use Case is successful when information derived from metadata analyse is displayed, otherwise an error message is displayed.

Table 5.5: Analyse Metadata Use Case

System: OOXML Application	Group ID:
Use Case Name: Analyse Metadata	Use Case ID: UC 3
Primary Actor: System	Priority:
Supporting Actor: User	Use Case Points:
Goal: Analyse OOXML metadata and display analysis information	
Trigger: User clicks the Analyse button	
Relationships:	
<ul style="list-style-type: none"> ▪ Association: +Analyse ▪ Includes: Browse OOXML File ▪ Extends: ▪ Generalization: ▪ Extension Points: 	
Input: OOXML file contents	
Preconditions:	
▪ Browse OOXML File Use Case is successful	
Normal Flow of Events (Main Success Scenario Steps):	
Actor	System
1. User clicks Analyse button on OOXML application	2. System analyses selected file to make sure it is OOXML supported 3. System analyses metadata 4. System displays analysis information in tabular format
Alternative and Exceptional Flows:	
2.1. File is not OOXML supported	
a. System displays error message and requests user to select correct file	
Post-conditions on success:	
▪ Metadata is analysed and analysis information displayed in tabular format	
Post-conditions on failure:	
▪ Error message displayed	

Table 5.6 details the Generate Forensic Report Use Case in fully dressed format. This Use Case illustrates the process to generate and present a forensic report. A user must have successfully selected a valid OOXML file and the Use Case is successful when a forensic report is displayed, otherwise an error message is displayed.

Table 5.6: Generate Forensic Report Use Case

System: OOXML Application	Group ID:
Use Case Name: Generate Forensic Report	Use Case ID: UC 4
Primary Actor: System	Priority:
Supporting Actor: User	Use Case Points:
Goal: Generate forensic report of analysed metadata	
Trigger: User clicks the Report button	
Relationships:	
<ul style="list-style-type: none"> ▪ Association: +Report ▪ Includes: Browse OOXML File ▪ Extends: ▪ Generalization: ▪ Extension Points: 	
Input: Analyse Metadata Use Case Output	
Preconditions:	
<ul style="list-style-type: none"> ▪ Browse OOXML File Use Case is successful 	
Normal Flow of Events (Main Success Scenario Steps):	
Actor	System
1. User clicks Report button on OOXML application	2. System generates forensic report 3. System displays forensic report
Alternative and Exceptional Flows:	
2.1. Forensic report not generated	
a. System displays error message	
Post-conditions on success:	
<ul style="list-style-type: none"> ▪ Forensic report generated and displayed 	
Post-conditions on failure:	
<ul style="list-style-type: none"> ▪ Error message displayed 	

Class Diagrams

Class diagrams for the tool are developed using Visual Paradigm version 14.0. Figures 5.2 and 5.3 show the class diagrams of all classes designed for the tool. The classes and members are:

- *ClsExtractMedatata* – main class that is accessed directly by metadata extraction, analysis and reporting modules and uses all the other classes.

- *ClsSpreadsheetDocumentProperties* - members and methods related to the Document Properties of the Spreadsheet document;
- *ClsSpreadsheetPackageProperties* - members and methods related to Package Properties of the Spreadsheet document;
- *ClsSpreadsheetFileProperties* – members and methods related to *ExtendedFilePropertiesPart* of the Spreadsheet document;
- *ClsSpreadsheetSharedUsers* – members and methods related to *WorkbookUserDataPart* of the *WorkbookPart* of the Spreadsheet document;
- *ClsSpreadsheetWorkbook* – members and methods related to the *WorkbookPart* of the Spreadsheet document such as *Worksheets*, *WorksheetComments*, *WorksheetDrawings*, *SingeCellTable* and *WorksheetSortMap*;
- *ClsSpreadsheetRevisionHeader* – members and methods related to *WorkbookRevisionHeaderPart* of the *WorkbookPart* of the Spreadsheet document;
- *ClsSpreadsheetSharedSheetRevisionHistory* – members and methods related to the *WorkbookRevisionHeaderPart* of the *WorkbookPart* of the Spreadsheet document and contains all revisions at Worksheet and Cell level. Information captured by this class is related to *ClsSpreadsheetSharedCellRevisionHistory* class by Relationship Id (rId).
- *ClsSpreadsheetSharedCellRevisionHistory* – members and methods related the *WorkbookRevisionLogParts* of the *WorkbookRevisionHeaderPart* of the *WorkbookPart* of the Spreadsheet document, and contains all revisions at Cell level including *RevisionRowColumn*, *RevisionCellChange*, *RevisionComments*, *RevisionInsertSheet*, *RevisionSheetName*, *RevisionMove*, *RevisionDefinedName*, *RevisionAutoFormat*, *RevisionFormat*, *RevisionCustomView*, *RevisionQueryTable* and *RevisionConflict*;
- *Extensions* – static class that contains common methods across all modules; and
- *ClsHelper* – contains function to validate input spreadsheet files.

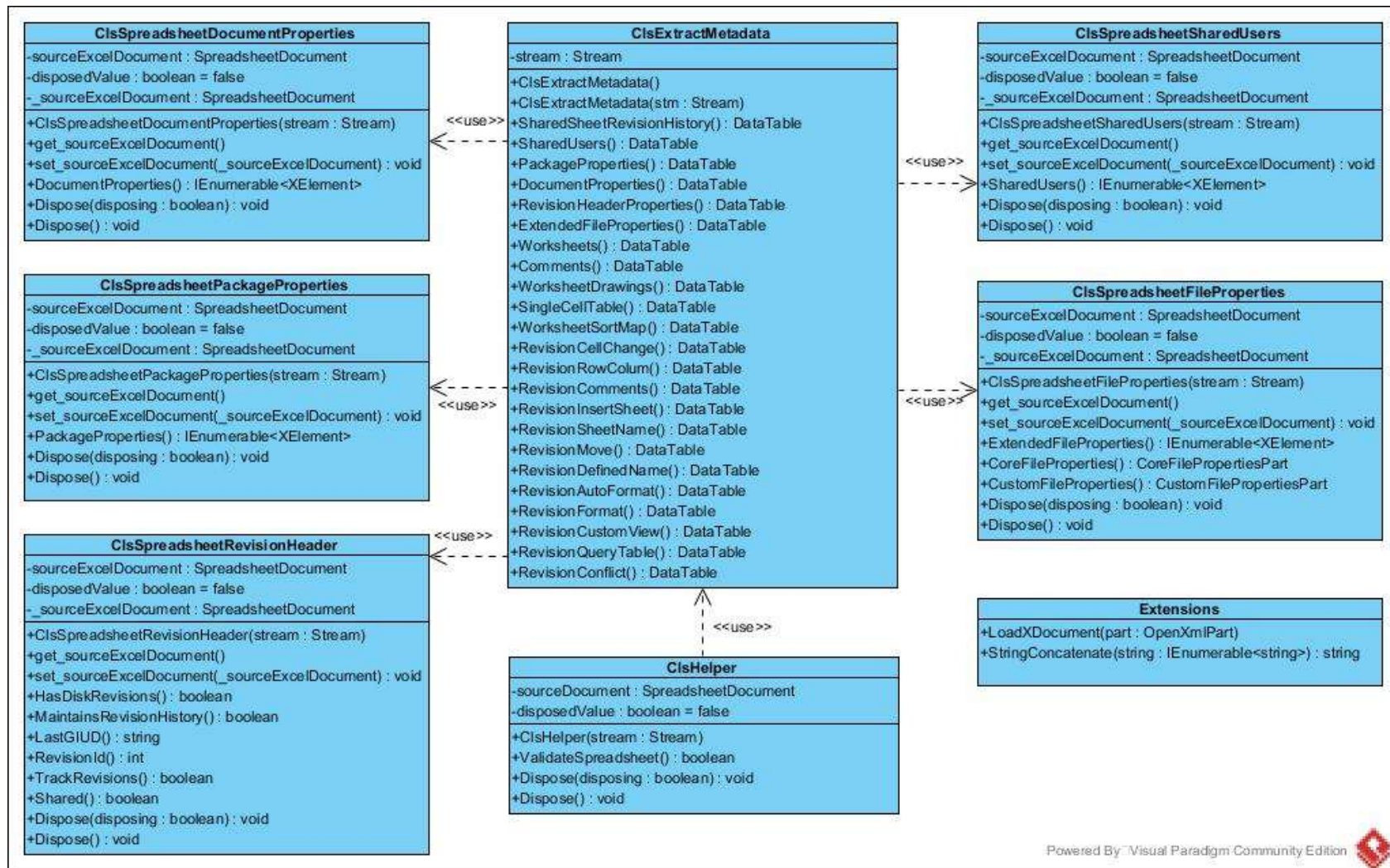


Figure 5.2: Class Diagrams

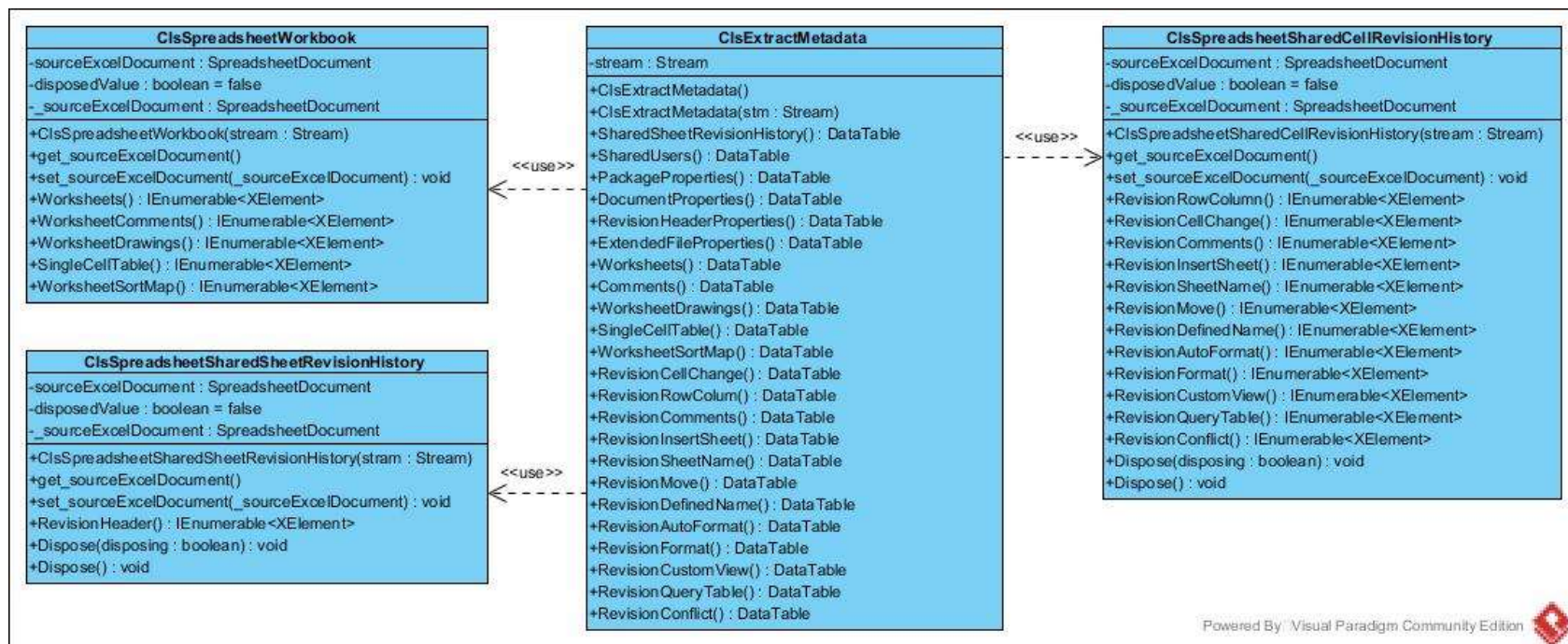


Figure 5.3: Class Diagrams

Wireframes

The design of OOXML tool has four modules namely About, Extract Metadata, Analyse Metadata and Forensic Tool. These modules are designed in separate wireframes using Pencil version 2.0.5. Figure 5.4 shows the wireframe design of “About” web page of the tool. This page contains important information about the tool that can be of relevance to a user such as description of the tool, supported data types, functionality and capability.

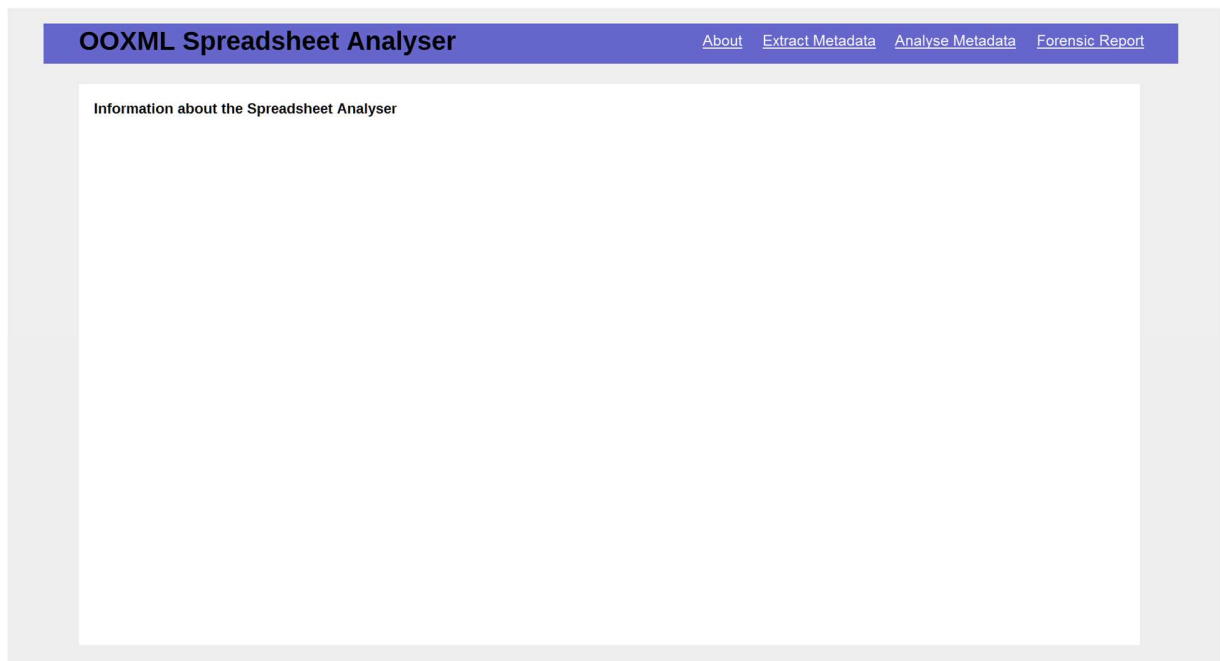


Figure 5.4: About OOXML Spreadsheet Analyser Wireframe

Figure 5.5 shows the wireframe design of “Extract Metadata” web page of the tool. This page contains for browsing OOXML file, extracting and displaying metadata. Metadata display can be categorised according to the types.

