Application of Reverse Engineering Technique in Software Forensic Analysis to Detect Infringements

Bassey Asuquo Ekanem and **Jacob Meye**

Abstract — The importance of reverse engineering in engineering and related disciplines cannot be overemphasized. In computer forensics, reverse engineering is widely applied in the areas of digital, mobile and memory forensics except in software forensics where automated tools are lacking but undertaken manually by experts. Software forensics are usually conducted on source codes to find evidence for legal proceedings. Unfortunately, low correlation scores