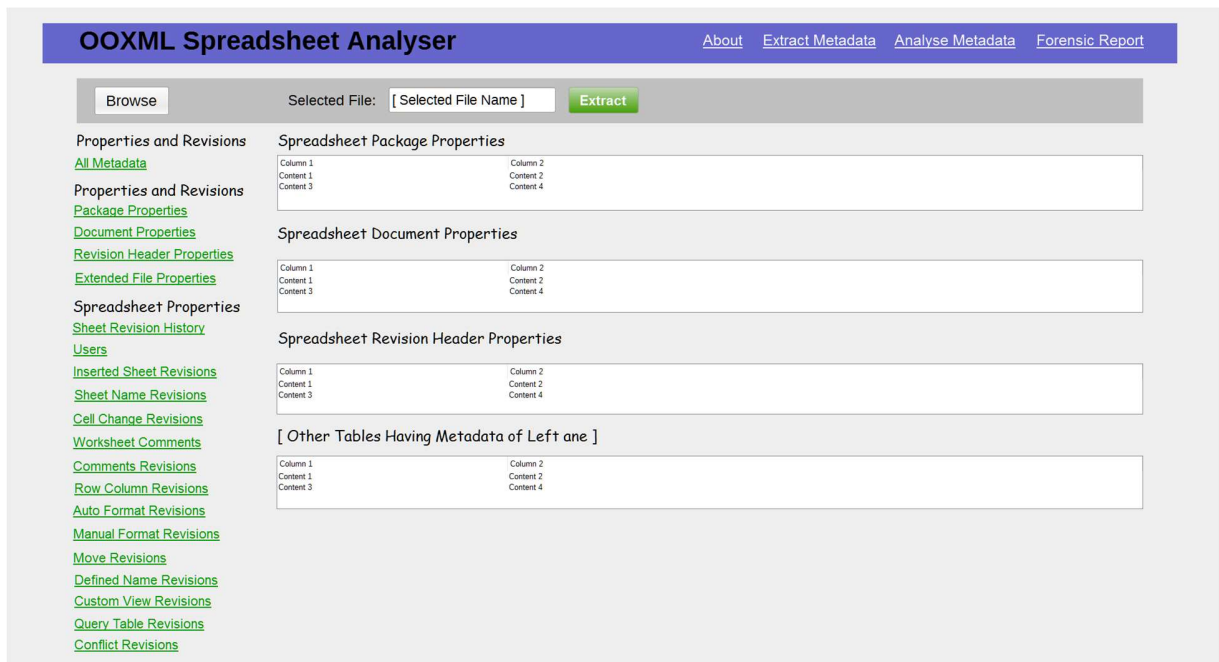


Figure 5.5: Extract Metadata Wireframe

Figure 5.6 shows the wireframe design of “Analyse Metadata” web page of the tool. This page contains functionality to analyse and present metadata and further segregate this information according to all users or selected a user in a time series format of metadata creation date.

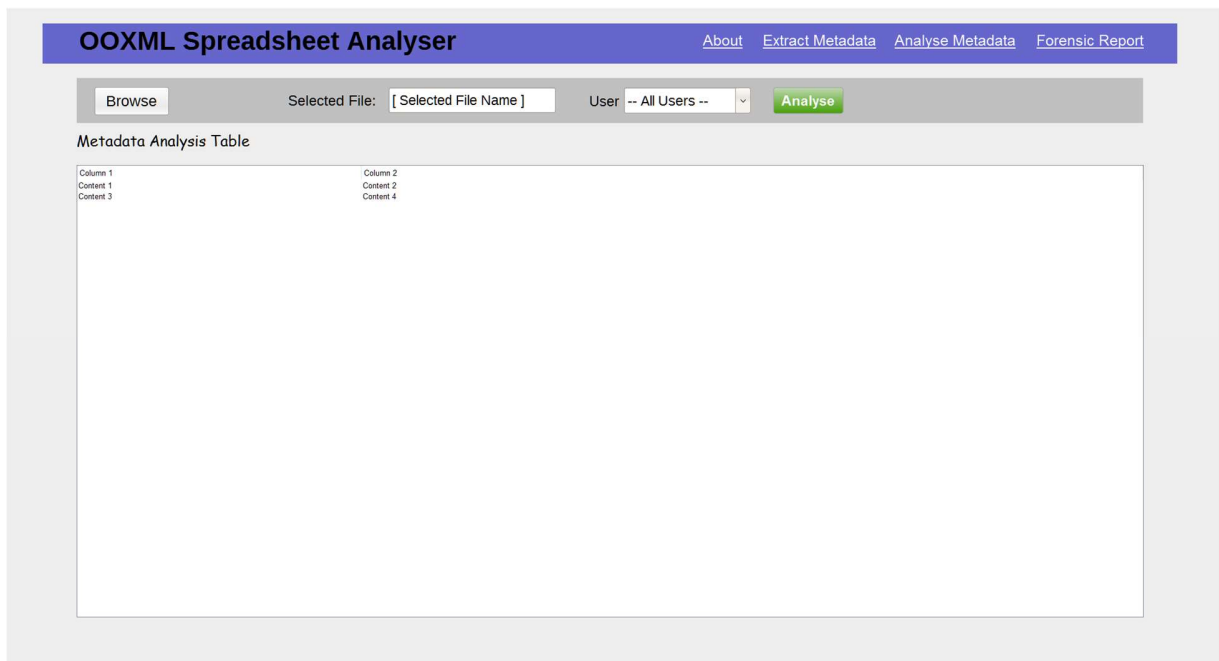


Figure 5.6: Analyse Metadata Wireframe

Figure 5.7 shows the wireframe design of the “Forensic Report” web page of the tool. This page contains functionality to generate and present a forensic report of important information from the analysed metadata.

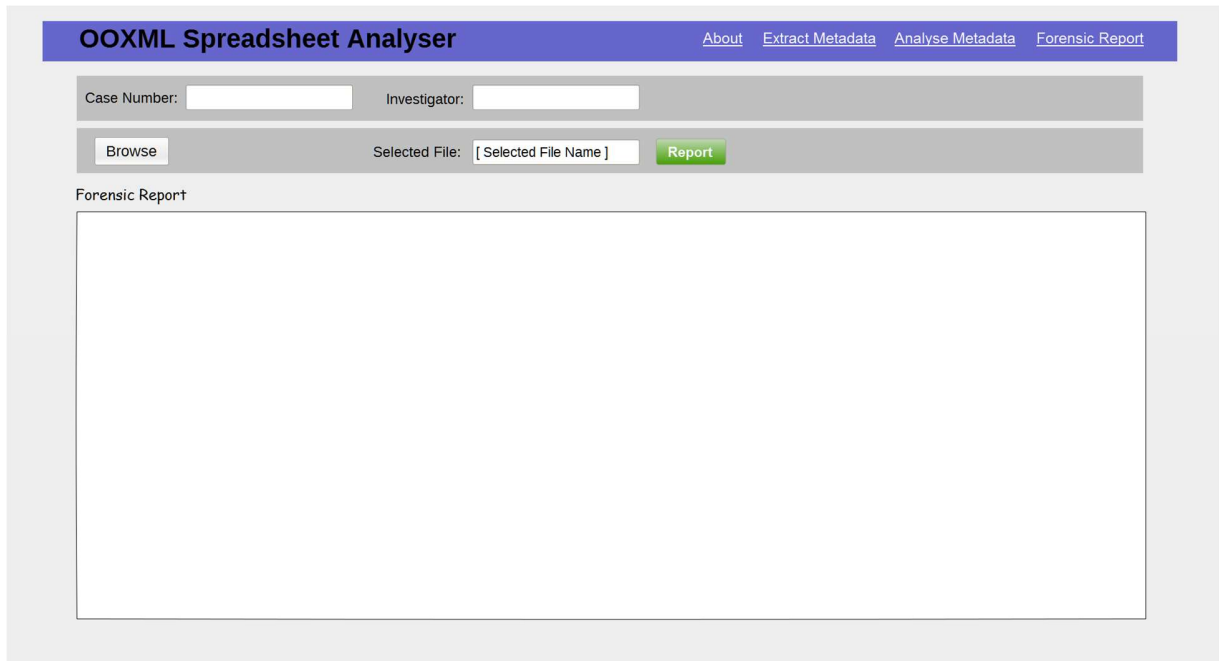


Figure 5.7: Forensic Report Wireframe

System Architecture

The system is designed to have a two-tier architecture with frontend client interface and a backend, which is hosted remotely. The client interface consists of a supported web browser sends requests to server and displays feedback information in a presentable manner. The backend consists of a web server, web application and related services which processes and logic that receives client requests, processes them and present feedback to the client in a presentable manner. The services include security, operational management and communication. The client interface and backend communicate with each other through the internet. Figure 5.8 shows the system architecture.

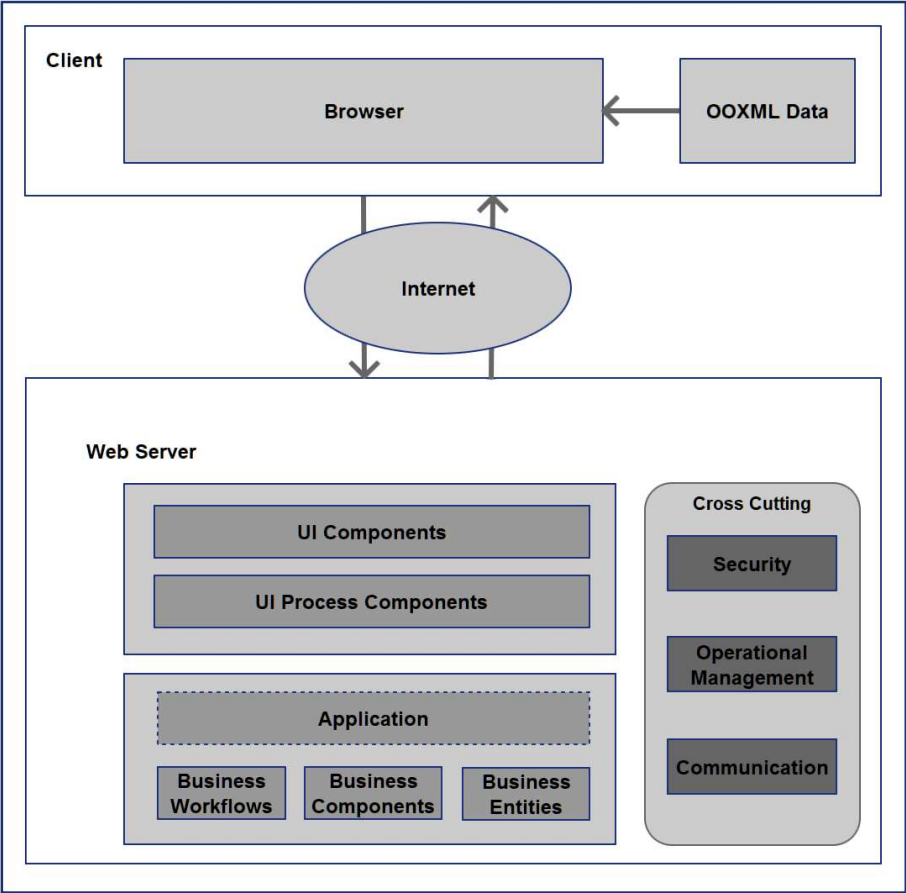


Figure 5.8: System Architecture

6. System Implementation, Testing and Validation

This chapter describes in detail how the implementation, testing and validation of the tool is carried out and the results obtained. Implementation is carried out in accordance to the methodology outlined in Chapter 3 of this research. Testing and validation were carried out on locally deployed version of the tool.

6.1. Implementation

The implementation of the tool incorporates requirements and specifications identified in system analysis so as to implement the system design and architecture. The tool accesses OOXML spreadsheets selected by a user in read-only format to avoid unintended modification of the file. The backend is implemented using the developed Use Cases and Class Diagrams in object-oriented programming technology, while frontend is implemented using the designed wireframes. From system analysis, the tool is best implemented to have three modules namely metadata extraction, analyses and reporting. Each of these modules requires an OOXML file to be selected by a user. The selected file is then uploaded to the application and validated using a validation process developed using the *OpenXmlValidator* available in the OpenXML SDK. The uploaded file is discarded when the metadata extraction, analysis and reporting is complete.

Metadata Extraction Implementation

The tool is able to extract metadata as described in Table A1 in Appendix A from selected shared OOXML spreadsheets and display the metadata in tabular format. It also has the option of extracting and displaying specific metadata that a user is interested in for example comments and specific revisions. Figure 6.1 shows the metadata extraction functionality. Results of the metadata extraction are displayed by type of metadata in tabular format. It is important to note that the tool extracts metadata only if that type of metadata exists within the spreadsheet. For example, if an OOXML does not have comments then no metadata of comments will be extracted and the corresponding tabular display for comments metadata will not be shown.

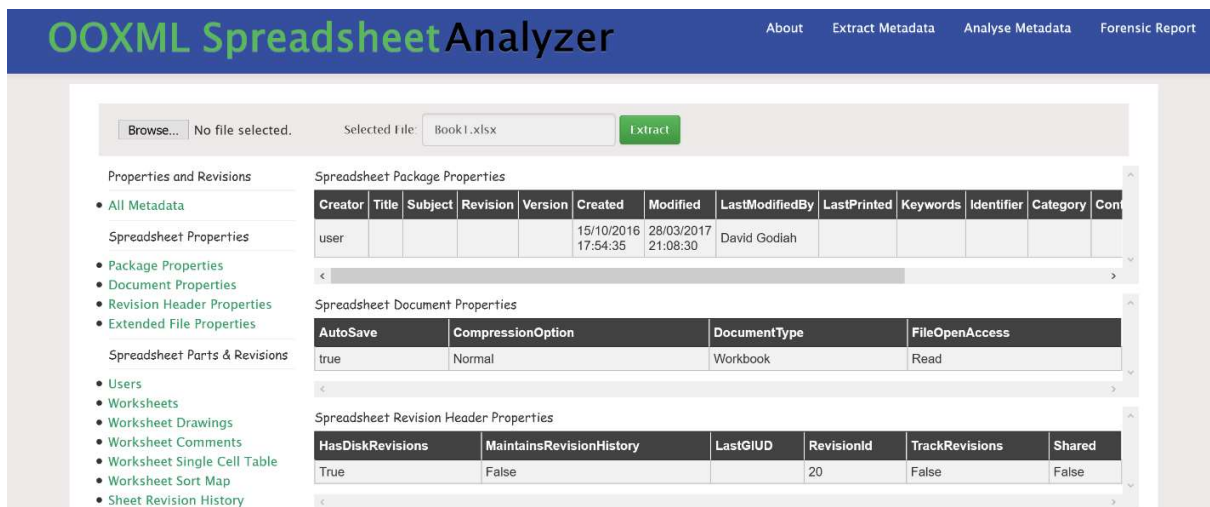


Figure 6.1: Metadata Extraction

Package Properties:

Figure 6.2 shows extracted package properties metadata consisting of user, date and time of creation and modification, user who last modified the file and date and time the file was last printed.

Spreadsheet Package Properties												
Creator	Title	Subject	Revision	Version	Created	Modified	LastModifiedBy	LastPrinted	Keywords	Identifier	Category	Content
user					15/10/2016 17:54:35	28/03/2017 21:08:30	David Godiah					

Figure 6.2: Extracted Package Properties Metadata

Document Properties:

Figure 6.3 shows extracted document properties metadata. This includes if the spreadsheet is in auto save mode, compression option, type of document and the mode of file access.

Spreadsheet Document Properties			
AutoSave	CompressionOption	DocumentType	FileOpenAccess
true	Normal	Workbook	Read

Figure 6.3: Extracted Document Properties Metadata

Revision Header Properties:

Figure 6.4 shows extracted Revision Header Properties metadata, including if the spreadsheet has disk revisions, if it maintains revision history, if it tracks revisions, if it is a shared spreadsheet and revision Id.

Spreadsheet Revision Header Properties

HasDiskRevisions	MaintainsRevisionHistory	LastGIUD	RevisionId	TrackRevisions	Shared
True	False		20	False	False

Figure 6.4: Extracted Revision Header Properties Metadata

Users:

Figure 6.5 shows extracted Users metadata. This metadata includes relationship id, user id, GUID, name of user, and date and time the user accessed the file.

Shared Users

RelationshipId	UserId	Guid	Name	LocalName	DateTime
rld8	-882804752	{0B161687-168D-4D8F-8B9D-D599BC37F1E8}	user	userInfo	2016-10-26T21:57:43
rld8	-429770265	{CC7278B7-0E6F-4B22-B229-9587872F6043}	David Godiah	userInfo	2017-02-28T12:30:36
rld8	-429768026	{CC7278B7-0E6F-4B22-B229-9587872F6043}	David Godiah	userInfo	2017-02-28T12:31:12
rld8	-429764846	{CC7278B7-0E6F-4B22-B229-9587872F6043}	David Godiah	userInfo	2017-02-28T12:32:03
rld8	-429771099	{5DCD52C0-419B-4AF5-ACC4-5A82BAE3E241}	David Godiah	userInfo	2017-03-28T21:09:43

Figure 6.5: Extracted Users Metadata

Worksheets:

Figure 6.6 shows extracted Worksheets metadata. This contains all metadata information within a Worksheet, including innerxml containing serialised markup of all child nodes of the Worksheet and outerxml and outerxml that gives details of the Worksheet and all its child nodes.

Worksheets

RelationshipId	Prefix	InnerText	InnerXml	OuterXml	LocalName
			<pre><x:dimension ref="A1:B11" xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main" /><x:sheetViews xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:sheetView tabSelected="1" workbookViewId="0"> <x:selection activeCell="B4" sqref="B4" /></x:sheetView></x:sheetViews> <x:sheetFormatPr defaultRowHeight="14.25" x14ac:dyDescent="0.45" xmlns:x14ac="http://schemas.microsoft.com/office/spreadsheetml/2009/9/ac" xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main" /><x:cols xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:col min="1" max="2" width="9.59765625" customWidth="1" /></x:cols><x:sheetData xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:row r="1" spans="1:2" x14ac:dyDescent="0.45" xmlns:x14ac="http://schemas.microsoft.com/office/spreadsheetml/2009/9/ac"><x:c r="A1" s="2"><x:v>12</x:v></x:c><x:c r="B1" s="2"><x:v>56</x:v></x:c></x:row> <x:row r="2" spans="1:2" ht="12" customHeight="1" x14ac:dyDescent="0.45" xmlns:x14ac="http://schemas.microsoft.com/office/spreadsheetml/2009/9/ac"><x:c r="A2" s="2" /><x:c r="B2" s="2" /></x:row></pre>	<pre><x:worksheet xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships" xmlns:xdr="http://schemas.openxmlformats.org/drawingml/2006/spreadsheetDrawing" xmlns:x14="http://schemas.microsoft.com/office/spreadsheetml/2009/9/main" xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:x14ac="http://schemas.microsoft.com/office/spreadsheetml/2009/9/ac" mc:Ignorable="x14ac" xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:dimension ref="A1:B11" /><x:sheetViews><x:sheetView tabSelected="1" workbookViewId="0"></pre>	

Figure 6.6: Extracted Worksheet Metadata

Worksheet Comments:

Figure 6.7 shows extracted Worksheet comments metadata that includes the Id of the author, comment text, the cell the comment is made and innerxml containing serialised markup of all child nodes of the cell that contains the comment.

Worksheet Comments

RelationshipId	AuthorId	CommentText	InnerText	InnerXml	Reference	Shapeld
rld2	0		David Godiah: I am graduating	<x:text xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:rPr><x:b /><x:sz val="9" /><x:color indexed="81" /><x:rFont val="Tahoma" /><x:charset val="1" /></x:rPr><x:t>David Godiah:</x:t></x:rPr><x:sz val="9" /><x:color indexed="81" /><x:rFont val="Tahoma" /><x:charset val="1" /></x:rPr><x:t xml:space="preserve"> I am graduating </x:t></x:rPr></x:text>	B4	0
rld2	0		user: good user: I am a hard working person	<x:text xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:rPr><x:b /><x:sz val="9" /><x:color indexed="81" /><x:rFont val="Tahoma" /><x:charset val="1" /></x:rPr><x:t>user:</x:t></x:rPr><x:sz val="9" /><x:color indexed="81" /><x:rFont val="Tahoma" /><x:charset val="1" /></x:rPr><x:t xml:space="preserve">good </x:t></x:rPr><x:rPr><x:b /><x:sz val="9" /><x:color indexed="81" /><x:rFont val="Tahoma" /><x:charset val="1" /></x:rPr><x:t>user:</x:t></x:rPr><x:sz val="9" /><x:color indexed="81" /><x:rFont val="Tahoma" /><x:charset val="1" /></x:rPr><x:t xml:space="preserve"> I am a hard working person </x:t></x:rPr></x:text>	A1	0
rld4	0		user: nice work. Keep it upuser:	<x:text xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:rPr><x:b /><x:sz val="9" /><x:color indexed="81" /><x:rFont val="Tahoma" /><x:charset val="1" /></x:rPr><x:t>user:</x:t></x:rPr><x:sz val="9" /><x:color indexed="81" /><x:rFont val="Tahoma" /><x:charset val="1" /></x:rPr><x:t xml:space="preserve">nice work. Keep it up</x:t></x:rPr><x:rPr><x:b /><x:sz	A1	0

Figure 6.7: Extracted Worksheet Comments

Sheet Revision History:

Figure 6.8 shows extracted Sheet revision history metadata that contains the relationship Id with parent Workbook, GUID, maximum and minimum revision Ids, maximum Sheet Id, innerxml, user, date and time the sheet is edited and innerxml containing serialised markup of all child nodes of the Sheet.

Spreadsheet Shared Sheet Revision History

RelationshipId	Guid	MaxRevisionId	MaxSheetId	MinRevisionId	SheetIdMapCount	UserName	Date Time	InnerText	InnerXml
rld14	{0B161687-168D-4D8F-8B9D-D599BC37F1E8}	11	4	10	3	user	2016-10-26T20:28:00		<x:sheetIdMap count="3" xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:val="1" /><x:sheetId val="2" /><x:val="3" /></x:sheetIdMap><x:recount="2" xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:rld="10" /><x:reviewed rld="11" /></x:reviewedList>
rld15	{B9C26DD4-FB73-4206-80E9-DB03B8280225}		4	12	3	user	2016-10-27T11:38:26		<x:sheetIdMap count="3" xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:val="1" /><x:sheetId val="2" /><x:val="3" /></x:sheetIdMap>
rld16	{F57931B1-CA35-46A5-A1E8-6A6BB803566D}	14	4	13	3	David Godiah	2016-11-01T10:42:42		<x:sheetIdMap count="3" xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:val="1" /><x:sheetId val="2" /><x:val="3" /></x:sheetIdMap>
rld17	{8B460077-D377-421F-8869-C38CC02C1E48}		4		3	David Godiah	2016-11-11T11:57:46		<x:sheetIdMap count="3" xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:val="1" /><x:sheetId val="2" /><x:val="3" /></x:sheetIdMap>
rld18	{7D399B8D-A7F9-4AE8-9702-CEA3D98E6A03}		4		3	David Godiah	2016-11-11T11:58:11		<x:sheetIdMap count="3" xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:val="1" /><x:sheetId val="2" /><x:val="3" /></x:sheetIdMap>

Figure 6.8: Extracted Sheet Revision History Metadata

Cell Change Revisions:

Figure 6.9 shows extracted Cell revision history metadata that consists of relationship Id with parent Sheet, revision Id, Sheet Id, previous and current cell values, user, innerxml, date and time the cell was revised and innerxml containing serialised markup of all child nodes of the cell.

Revision Cell Change

RelationshipId	RevisionId	SheetId	StyleRevision	OldCell	NewCell	InnerText	UserName	DateTime	InnerXml
rld21	16	2	0		hdghhh	hdghhh	David Godiah	2016-11-21T22:34:24	<x:nc r="A3" t="inlin xmlns:x="http://sche /spreadsheetml/200< /><x:t>hdghhh</x:t></
rld21	17	2	0		Gosiah	Gosiah	David Godiah	2016-11-21T22:34:24	<x:nc r="B3" t="inlin xmlns:x="http://sche /spreadsheetml/200< /><x:t>Gosiah</x:t></
rld25	20	3	0		ff	ff	David Godiah	2017-03-28T21:08:14	<x:nc r="A4" t="inlin xmlns:x="http://sche /spreadsheetml/200< /><x:t>ff</x:t></x:is></
rld16	14	2	0		sdfff	sdfff	David Godiah	2016-11-01T10:42:42	<x:nc r="A2" t="inlin xmlns:x="http://sche /spreadsheetml/200< /><x:t>sdfff</x:t></x:is
rld15	12	2	0	This is a rabbit	This is a rabbit		user	2016-10-27T11:38:26	<x:nc r="B1" t="inlin xmlns:x="http://sche /spreadsheetml/200< />is a rabbit</x:t></x:is

Figure 6.9: Extracted Cell Revision History Metadata

Comment Revisions:

Figure 6.10 shows extracted comment revisions metadata. This has relationship Id with parent cell, author, cell, Sheet Id, GUID, old and new lengths of the revision, user, date and time the revision was done.

Shared Revision Comments

RelationshipId	Author	Cell	SheetId	Guid	OldLength	NewLength	Action	UserName	DateTime
rld26	David Godiah	B4	3	{AF6905C1-127D-4FB8-8098-051B85AE4804}		30		David Godiah	2017-03-28T21:08:30
rld20	user	B2	1	{16CF0351-BACD-441D-ACD0-03DFD3C953A3}	24	37		David Godiah	2016-11-21T22:31:07
rld24	David Godiah	B4	1	{A5299794-8FCC-4306-97D3-A1E42A2ECAAB}		24		David Godiah	2017-03-20T23:28:44

Figure 6.10: Extracted Comment Revisions Metadata

Row and Column Revisions:

Figure 6.11 shows extracted row and column revisions metadata consisting of relationship Id with parent Sheet, revision Id, revision action, revision reference, Sheet Id, user, date and time the revision was done.

Shared Revision Row Column

RelationshipId	RevisionId	RevisionAction	RevisionReference	SheetId	LocalName	UserName	DateTime
rld21	15	insertRow	A3:XFD3	2	rrc	David Godiah	2016-11-21T22:34:24
rld16	13	insertRow	A2:XFD2	2	rrc	David Godiah	2016-11-01T10:42:42

Figure 6.11: Extracted Row and Column Revisions Metadata

Custom View:

Figure 6.12 shows extracted custom view metadata consisting of relationship Id, action performed, GUID, user, date and time.

Shared Custom View

RelationshipId	Action	Guid	LocalName	InnerText	InnerXml	UserName	DateTime
rld18	delete	{5A643E54-D3D0-4934-8B59-315B3E8A851F}	rcv			David Godiah	2016-11-11T11:58:11
rld18	add	{5A643E54-D3D0-4934-8B59-315B3E8A851F}	rcv			David Godiah	2016-11-11T11:58:11
rld16	add	{5A643E54-D3D0-4934-8B59-315B3E8A851F}	rcv			David Godiah	2016-11-01T10:42:42
rld19	delete	{5A643E54-D3D0-4934-8B59-315B3E8A851F}	rcv			David Godiah	2016-11-11T11:58:31
rld19	add	{5A643E54-D3D0-4934-8B59-315B3E8A851F}	rcv			David Godiah	2016-11-11T11:58:31

Figure 6.12: Extracted Custom View Metadata

Metadata Analysis Implementation

Analysis was performed on the extracted OOXML metadata, specifically revisions at Worksheet and Cell levels of the spreadsheet document. This is based on relationship and revision identifiers that relate information in *ClsSpreadsheetSharedCellRevisionHistory* class and all other revision classes. The analysis output a chronological timeline of events right from the time the spreadsheet document was created to the time analysis was being performed. The tool provides for the option to analyse revisions done by all users or specific user. Figure 6.13 shows the metadata analysis process.

Figure 6.13: Metadata Analysis

Figure 6.14 shows the analysis of two cell revisions that are related to *WorkbookRevisionHeaderPart* by Relationship Id rId14. Both cells belong to the same Sheet 3 with different revision Ids and the revisions are carried out by the same user.

rId14	
MetadataType: Shared Revision Cell Change	MetadataType: Shared Revision Cell Change
RelationshipId: rId14	RelationshipId: rId14
RevisionId: 10	RevisionId: 11
SheetId: 3	SheetId: 3
StyleRevision: 0	StyleRevision: 0
OldCell:	OldCell:
NewCell: 12	NewCell: 56
InnerText: 12	InnerText: 56
UserName: user	UserName: user
DateTime: 2016-10-26T20:28:00	DateTime: 2016-10-26T20:28:00
InnerXml: 12	InnerXml: 56
ExtensionList:	ExtensionList:
Format: 0	Format: 0
HasPhoneticText: 0	HasPhoneticText: 0
LocalName: rcc	LocalName: rcc
NumberFormatId:	NumberFormatId:
OldDifferentialFormat:	OldDifferentialFormat:
OldFormatting: 0	OldFormatting: 0
OldPhoneticText: 0	OldPhoneticText: 0
RowColumnFormattingAffected: 0	RowColumnFormattingAffected: 0

Figure 6.14: Unique Identifier rId14

Figure 6.15 shows the analysis of two cell revisions that are related to *WorkbookRevisionHeaderPart* by Relationship Ids rId15 and rId16. Both cells belong to the same Sheet 2 with different revision Ids and the revisions are carried out at different times by different users.

rld15	rld16
MetadataType: Shared Revision Cell Change	MetadataType: Shared Revision Cell Change
RelationshipId: rld15	RelationshipId: rld16
RevisionId: 12	RevisionId: 14
SheetId: 2	SheetId: 2
StyleRevision: 0	StyleRevision: 0
OldCell:	OldCell:
NewCell: This is a rabbit	NewCell: sdfff
InnerText: This is a rabbit	InnerText: sdfff
UserName: user	UserName: David Godiah
DateTime: 2016-10-27T11:38:26	DateTime: 2016-11-01T10:42:42
InnerXml: This is a rabbit	InnerXml: sdfff
ExtensionList:	ExtensionList:
Format: 0	Format: 0
HasPhoneticText: 0	HasPhoneticText: 0
LocalName: rcc	LocalName: rcc
NumberFormatId:	NumberFormatId:
OldDifferentialFormat:	OldDifferentialFormat:
OldFormatting: 0	OldFormatting: 0
OldPhoneticText: 0	OldPhoneticText: 0
RowColumnFormattingAffected: 0	RowColumnFormattingAffected: 0

Figure 6.15: Unique Identifiers rld15 and rld16

Figure 6.16 shows the analysis of a cell revision and a comment revision that are related to *WorkbookRevisionHeaderPart* by relationship Ids rld23 and rld24 respectively. Both the cell and comment belong to the same Sheet 1 and the revisions are carried out by the same user at different times.

rld23

MetadataType: Shared Revision Cell Change
RelationshipId: rld23
RevisionId: 19
SheetId: 1
StyleRevision: 0
OldCell: jkk
NewCell: jkk13
InnerText: jkkjkk13
UserName: David Godiah
DateTime: 2017-03-20T23:27:01
InnerXml: jkkjkk13
ExtensionList:
Format: 0
HasPhoneticText: 0
LocalName: rcc
NumberFormatId:
OldDifferentialFormat:
OldFormatting: 0
OldPhoneticText: 0
RowColumnFormattingAffected: 0

rld24

MetadataType: Shared Revision Comments
RelationshipId: rld24
Author: David Godiah
Cell: B4
SheetId: 1
Guid: {A5299794-8FCC-4306-97D3-A1E42A2ECAAB}
OldLength:
NewLength: 24
Action:
UserName: David Godiah
DateTime: 2017-03-20T23:28:44

Figure 6.16: Unique Identifiers rld23 and rld24

Forensic Report Implementation

The forensic report shows a summary of package properties, document properties, comments, users and other Parts of the spreadsheet document, together with analysed revisions at Sheet and Cell levels. A user is able to input the case number and name of investigator that will appear in the report. Figure 6.17 shows a generated forensic report that contains report title, case number, name of investigator, file name, date of investigation and report contents. Figures D1-1, D1-2, D1-3 and D1-4 in Appendix D1 show the various contents of the forensic report with regard to analysed metadata.

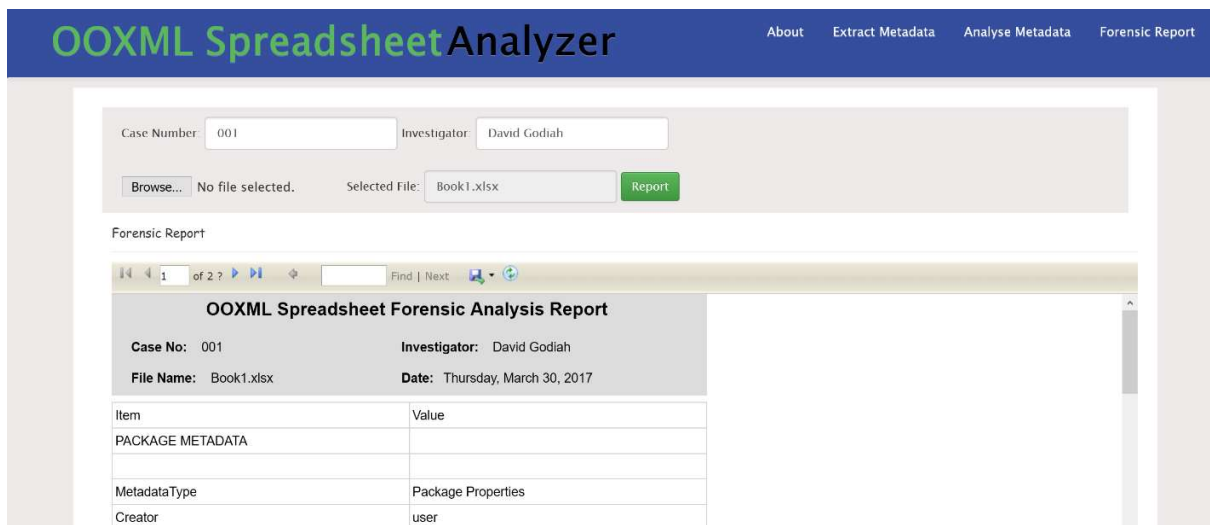


Figure 6.17: Forensic Report

6.2. Testing

Manual Tests

Manual tests were carried out on all the modules of the application by different users. A manual test plan was created that detailed the scope of the testing including functions to be tested (in scope) and those not to be tested (out of scope), and datasets used to carry out the tests. Table 6.1 shows in detail the manual test plan. Test Cases were developed with step by step procedures for metadata extraction, analysis and reporting with the testing results. The sample controlled datasets were used to independently test that metadata extraction, analysis and reporting functionality respond as expected. Users participating in the testing filled in prepared manual test cases with the results of the testing. Each user was required to repeat testing each module at least three times using the same datasets to ensure the results are repeatable. Table D2-1 in Appendix D1 shows manual test cases template for metadata extraction, analysis and reporting used for the tests.

Table 6.1: Manual Test Plan

Item	Value
Features/ Functions Tested (In Scope)	Browsing and validation of OOXML spreadsheet
	Extraction and display of metadata from spreadsheets including write protected files. The metadata to be extracted is related to Package Properties, Document Properties, File Properties, Users, Revision Header, Workbook, Worksheet, Sheet Revision History and Cell Revision History

Item	Value
	Analysis of spreadsheet metadata and display of analysed information
	Generation and display of forensic report
Features/ Functions Not Tested (Out of Scope)	None
Datasets Used	Prepared controlled datasets of varying sizes including write protected files
Testing Environment	Locally hosted application

Figures 6.18 and 6.19 show the test use case results filled in by one of the users testing the tool. From the actual results obtained, it is evident that all test cases succeeded and that the tool is functioning properly as intended.

Test Case ID	Test Case and Procedure	Expected Results	Actual Results	Pass/Fail
1	<p>Test Name: Metadata Extraction</p> <p>Test Procedure:</p> <ul style="list-style-type: none"> Click "Extract Metadata" link Browser OOXML file by clicking on "Browser" button and select the file Click "Extract" button 	<ul style="list-style-type: none"> Metadata extraction page is displayed Selected file name is displayed in "Selected File" text box Metadata is displayed in tabular format segregated into properties and revisions If an invalid file is selected an error message is displayed requesting user to select correct file 	<p>- Metadata extraction page is displayed</p> <p>- Selected file name is displayed in "Selected File" text box</p> <p>- Metadata is displayed in tabular format segregated into properties and revisions</p> <p>- If an invalid file is selected an error message is displayed requesting user to select correct file</p>	Pass
2	<p>Test Name: Metadata Analysis</p> <p>Test Procedure:</p> <ul style="list-style-type: none"> Click "Analyse Metadata" link Browser OOXML file by clicking on "Browse" button and select the file Click "Analyse" button Optionally select a user from "User" dropdown menu 	<ul style="list-style-type: none"> Metadata analysis page is displayed Selected file name is displayed in "Selected File" text box Analysed metadata information is displayed in a time series manner by default for all users If an invalid file is selected an error message is displayed requesting user to select correct file If a user is selected, only analysed metadata specific to that user is displayed 	<p>- Metadata analysis page is displayed</p> <p>- Selected file name is displayed in "Selected File" text box</p> <p>- Analysed metadata information is displayed in a time series manner by default for all users</p> <p>- If an invalid file is selected an error message is displayed requesting user to select correct file</p> <p>- If a user is selected only analysed metadata specific to that user is displayed</p>	Pass

Figure 6.18: Test Use Case Results by User

Test Case ID	Test Case and Procedure	Expected Results	Actual Results	Pass/Fail
3	<p>Test Name: Forensic Report</p> <p>Test Procedure:</p> <ul style="list-style-type: none"> ▪ Click "Forensic Report" link ▪ Input case number and name of investigator ▪ Browser OOXML file by clicking on "Browse" button and select the file ▪ Click "Report" button ▪ Navigate pages of the report by clicking on navigation buttons on the report header ▪ Save report in Excel/PDF/Word format by clicking on "Save" button on the report header 	<ul style="list-style-type: none"> ▪ Forensic report page is displayed ▪ Selected file is displayed in "Selected File" text box ▪ Forensic report is displayed with header containing title, file name, case number, name of investigator, report date and metadata information ▪ Different pages of the report are shown depending on which navigation button is clicked ▪ Report is saved in Excel/PDF/Word format 	<p>- Forensic report page is displayed</p> <p>- Selected file is displayed.</p> <p>- Forensic reports displayed with expected components.</p> <p>- Page navigation works as expected.</p> <p>- Report is saved in selected format</p>	Pass
Tested By: Lydia Ngiri			Date Tested: 2/04/2017	

Figure 6.19: Test Use Case Results by User

Functional Tests

Large spreadsheet dataset of about 12MB was used in the tool to test stability and responsiveness of the tool. Results of this test show that the tool took approximately four seconds to extract metadata from the large file; five seconds to analyse the metadata and six seconds to generate and display the forensic report. The following steps were carried out as functional tests on the developed tool using different datasets in the same testing environment.

The sheet "Sheet2" of spreadsheet document was renamed to "Godiah2" on 3rd April 2017 at 2:56PM. The tool was able to extract metadata of user and last modified date of the document as shown in Figure 6.20. Figure 6.21 shows extracted metadata of the old sheet name as "Sheet2", new sheet name as "Godiah2", Sheet Id as 3, revision Id as 20, relationship Id as rId25, user, date and time.

Spreadsheet Package Properties

Creator	Title	Subject	Revision	Version	Created	Modified	LastModifiedBy	LastPrinted	Keywords	Identifier	Category	Content
user					10/15/2016 5:54:35 PM	4/3/2017 2:56:36 PM	David Godiah					

Figure 6.20: Document Revision Metadata Extraction Testing

Revision Sheet Name

RelationshipId	RevisionId	SheetId	OldName	NewName	LocalName	ExtensionList	UserName	DateTime
rId25	20	3	[Book1.xlsx]Sheet2	[Book1.xlsx]Godiah2	rsnm		David Godiah	2017-04-03T14:56

Figure 6.21: Sheet Name Revision Metadata Extraction Testing

A new Sheet was inserted and renamed to “New Sheet”. Figure 6.22 shows extracted metadata of new inserted sheet including the user, date and time the sheet was inserted, Sheet name as “New Sheet”, Sheet Id as 4, Sheet position as 3, revision Id as 21 and relationship Id as rId27.

Revision Inserted Sheets

RelationshipId	RevisionId	SheetId	SheetPosition	Name	UserName	DateTime
rId27	21	4	3	[Book1.xlsx]New Sheet	David Godiah	2017-04-03T15:16:09

Figure 6.22: Inserted Sheet Metadata Extraction Testing

A new comment with comment text “This is validation” was inserted in cell B3 in the second sheet. Figure 6.23 shows extracted metadata of the comment including text as “This is validation” and the cell as “A1” that the comment was inserted.

Worksheet Comments

RelationshipId	AuthorId	CommentText	InnerText	InnerXml	Reference	Shapeld
rld2	0		user: good user: I am a hard working person	<x:text xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:rPr><x:b /><x:sz val="9" /><x:color indexed="81" /><x:rFont val="Tahoma" /><x:charset val="1" /><x:rPr><x:t>user:</x:t></x:r><x:rPr><x:sz val="9" /><x:color indexed="81" /><x:rFont val="Tahoma" /><x:charset val="1" /><x:rPr><x:t xml:space="preserve">good</x:t></x:r><x:rPr><x:b /><x:sz val="9" /><x:color indexed="81" /><x:rFont val="Tahoma" /><x:charset val="1" /><x:rPr><x:t>user:</x:t></x:r><x:rPr><x:sz val="9" /><x:color indexed="81" /><x:rFont val="Tahoma" /><x:charset val="1" /><x:rPr><x:t xml:space="preserve">I am a hard working person</x:t></x:r></x:text>	A1	0
rld2	1		David Godiah: This is validation comment	<x:text xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:rPr><x:b /><x:sz val="9" /><x:color indexed="81" /><x:rFont val="Tahoma" /><x:charset val="1" /><x:rPr><x:t>David Godiah:</x:t></x:r><x:rPr><x:sz val="9" /><x:color indexed="81" /><x:rFont val="Tahoma" /><x:charset val="1" /><x:rPr><x:t xml:space="preserve">This is validation comment</x:t></x:r></x:text>	B3	0

Figure 6.23: Inserted Comment Metadata Extraction Testing

The text “New Cell” was inserted into a blank cell A1 of sheet “New Sheet”. Figure 6.24 shows extracted metadata for the new cell entry including the cell text as “New Cell”, user, date and time of revision, Sheet Id as 4, relationship Id as rld28 and revision Id as 22.

Revision Cell Change

RelationshipId	RevisionId	SheetId	StyleRevision	OldCell	NewCell	InnerText	UserName	DateTime	InnerXml
rld28	22	4	0		New Cell	New Cell	David Godiah	2017-04-03T15:28:04	<x:nc r="A1" xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:rPr><x:b /><x:sz val="9" /><x:color indexed="81" /><x:rFont val="Tahoma" /><x:charset val="1" /><x:rPr><x:t>New Cell</x:t></x:nc>

Figure 6.24: Inserted Cell Text Metadata Extraction Testing

Cell text “New Cell” in Figure 6.24 was edited to “Edited Text”. Extracted metadata for this revision as in Figure 6.25 shows old cell text as “New Cell”, new cell text as “Edited Text”, user, date and time of revision, Sheet Id as 4 and revision Id as 23 and relationship Id as rld29.

Revision Cell Change

RelationshipId	RevisionId	SheetId	StyleRevision	OldCell	NewCell	InnerText	UserName	DateTime	InnerXml
rld21	16	2	0		hdghhh	hdghhh	David Godiah	2016-11-21T22:34:24	<x:nc r="A3" t="inlin xmlns:x="http://sche /spreadsheetml/200 <x:t>hdghhh</x:t></
rld21	17	2	0		Gosiah	Gosiah	David Godiah	2016-11-21T22:34:24	<x:nc r="B3" t="inlin xmlns:x="http://sche /spreadsheetml/200 <x:t>Gosiah</x:t></
rld16	14	2	0		sdfff	sdfff	David Godiah	2016-11-01T10:42:42	<x:nc r="A2" t="inlin xmlns:x="http://sche /spreadsheetml/200 <x:t>sdfff</x:t></x:is
rld29	23	4	0	New Cell	Edited Text	New CellEdited Text	David Godiah	2017-04-03T15:35:12	<x:oc r="A1" t="inlin xmlns:x="http://sche /spreadsheetml/200 Cell</x:t></x:is></x: t="inlineStr" xmlns:x="http://sche /spreadsheetml/200 <x:t>Edited Text</x:

Figure 6.25: Cell Text Revision Metadata Extraction Testing

The spreadsheet document was printed using “Microsoft to PDF Printer” and saved. Figure 6.26 shows extracted metadata showing when the document was last printed.

Spreadsheet Package Properties

Creator	Title	Subject	Revision	Version	Created	Modified	LastModifiedBy	LastPrinted	Keywords	Identifier	Category	Conte
user			0		10/15/2016 5:54:35 PM	4/3/2017 3:35:12 PM	David Godiah	4/4/2017 1:26:08 PM				

Figure 6.26: Last Printing Date Metadata Extration Testing

Spreadsheet file created using Microsoft Office Excel was opened and saved as a different file using LibreOffice Calc. When metadata was extracted and analysed it was interesting to find that the numbering of relationship and revision identifiers was restarted, but the sequence of revisions was maintained as the analysis of the file saved using Microsoft Office Excel. This means the same chronology of event timeline is obtained when this file is analysed. Figure 6.27 shows analysed OOXML spreadsheet saved using LibreOffice Calc.

rld1	
MetadataType: Shared Revision Cell Change	MetadataType: Shared Revision Cell Change
RelationshipId: rld1	RelationshipId: rld1
RevisionId: 1	RevisionId: 2
SheetId: 3	SheetId: 3
StyleRevision: 0	StyleRevision: 0
OldCell:	OldCell:
NewCell: 12	NewCell: 56
InnerText: 12	InnerText: 56
UserName: user	UserName: user
DateTime: 2016-10-26T20:28:00.000000000Z	DateTime: 2016-10-26T20:28:00.000000000Z
InnerXml: 12	InnerXml: 56
ExtensionList:	ExtensionList:
Format: 0	Format: 0
HasPhoneticText: 0	HasPhoneticText: 0
LocalName: rcc	LocalName: rcc
NumberFormatId:	NumberFormatId:
OldDifferentialFormat:	OldDifferentialFormat:
OldFormatting: 0	OldFormatting: 0
OldPhoneticText: 0	OldPhoneticText: 0
RowColumnFormattingAffected: 0	RowColumnFormattingAffected: 0

Figure 6.27: Analysed Metadata from Spreadsheet Created By LibreOffice Calc

Security Tests

Security tests performed were limited to the web application since it is a Proof of Concept implementation and focus is on the application itself. The tests did not cover the hosting infrastructure and network since these are dynamic depending on where the application is hosted.

The web application was deployed on local IIS web server in Windows 10 machine with IP address 192.168.1.11 on port 8071. Cyborg Hawk 1.1 was installed as a virtual machine within the Windows machine with IP address 10.0.2.15 on interface eth0. Figures 6.28 and 6.29 show the IP addresses of the two machines.

```
Administrator: cmd.exe - Shortcut
Connection-specific DNS Suffix . . . : TOTOLINK
Link-local IPv6 Address . . . . . : fe80::a584:8488:ac0c:2f0c%25
IPv4 Address. . . . . : 192.168.1.11
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.1.1

Tunnel adapter isatap.{AB1CFC47-F6B7-4318-97A7-A8282C8C1C96}:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . . :

Tunnel adapter isatap.TOTOLINK:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . . : TOTOLINK

Tunnel adapter Local Area Connection* 2:

Connection-specific DNS Suffix . . :
IPv6 Address. . . . . : 2001:0:9d38:90d7:2826:a25:3a63:7aae
Link-local IPv6 Address . . . . . : fe80::2826:a25:3a63:7aae%16
Default Gateway . . . . . : ::

C:\WINDOWS\system32>
```

Figure 6.28: Windows 10 Machine IP Address

```
cyborg@cyborg: ~
File Edit View Search Terminal Help
cyborg@cyborg:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:85:55:ee
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe85:55ee/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9909 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5746 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7792042 (7.7 MB)  TX bytes:1774798 (1.7 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:378 errors:0 dropped:0 overruns:0 frame:0
          TX packets:378 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:49935 (49.9 KB)  TX bytes:49935 (49.9 KB)

teredo    Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet6 addr: 2001:0:53aa:64c:d7:2057:3a63:7aae/32 Scope:Global
          inet6 addr: fe80::ffff:ffff:ffff/64 Scope:Link
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1280  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:1056 (1.0 KB)  TX bytes:1296 (1.2 KB)
```

Figure 6.29: Cyborg Hawk 1.1 IP Address

It was confirmed that the web application could be accessed from Cyborg Hawk as shown in Figure 6.30.

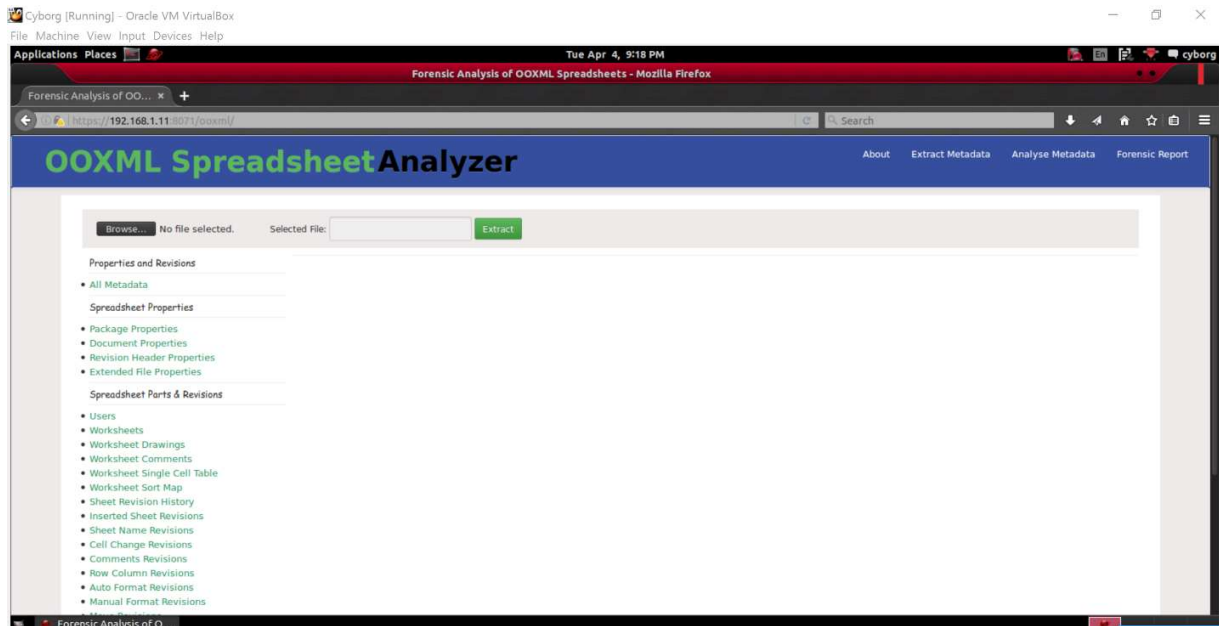


Figure 6.30: Application Accessed from Cyborg Hawk Virtual Machine

Vulnerability assessment was carried out using Vega 1.0 installed on the Windows 10 machine. A number of vulnerabilities were detected in the web application as shown in Figure 6.31.

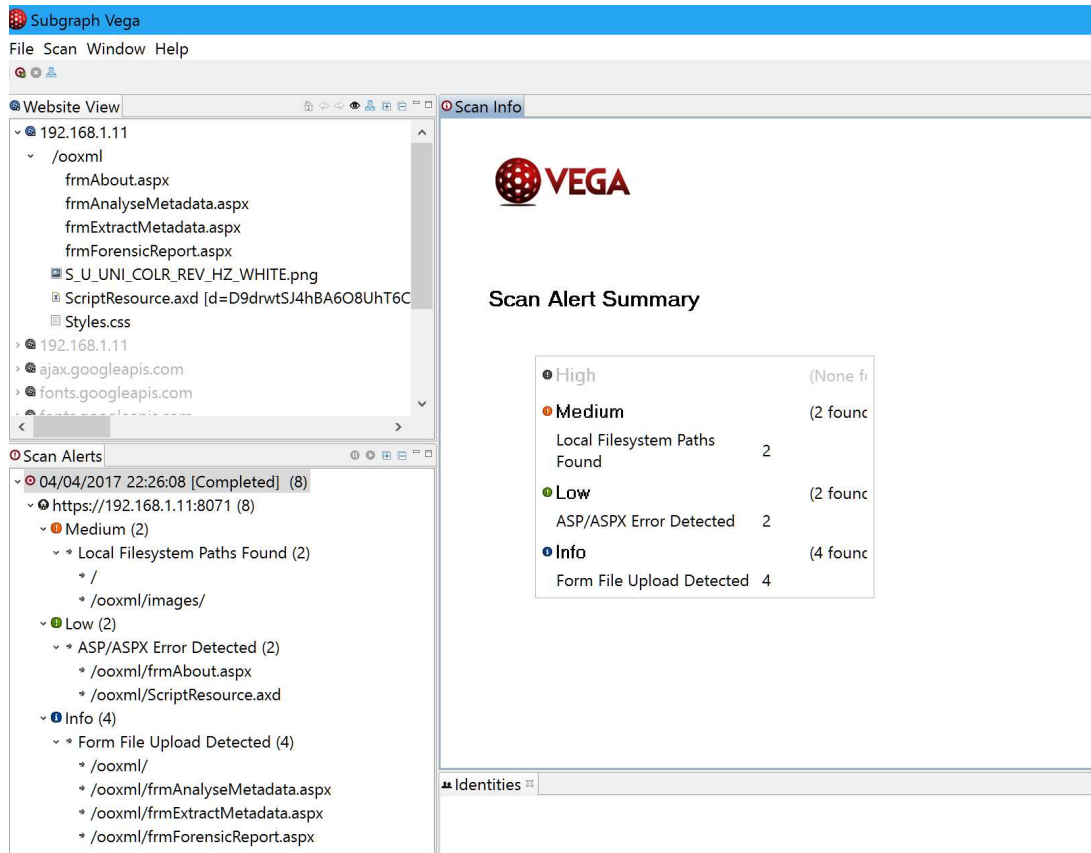


Figure 6.31: Web Application Vulnerabilities

These vulnerabilities are categorised and summarised below.

- Medium (2):
 - Local File System Paths Found. This allows a full URL of files to be exposed to clients.
- Low (2):
 - ASP/ASPX Error Detected. Possible verbose error messages can be exposed to remote clients hence giving an attacker an idea of the web server and application.

The above vulnerabilities were resolved by the following:

- Editing the source code to canonicalize the file paths to solve the page differential vulnerability and make sure absolute URLs for file paths are not sent to clients;
- Editing `Web.config` file to disable error messages for remote users by setting mode of `customErrors` flag to “RemoteOnly”.

Vulnerability assessment was again performed on the web application and the results show no more vulnerabilities existing in the web application as shown in Figure 6.32. There was therefore no need to continue with penetration testing in the absence of vulnerabilities.

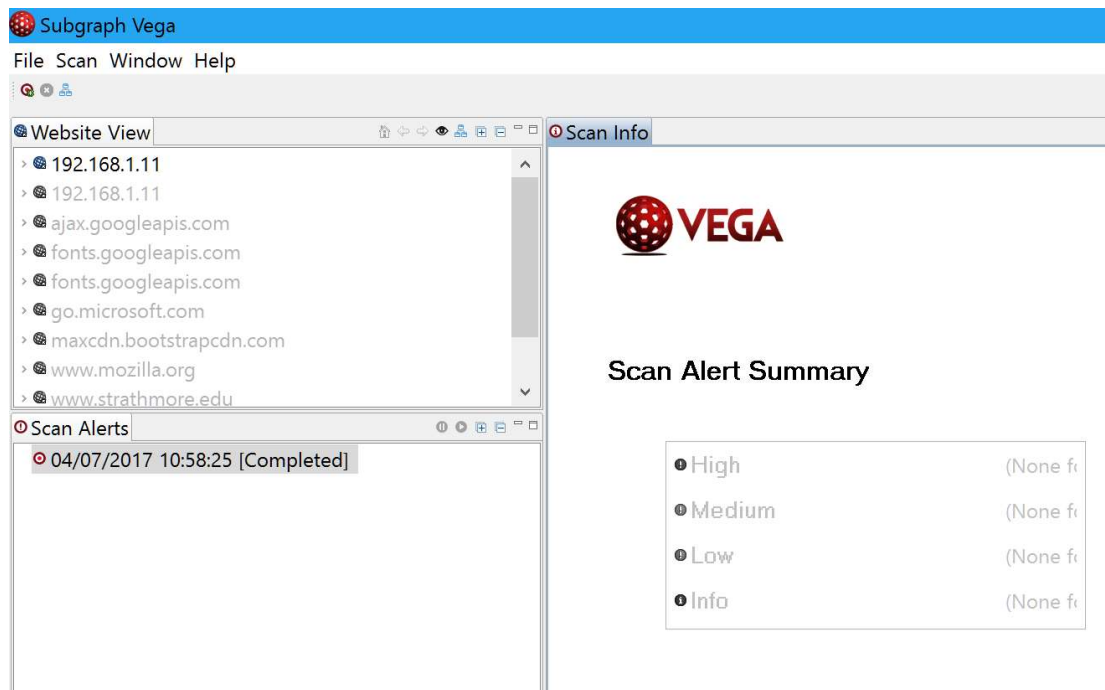


Figure 6.32: Web Application Vulnerabilities after Resolution

Quality Assurance

Results of the tests of the tool were compared against functional specifications to determine if the tool met the specifications and requirements. Below is a summary of the functional specifications and the tests results that correspond to these specifications.

Functionality:

- The tool accesses spreadsheet files in read only mode so as not to modify the files. This makes sure forensic evidence is preserved in the file without modification;
- The tool extracts metadata from OOXML spreadsheets including write protected files so long as the metadata exists in the files. This metadata include package, document, revision header and extended file properties; users, Worksheets, Worksheets comments and custom views. It is also able to extract metadata related to revisions in the Sheet and Cell levels of a spreadsheet document including Cell, Worksheet, comments, columns, rows, format, moved Sheets

revisions. This is proven in the implementation as shown in Figures 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8, 6.9, 6.10, 6.11 and 6.12;

- The tool is able to analyse OOXML spreadsheet metadata and output the results in a time series right from the time the file was created. This can be done for all users or for a selected user as demonstrated in the implementation in Figures 6.14, 6.15 and 6.16; and
- A forensic report consisting of the most useful metadata information is output in a readable format as shown in Figure 6.16 and Figures D1-1, D1-2, D1-3 and D1-4 in Appendix D1.

Usability:

- The tool is fully web based and is hosted remotely. The clients who access the system via URL do not have to install any additional software, library or plugin; and
- It has a user-friendly interface that can be operated by low skilled users since all application business processes and logic are hidden from the user.

Reliability:

- Currently a version of the system is hosted at Microsoft Azure cloud on trial basis. However, it could be hosted in any cloud that supports Microsoft IIS and can be accessed on demand.

Performance:

- Metadata extraction, analysis and reporting takes approximately 7 seconds each when files more than 12 MB are used.

Supportability:

- The application is developed as a Proof of Concept implementation and a lot of room has been left for future development to commercial version.

Security:

- OWASP secure coding standard is used to implement the tool. The locally hosted version of the application is secured using self-signed server certificate and it can be accessed via HTTPS; and
- Security tests performed on the application show very few vulnerabilities mainly exposure of full file URL to clients as in Figure 6.31. These vulnerabilities were addressed and the web application is secure as shown in vulnerability scan in Figure 6.32.

6.3. Validation

Validation of the Tool

Document Checks:

Functional specifications in Table 5.1; technical specifications in Table 5.2; and the design consisting of Use Cases in Figure 5.1 and Tables 5.3, 5.4, 5.5, 5.6, Class diagrams in Figures 5.2 and 5.3, and wireframes in Figures 5.4, 5.5, 5.6 and 5.7 and System Architecture in Figure 5.8 were scrutinized to ascertain that the specifications were designed correctly and that design is correct and implements the specifications as expected. Functional and technical specifications cover all required functionality of the tool. The Use Cases cover the required flows and functional procedures of the system. System architecture, Class Diagrams and wireframes cover the design of the tool so as to implement the functional specifications.

Source Code Review:

The Code Analysis feature of Microsoft Visual Studio was used to analyse source code for any incorrect implementation. Figure 6.33 shows the Code Analysis feature.

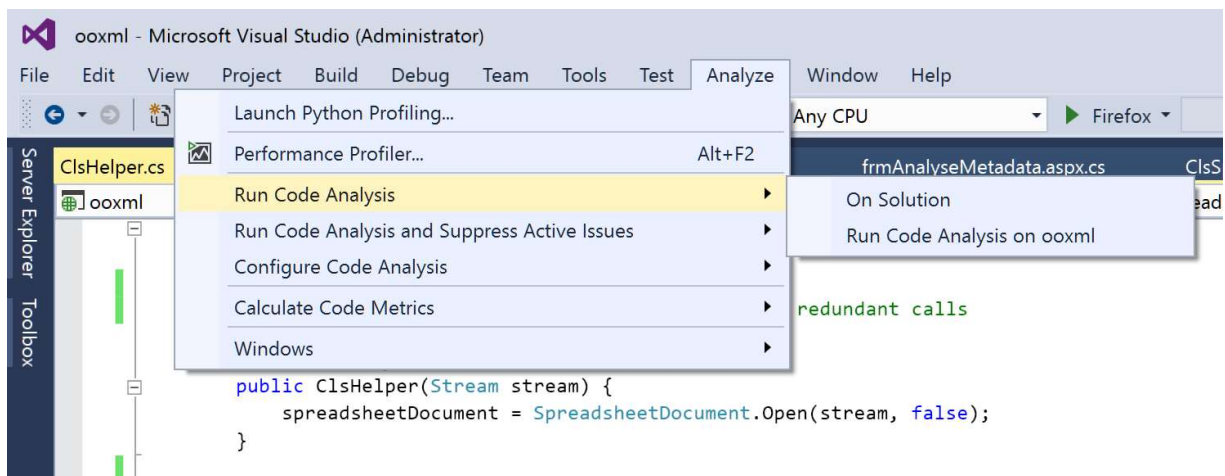


Figure 6.33: Code Analysis Feature of Microsoft Visual Studio

Figure 6.34 shows the results of this code analysis. The results show two warnings and no errors, which is evidence that the source code implementation is correct.

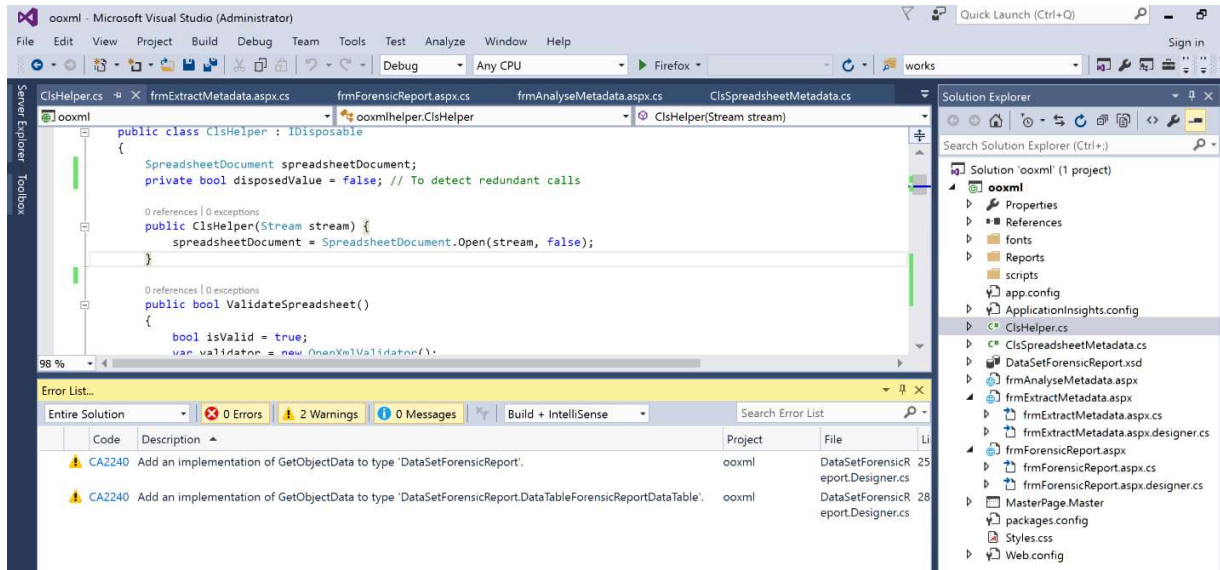


Figure 6.34: Code Analysis Results

Functional Completeness:

Tests were carried out on the tool using spreadsheets datasets of varied sizes and nature. These datasets also included write protected files. A large dataset of approximately 12MB was used to validate how the tool responds and how long it takes to extract and analyse metadata. The tool took about four seconds to extract metadata and five seconds to analyse metadata and 7 seconds to generate and display a forensic report. When write protected spreadsheet files were used, the tool was able to extract all metadata as summarised in Table 1A of Appendix A, similar to unprotected files.

Validation of Accuracy and Correctness of Results

Validation Plan:

A validation plan was developed elaborating the features/ functions and corresponding results to validate and the controlled datasets to be used. The results obtained by the tool were compared against expected results registered when manual changes are made to the files. Table 6.2 summarises the validation plan.

Table 6.2: Validation Plan

Item	Procedure and Expected Outcome
Features/ Functions Validated	Extraction and display of metadata from spreadsheets including write protected files. Changes and revisions are made to the spreadsheet datasets for Package Properties, Document Properties, File Properties, Users, Revision Header, Workbook, Worksheet, Sheet Revisions and Cell Revisions. It is expected that the tool is able to extract metadata related to these manual changes
	Analysis of spreadsheet metadata and display of analysed information. Relationship and revision identifiers should properly be linked together
	Generation and display of forensic report. Forensic report should consist of the most important analysed metadata information displayed in a readable manner.
Features/ Functions Not Validated	None
Datasets Used	Prepared controlled datasets of varying sizes including write protected files
Validation Environment	Locally hosted application
How many times a procedure is repeated	Three times

Validation of Results:

Results of the manual and functional tests were used to validate the accuracy and correctness of the results of the tool. Figures 6.18 and 6.19 show the results of guided end user testing that validates that the tool can extract and analyse metadata from OOXML spreadsheets. The tool can specifically extract metadata on document properties, sheet name changes, comments, cell revisions and print information from OOXML spreadsheets as shown in Figures 6.20, 6.21, 6.22, 6.23, 6.24, 6.25 and 6.26.

7. Discussion of Results

This chapter discusses the results of this research which include testing of existing forensic tools supporting OOXML spreadsheets, spreadsheet metadata extraction and analysis using the developed forensic tool, testing and validation. This research found that existing forensic tools supporting OOXML are still limited in metadata extraction, analysis and reporting on spreadsheet documents. The Proof of Concept (PoC) implementation of an improved forensic tool was able to address most of the limitations of the existing forensic tools and is of added value to forensic investigations of OOXML spreadsheets. This research also confirmed an earlier finding in the research by Didriksen (2014) that an OOXML spreadsheet created in Microsoft Office Excel, edited and saved in LibreOffice Calc will have its relationship and revision identifiers stripped by LibreOffice Calc and the numbering of these identifiers will be restarted.

7.1. Limitations of Existing Forensic Tools

Metadata exists in OOXML spreadsheets that is very useful for forensic investigators. Previous research however has dwelt majorly in metadata extraction from Office documents thus leaving a gap in forensic analysis of spreadsheets. Most of this metadata can only exist in shared spreadsheets. This means if a spreadsheet is created and never shared then only package, document and extended file properties can be extracted since all other types of metadata is not stored in non-shared spreadsheets.

Existing tools supporting OOXML were tested using controlled OOXML datasets of varying size and nature, including write protected files. It was found that most of these tools are not capable of extracting and analysing important metadata in spreadsheets, thus confirming the hypothesis that existing forensic tools are limited in adequately extracting and analysing all metadata in OOXML spreadsheets. They are limited only in extracting document properties which in many cases is not sufficient metadata to support forensic investigations since forensic investigators would like to get more information such as all revisions that have taken place in the spreadsheets. A few of the tested tools such as Metagoofil, Libextrator and OfficeDisector were capable of extracting metadata related to individual package parts such as comments, Workbook, Worksheet and relationships between different package parts and revisions done on the parts. However, their output is text format that is difficult to analyse. For example, Libextrator outputs a plain text format report that displays the corresponding XML files without giving much details on the content of these files.

Another limitation of these tools is that they are command line based. OfficeDissector requires a forensic investigator to have good Python command line skills and an understanding of the tool library itself in order to extract meaningful metadata. Most forensic investigators may not have these skills thus posing a challenge to the productive use of the tool. In addition, all the tools tested require installation in a particular OS and there may not be a version of the tool corresponding to a different OS. This limits a forensic investigator in that the investigator may not use a tool simply because the Operating System requirement limitation. For instance, an investigator running Windows machine cannot install a Linux based tool. The results further reveal that write protection using passwords of OOXML spreadsheets does not prevent metadata extraction and analysis since the tools were able to extract metadata from write protected files, thus negating the hypothesis that it is not possible to extract metadata from write protected OOXML spreadsheets.

7.2. Proof of Concept Implementation of Forensic Tool

A Proof of Concept implementation of a new OOXML forensic tool was developed so as to address the limitations identified in the testing of existing forensic tools supporting OOXML. This new tool can extract and analyse a lot more metadata than the tested existing tools, including package properties, document properties, extended file properties, revision header properties, users who have edited the document, Worksheets, Worksheet comments, Sheet revisions, comment revisions row and column revisions, cell revisions and custom view. This brings lots of advantage to an investigator in that a lot more evidence can be unearthed from a shared spreadsheet document. Analysis of extracted metadata is made easier by using relationship and revision identifiers present in OOXML spreadsheets. Data in very component of a spreadsheet including packages and parts are related to each other by relationship identifiers. For example, a cell is related to the parent Sheet by a relationship identifier and metadata for every revision done on the cell is also referenced in the parent Sheet by the relationship identifier. All revisions done on the spreadsheet have unique revision identifiers. The developed tool correlates relationship and revision identifiers sequentially so as to perform analysis on the chronology of events by different users as they took place in a spreadsheet.

7.3. Testing and Validation

Testing and validation of the Proof of Concept implementation reveals a functional prototype that is able to extract, analyse and present metadata in OOXML spreadsheets. Manual tests done by

different users show that the tool is able to extract and analyse metadata as expected. Functional tests reveal that even with large spreadsheet datasets of 12MB the tool running on localhost is able to extract and analyse metadata in approximately 6 seconds, which is fast enough for this purpose. In addition, the tests also prove that the tool can extract and analyse metadata from write protected files. Security tests performed reveal two medium level vulnerabilities related to local file system that might expose full file URL to remote clients, and two low level ASP.NET vulnerabilities related to verbose error output. These vulnerabilities were fixed and further vulnerability scan did not reveal any vulnerability. Quality assurance carried out on functionality, usability, reliability, performance, supportability and security reveal that the tool performs according the functional specifications and requirements. Validation of the tool includes document checks for documents such as Specifications, Use Cases, Class Diagrams and wireframes, source code review and functional completeness. Specifications were examined and found to be in line with end user requirements and that implementation of the specifications would achieve what the tool was intended to do. Use Cases, Class Diagrams and wireframes were examined and found to be in sync and correct according to the specifications, while source code review did not find any errors or wrong implementation in the source code. Validation of accuracy of results was also done and the results showed that the tool extracts and analyses metadata accurately and outputs a forensic report.

8. Conclusions, Recommendations and Future Work

8.1. Conclusions

Shared OOXML spreadsheets have a lot of metadata that can be very useful to forensic investigators including package properties, document properties, extended file properties, metadata related to Parts of the spreadsheet document and revisions. There are many existing forensic tools that support OOXML, but very few of them can extract meaningful metadata that can support a forensic investigation. These tools are not able to analyse metadata and present it in a user-friendly manner.

The Proof of Concept implementation of an improved forensic tool has proved that is possible to extract and analyse lots of metadata in OOXML spreadsheets as it addresses the limitations of the existing forensic tool. The tool is implemented in accordance with specifications of functionality, usability, reliability, performance, supportability and security. In addition, from results of the testing and validation of the tool it can be concluded that the application performs as is expected and is secure.

8.2. Recommendations

This research has come up with a Proof of Concept implementation of OOXML forensic tool that is capable of extracting, analysing and reporting metadata in OOXML spreadsheets. It can be useful for forensic investigators who wish to extract and analyse metadata in OOXML spreadsheets without the need to install any software, thus saving them a great deal of time and resources irrespective of their level of skill. This tool can also be used by auditors who may wish to track unauthorised changes to spreadsheets made by different users and hence be able to discover potential losses especially to financial institutions.

8.3. Future Work

This research has identified recommendations that can be incorporated in additional research in metadata extraction and analysis of OOXML spreadsheets in the context of digital forensics.

Developing a commercial version of the Proof of Concept implementation

This research focussed on developing a Proof of Concept implementation of a forensic tool that can extract and analyse metadata from OOXML spreadsheets. Future research should use the

findings of this research to develop a commercial version of the forensic tool. The commercial version should have a more thorough testing and validation as the scope of this research was limited to implementing the first working prototype of the forensic tool.

Extracting and analysing all possible metadata in OOXML spreadsheets

Not all metadata existing in OOXML spreadsheets can be extracted using the Proof of Concept implementation of forensic tool. This research concentrated on metadata as summarised in Table 1A of Appendix A. Future work should focus on improving the tool to be able to extract and analyse all metadata that can possibly exist in OOXML spreadsheets including imbedded objects such as images.

Investigating stripping of relationship and revision identifiers in LibreOffice Calc

Validation of accuracy of results revealed that revisions made on OOXML spreadsheets created by Microsoft Office Excel using LibreOffice Calc have the numbering of relationship and revision identifiers stripped and restarted, although the chronological order of this numbering is the same. Due to limited time, this research could not find out why this is the case. Future research should try to investigate why LibreOffice strips unique identifiers in OOXML spreadsheets when a file is edited and saved in LibreOffice Calc.

References

- AccessData. (2017). Forensic Toolkit (Version 5.5) [Windows Software]. AccessData. Retrieved from <http://accessdata.com/solutions/digital-forensics/forensic-toolkit-ftk?/solutions/digital-forensics/ftk>
- Aghire, I. (2012, June 10). Microsoft's Office Has over One Billion Users. Retrieved from <http://news.softpedia.com/news/Microsoft-s-Office-Has-Over-One-Billion-Users-280426.shtml>
- Ambler, S. (2014a). Examining the Agile Manifesto. *Ambyssoft Inc.* Retrieved from <http://www.ambyssoft.com/essays/agileManifesto.html>
- Ambler, S. (2014b). UML 2 Class Diagrams: An Agile Introduction. *Ambyssoft Inc.* Retrieved from <http://www.agilemodeling.com/artifacts/classDiagram.htm>
- Bajpai, P. (2014, November 30). FOCA Metadata Analysis Tool [Security Journal]. Retrieved 19 December 2016, from <http://lifeofpentester.blogspot.com/2014/11/foca-metadata-analysis-tool.html>
- Bechtsoudis, A. (2011, May 2). Gathering & Analysing Metadata Information. *Gathering & Analyzing Metadata Information*. Retrieved 2 June 2017, from <http://bechtsoudis.com/archive/2011/05/02/gathering-analyzing-metadata-information/index.html>
- Brunty, J. (2011). Validation of Forensic Tools and Software: A Quick Guide for the Digital Forensic Examiner. Retrieved from <http://www.forensicmag.com/article/2011/03/validation-forensic-tools-and-software-quick-guide-digital-forensic-examiner>
- Castiglione, A., D'Alessio, B., De Santis, A., & Palmieri, F. (2012). New Steganographic Techniques for the OOXML File Format. Retrieved from <http://dl.ifip.org/db/conf/IEEEEares/murpbs2011/CastiglioneDSP11.pdf>
- Chema, A., Rando, E., Oca, F., & Guzman, E. (2008). Disclosing Private Information from Metadata, hidden info and lost data. Retrieved from

- https://www.blackhat.com/presentations/bh-europe-09/Alonso_Rando/Blackhat-Europe-09-Alonso-Rando-Fingerprinting-networks-metadata-whitepaper.pdf
- Dennis, A., Wixom, B., & Roth, R. (2012). *System Analysis and Design* (5th ed.). United States of America: John Wiley & Sons, Inc. Retrieved from http://www.saigontech.edu.vn/faculty/huynq/SAD/Systems_Analysis_Design_UML_5th%20ed.pdf
- Didriksen, E. (2014). *Forensic Analysis of OOXML Documents* (Master's Thesis). Department of Computer Science and Media Technology, Gjøvik University College, Norway. Retrieved from <https://brage.bibsys.no/xmlui/bitstream/id/211829/EDidriksen.pdf>
- Document Foundation Wiki. (2016, September 9). The Document Foundation, LibreOffice and OOXML. In *Wiki*. Document Foundation. Retrieved from https://wiki.documentfoundation.org/LibreOffice_OOXML
- ECMA International. (2006a, June). C # Language Specification. ECMA International. Retrieved from www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf
- ECMA International. (2006b, December). Office Open XML File Formats (Standard ECMA-376). ECMA International. Retrieved from <http://web.mit.edu/~stevenj/www/ECMA-376-new-merged.pdf>
- Eleven Paths. (2015). *FOCA (Fingerprinting Organizations with Collected Archives)* [Computer Software]. English. Retrieved from <https://www.elevenpaths.com/labstools/foca/index.html>
- Eoghan, C. (2011). *Digital Evidence and Computer Crime - Forensic Science, Computers and the Internet* (Third Edition). Academic Press. Retrieved from http://booksite.elsevier.com/samplechapters/9780123742681/Front_Matter.pdf
- Free Software Foundation. (2016). GNU Libextractor (Version 1.1) [Linux Software]. Retrieved from <https://www.gnu.org/software/libextractor/>
- Garfinkel, S. (2009). New XML-based Files Implications for Forensics. *ResearchGate*. <https://doi.org/10.1109/MSP.2009.44>

- GNU. (2008, November). The GNU libextractor Reference Manual [Reference Manual]. Retrieved 6 February 2017, from <https://www.gnu.org/software/libextractor/manual/libextractor.html>
- Grier Forensics. (2015). OfficeDissector (Version 1.0) [Python Toolkit]. Grier Forensics. Retrieved from <https://www.officedissector.com>
- Gudjonsson, K. (2009, October 7). Office 2007 Metadata [Blog]. Retrieved 11 November 2016, from <https://digital-forensics.sans.org/blog/2009/07/10/office-2007-metadata/>
- ISO/IEC. (2012, 2016). ISO/IEC 29500-1:2016, ISO/IEC 29500-2:2012, ISO/IEC 29500-3:2016, ISO/IEC 29500-2:2016 - Office Open XML File Formats [ISO/IEC Standards]. Retrieved 26 September 2016, from <http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html>
- Jacobson, I., Spence, I., & Bittner, K. (2011). *Use Case 2.0*. Retrieved from https://www.ivarjacobson.com/sites/default/files/field_iji_file/article/use-case_2_0_jan11.pdf
- jQuery Foundation. (2017). JQuery [Documentation]. Retrieved 10 February 2017, from <https://jquery.com/>
- Kabay, M. E. (2002). Salami Fraud. Norwich University. Retrieved from http://www.mekabay.com/nwss/116p--salami_fraud.pdf
- Kumar, M. (2012, April 15). Forensic FOCA - Power of Metadata in digital forensics. *The Hacker News*. Retrieved 2 June 2017, from <http://thehackernews.com/2012/04/forensic-foca-power-of-metadata-in.html>
- Langweg, H. (2012). An OOXML File Analysis of the Terrorist Manual Related to the 22/7 Attacks. Retrieved from <http://dl.ifip.org/db/conf/cms/cms2012/Langweg12.pdf>.
- LibreOffice. (2016a). What is LibreOffice [Documentation]. Retrieved 3 December 2016, from <https://www.libreoffice.org/discover/libreoffice/>
- LibreOffice. (2016b). What is OpenDocument? [Documentation]. Retrieved 3 December 2016, from <https://www.libreoffice.org/discover/what-is-opendocument/>

- LibreOffice, B. (2014, June). Does Libre Office fully support microsoft .xlsx and .docx file formats? [LibreOffice Blog]. Retrieved 12 February 2016, from <https://ask.libreoffice.org/en/question/36148/does-libre-office-fully-support-microsoft-xlsx-and-docx-file-formats/>
- Microsoft. (2006, May). Introducing the Office (2007) Open XML File Formats [Documentation]. Retrieved 2 February 2017, from [https://msdn.microsoft.com/en-us/library/aa338205\(v=office.12\).aspx](https://msdn.microsoft.com/en-us/library/aa338205(v=office.12).aspx)
- Microsoft. (2011, April 7). Office Open XML, ECMA-376, and ISO/IEC 29500. Retrieved 12 February 2026, from https://msdn.microsoft.com/en-us/library/office/gg607163.aspx#IIOXML_H2
- Microsoft. (2017a). IIS Overview [Documentation]. Retrieved 10 February 2017, from <https://www.iis.net/overview>
- Microsoft. (2017b). *Microsoft Azure* [ASP.NET Cloud Hosting]. English, Microsoft. Retrieved from <https://azure.microsoft.com/en-us/?b=17.06>
- Microsoft. (2017c). Open XML SDK 2.5 for Microsoft Office [Documentation]. Retrieved 10 February 2017, from <https://www.microsoft.com/en-us/download/details.aspx?id=30425>
- Microsoft, C. (2017d). DocumentFormat.OpenXml.Spreadsheet Class Library Reference [Open XML SDK 2.5 Class Library Reference]. Retrieved 26 September 2016, from <https://msdn.microsoft.com/en-us/library/office/documentformat.openxml.spreadsheet.aspx>
- Microsoft, C. (2017e). Open XML Formats and file name extensions [Article]. Retrieved 3 December 2016, from <https://support.office.com/en-us/article/Open-XML-Formats-and-file-name-extensions-5200d93c-3449-4380-8e11-31ef14555b18>
- Muhamad, A. R. (2011, March). *DATA HIDING AND DETECTION IN OFFICE OPEN XML (OOXML) DOCUMENTS* (Master's Thesis). University of Ontario Institute of Technology, Oshawa, Ontario, Canada. Retrieved from https://ir.library.utoronto.ca/bitstream/10155/146/1/Raffay_Muhammad.pdf
- Neudesic, LLC. (2015, January 27). Open Source ASP.NET MVC, Web API, and Web Pages. Retrieved 4 June 2017, from <https://www.asp.net/open-source>

- NISO. (2004). Understanding Metadata. In *Understanding Metadata*. Bethesda, MD 20814 USA: National Information Standards Press. Retrieved from <http://www.niso.org/publications/press/UnderstandingMetadata.pdf>
- NIST. (2015, August). Computer Forensics Tool Testing (CFTT) Project Web Site [Computer Forensics Tool Testing Program]. Retrieved 26 February 2017, from <http://www.cftt.nist.gov/>
- OWASP. (2016, March). OWASP Developer Guide [Software Developer Guide]. Retrieved 23 March 2017, from https://www.owasp.org/index.php/OWASP_Guide_Project
- OWASP. (2017, February). OWASP Testing Guide [Penetration Testing Guide]. Retrieved 23 March 2017, from https://www.owasp.org/index.php/OWASP_Testing_Project
- Pallmer, G. (2001). A Road Map for Digital Forensic Research - First Digital Forensic Research Workshop. In *The Digital Forensic Research Conference 2001, USA*. USA: AFRL/IFGB, Utica, New York. Retrieved from https://www.dfrws.org/sites/default/files/session-files/a_road_map_for_digital_forensic_research.pdf
- Popik, B. (2010, October 14). Salami Slicing (Salami Technique; Salami Attack). Retrieved from http://www.barrypopik.com/index.php/new_york_city/entry/salami_slicing_salami_technique_salami_attack
- Python Software Foundation. (2016). Python-OOXML 0.13 (Version 0.13) [Python Library]. Python Software Foundation. Retrieved from <https://pypi.python.org/pypi/Python-OOXML>
- Schicht, J. (2011). Advanced OOXML (zip) carving/ recovery (Version 4) [Linux Software]. Retrieved from <http://www.forensicfocus.com/Forums/viewtopic/t=7814/>
- WHATWG Community. (2017, June 2). HTML Standard. Retrieved 4 June 2017, from <https://html.spec.whatwg.org/multipage/interaction.html#editing-2>
- Wiley. (2016). What is JavaScript. Wiley. Retrieved from http://media.wiley.com/product_data/excerpt/88/07645790/0764579088.pdf

Zhangjie, F., Xingming, S., Yuling, L., & Li, B. (2011). Forensic investigation of OOXML format documents. *Elsevier Ltd*. Retrieved from <http://fulltext.study/preview/pdf/457873.pdf>

Appendix A Metadata Present in OOXML Spreadsheets

The OOXML standard has a lot of metadata can be possibly present in an OOXML spreadsheet as shown in Table A1.

Table A1 OOXML Spreadsheet Metadata (Microsoft, 2017d)

Property/ Description	Metadata
Package Properties	Creator, Title, Subject, Category, Content Type, Content Status, Identifier, Description, Revision, Version, Created, Modified, Last Printed
Document Properties	Auto Save, Compression Option, Document Type, File Open Access
Document Header Properties	Has Disk Revisions, Maintains Revision History, Last GIUD, Revision Id, Track Revisions, Shared
Extended File Properties	Application, Application Version, Company, Digital Signature, Document Security, Heading Pairs, Hyperlink List, Hyperlinks Changed, Shared Document
Users	Date Time, GIUD, User Id, Name
Worksheet Comments	GIUD, Author Id, Reference, Comment Text
Worksheet Revision Header	Date Time, GIUD, Relationship Id, Maximum Revision Id, Maximum Sheet Id, Minimum Revision Id, Sheet Id Map Count, Username
Workbook Revision Log	Revision Id, Sheet Id, Revision Action, Revision Reference
Cell Revisions	Revision Id, Sheet Id, Style Revision, Old Cell, New Cell, Extension List, Format, Has Phonetic Text, Number Format Id, Old Differential Format, Old Formatting, Old Phonetic Text, Row Column Formatting Affected
Comment Revisions	Sheet Id, Author, Cell, GIUD, Old Length, New Length, Action
Inserted Sheets	Revision Id, Sheet Id, Sheet Position, Name
Sheet Revisions	Revision Id, Sheet Id, Extension List, New Name, Old Name
Moved Cells	Revision Id, Sheet Id, Source, Source Sheet Id, Destination
Format Revisions	Sheet Id, Length, Row or Column Affected
Revision Conflicts	Revision Id, Sheet Id, Inner Text
Relationships	Relationship metadata between parts

Table A2 SpreadsheetML Components (ECMA International, 2006b)

Part	Relationship Target of	Root Element
Calculation Chain	Workbook	calcChain
Chartsheet	Workbook	chartsheet
Comments	Chartsheet, Dialogsheet, Worksheet	comments
Connections	Workbook	connections

Part	Relationship Target of	Root Element
Custom Property	Workbook	Not applicable
Custom XML Mappings	Workbook	mapInfo
Dialogsheet	Workbook	dialogSheet
Drawings	Chartsheet, Worksheet	wsDr
External Workbook References	Workbook	externalReference
Metadata	Workbook	metadata
Pivot Table	Worksheet	pivotTableDefinition
Pivot Table Cache Definition	Pivot Table, Workbook	pivotCacheDefinition
Pivot Table Cache Records	Pivot Table Cache Definition	pivotCacheRecords
Query Table	Worksheet	queryTable
Shared String Table	Workbook	sst
Shared Workbook Revision Headers	Workbook	headers
Shared Workbook Revision Log	Shared Workbook Revision Headers	revisions
Shared Workbook User Data	Workbook	users
Single Cell Table Definitions	Dialogsheet, Worksheet	singleCells
Styles	Workbook	styleSheet
Table Definition	Dialogsheet, Worksheet	table
Volatile Dependencies	Workbook	volTypes
Workbook	SpreadsheetML package	workbook
Worksheet	Workbook	worksheet

Appendix B Minimum Parts and Relationships Requirements for a Workbook

Appendix B1 Content Types for Relationship Parts, Workbook Part and Sheet Part

The content types for relationship parts, the Workbook part, and at least one Sheet part must be defined (physically located at `/[Content_Types].xml` in the package) as shown in Figure B1 (ECMA International, 2006b, p. 60).

```
<Types xmlns="...">
  <Default Extension="rels"
    ContentType="application/vnd.openxmlformats-package.relationships+xml"
  />
  <Override PartName="/workbook.xml"
    ContentType="application/vnd.openxmlformats-officedocument.
spreadsheetml.sheet.main+xml" />
  <Override PartName="/sheet1.xml"
    ContentType="application/vnd.openxmlformats-
officedocument.spreadsheetml.worksheet+xml" />
</Types>
```

Figure B1-1: Content_Types.xml Representation (ECMA International, 2006b)

Appendix B2 Package-Level Relationship to Workbook Part

The required package-level relationship to the Workbook part must be defined (physically located at `/_rels/.rels` in the package) as shown in Figure B2 (ECMA International, 2006b, p. 60).

```
<Relationships xmlns="...">
  <Relationship Id="rId1"
    Type="http://schemas.openxmlformats.org/officeDocument/2006/
relationships/officeDocument"
    Target="workbook.xml" />
</Relationships>
```

Figure B2-1: Workbook Part Package-Level Relationship (ECMA International, 2006b)

Appendix B3 Minimum Content for Workbook

The minimum content for the Workbook part must be defined (physically located at `/workbook.xml` in the package) as shown in Figure B3 (ECMA International, 2006b, p. 61).

```

<workbook xmlns="" xmlns:r="">
  <sheets>
    <sheet name="1" sheetId="1" r:id="rId1" />
  </sheets>
</workbook>

```

Figure B3-1: Minimum Content for Workbook Part (ECMA International, 2006b)

Appendix B4 Workbook-Level Relationship to a Single Sheet

The required workbook-level relationship to the single Sheet part must be defined, (physically located at `/_rels/workbook.xml.rels` in the package) as shown in Figure B4 (ECMA International, 2006b, p. 61).

```

<Relationships xmlns="">
  <Relationship Id="rId1"
    Type="http://schemas.openxmlformats.org/officeDocument/2006/
relationships/worksheet" Target="sheet1.xml" />
</Relationships>

```

Figure B4-1: Workbook-Level Relationship to a Single Sheet Part (ECMA International, 2006b)

Appendix B5 Minimum Content for a Single Sheet Part

The minimum content for a single Sheet part must be defined (physically located at `/sheet1.xml` in the package) as shown in Figure B5 (ECMA International, 2006b, p. 61).

```

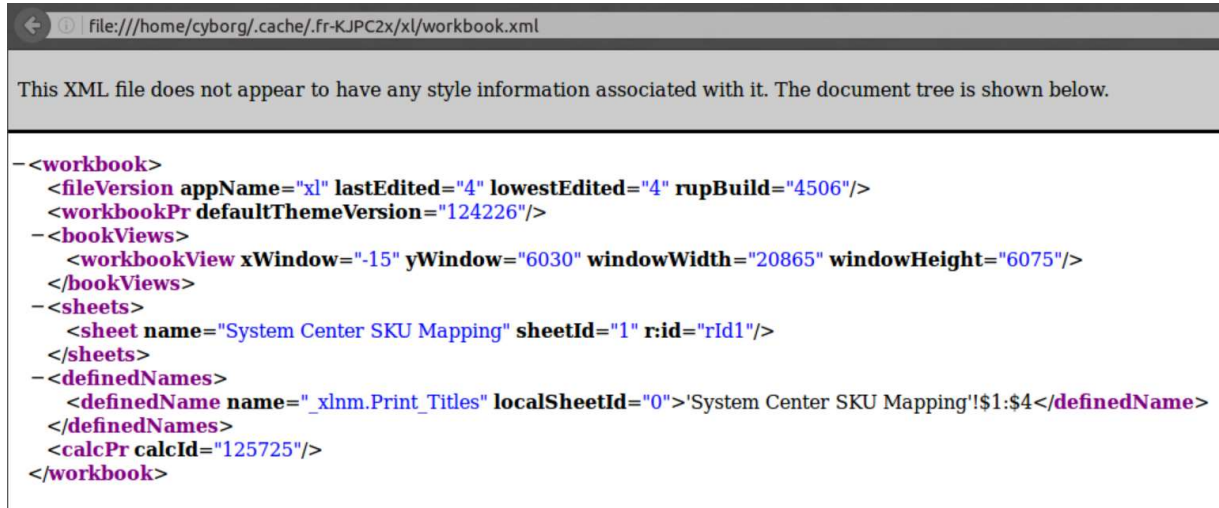
<worksheet xmlns="" xmlns:r="">
  <sheetData />
</worksheet>

```

Figure B5-1: Minimum Content for a Single Sheet Part (ECMA International, 2006b)

Appendix C Testing of Existing Forensic Tools

Appendix C1 Extracting Metadata from Workbook.xml Using Metagoofil 2.2

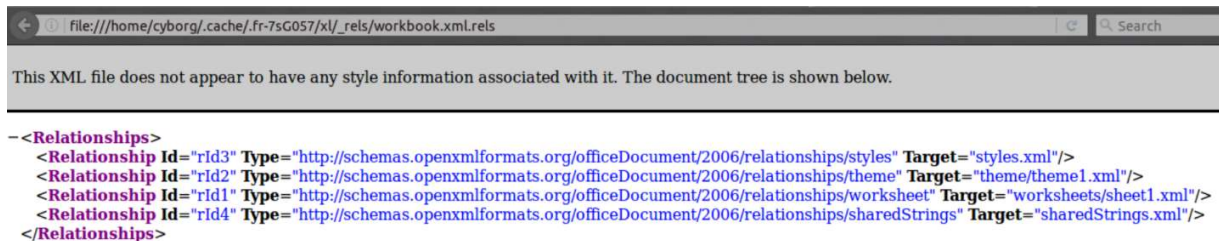


The screenshot shows a web browser window with the address bar containing the file path: file:///home/cyborg/.cache/.fr-KJPC2x/xl/workbook.xml. Below the address bar, a message states: "This XML file does not appear to have any style information associated with it. The document tree is shown below." The XML document tree is displayed as follows:

```
--<workbook>
  <fileVersion appName="xl" lastEdited="4" lowestEdited="4" rupBuild="4506"/>
  <workbookPr defaultThemeVersion="124226"/>
  <bookViews>
    <workbookView xWindow="-15" yWindow="6030" windowWidth="20865" windowHeight="6075"/>
  </bookViews>
  <sheets>
    <sheet name="System Center SKU Mapping" sheetId="1" r:id="rId1"/>
  </sheets>
  <definedNames>
    <definedName name="_xlnm.Print_Titles" localSheetId="0">'System Center SKU Mapping'!$1:$4</definedName>
  </definedNames>
  <calcPr calcId="125725"/>
</workbook>
```

Figure C0.1-1: Metadata Extraction Results in Workbook.xml

Appendix C2 Extracting Metadata from Workbook.xml.rels Using Metagoofil 2.2



The screenshot shows a web browser window with the address bar containing the file path: file:///home/cyborg/.cache/.fr-7sG057/xl/_rels/workbook.xml.rels. Below the address bar, a message states: "This XML file does not appear to have any style information associated with it. The document tree is shown below." The XML document tree is displayed as follows:

```
--<Relationships>
  <Relationship Id="rId3" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/styles" Target="styles.xml"/>
  <Relationship Id="rId2" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/theme" Target="theme/theme1.xml"/>
  <Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/worksheet" Target="worksheets/sheet1.xml"/>
  <Relationship Id="rId4" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/sharedStrings" Target="sharedStrings.xml"/>
</Relationships>
```

Figure C2-1: Metadata Extraction Results in Workbook.xml.rels

Appendix C3 Extracting Metadata from Worksheet.xml Using Metagoofil 2.2

Figure C3-1: Metadata Extraction Results in Worksheet.xml

Appendix C4 Extracting Metadata from Worksheet.xml.rels Using Metagoofil 2.2

Figure C4-1: Metadata Extraction Results in Worksheet.xml.rels

94

Appendix D System Implementation, Testing and Validation

Appendix D1 Implementation of Forensic Report

Case Number: Investigator:

No file selected. Selected File:

Forensic Report

1 of 2 ? Find Next	
MetadataType	Package Properties
Creator	user
Title	
Subject	
Revision	
Version	
Created	10/15/2016 5:54:35 PM
Modified	3/28/2017 9:08:30 PM
LastModifiedBy	David Godiah
LastPrinted	
Keywords	
Identifier	

Figure D1-1: Package Properties Part of Forensic Report

Case Number: Investigator:

No file selected. Selected File:

Forensic Report

Content type	
Description	
MetadataType	Document Properties
AutoSave	true
CompressionOption	Normal
DocumentType	Workbook
FileOpenAccess	Read

Figure D1-2: Document Properties Part of Forensic Report

Case Number: Investigator:

No file selected. Selected File:

Forensic Report

Users	
DateTime	2016-10-26T21:57:43
RelationshipId	rld8
UserId	-429770265
Guid	{CC7278B7-0E6F-4B22-B229-9587872F6043}
Name	David Godiah
LocalName	userInfo
DateTime	2017-02-28T12:30:36
RelationshipId	rld8
UserId	-429768026
Guid	{CC7278B7-0E6F-4B22-B229-9587872F6043}
Name	David Godiah
LocalName	userInfo
DateTime	2017-02-28T12:31:12

Figure D1-3: Users Part of Forensic Report

Case Number: 001 Investigator: David Godiah

Browse... No file selected. Selected File: Book1.xlsx Report

Forensic Report

5 of 6 ? Find Next	
MetadataType	Shared Revision Cell Change
RelationshipId	rld14
RevisionId	11
SheetId	3
StyleRevision	0
OldCell	
NewCell	56
InnerText	56
UserName	user
DateTime	2016-10-26T20:28:00
InnerXml	<x:nc r="B1" xmlns:x="http://schemas.openxmlformats.org/spreadsheetml/2006/main"><x:v>56</x:v></x:nc>

Figure D1-4: Cell Revisions Part of Forensic Report

Appendix D2 Manual Test Cases Template

Table D2-1 Manual Test Cases Template

Test Case ID	Test Case and Procedure	Expected Results	Actual Results	Pass/Fail
1	<p>Test Name: Metadata Extraction</p> <p>Test Procedure:</p> <ul style="list-style-type: none"> ▪ Click “Extract Metadata” link ▪ Browser OOXML file by clicking on “Browser” button and select the file ▪ Click “Extract” button 	<ul style="list-style-type: none"> ▪ Metadata extraction page is displayed ▪ Selected file name is displayed in “Selected File” text box ▪ Metadata is displayed in tabular format segregated into properties and revisions ▪ If an invalid file is selected an error message is displayed requesting user to select correct file 		

Test Case ID	Test Case and Procedure	Expected Results	Actual Results	Pass/Fail
2	<p>Test Name: Metadata Analysis</p> <p>Test Procedure:</p> <ul style="list-style-type: none"> ▪ Click “Analyse Metadata” link ▪ Browser OOXML file by clicking on “Browse” button and select the file ▪ Click “Analyse” button ▪ Optionally select a user from “User” dropdown menu 	<ul style="list-style-type: none"> ▪ Metadata analysis page is displayed ▪ Selected file name is displayed in “Selected File” text box ▪ Analysed metadata information is displayed in a time series manner by default for all users ▪ If an invalid file is selected an error message is displayed requesting user to select correct file ▪ If a user is selected, only analysed metadata specific to that user is displayed 		
3	<p>Test Name: Forensic Report</p> <p>Test Procedure:</p> <ul style="list-style-type: none"> ▪ Click “Forensic Report” link ▪ Input case number and name of investigator ▪ Browser OOXML file by clicking on “Browse” button and select the file ▪ Click “Report” button ▪ Navigate pages of the report by clicking on navigation buttons on the report header ▪ Save report in Excel/PDF/Word format by clicking on “Save” button on the report header 	<ul style="list-style-type: none"> ▪ Forensic report page is displayed ▪ Selected file is displayed in “Selected File” text box ▪ Forensic report is displayed with header containing title, file name, case number, name of investigator, report date and metadata information ▪ Different pages of the report are shown depending on which navigation button is clicked ▪ Report is saved in Excel/PDF/Word format 		
Tested By:			Date Tested:	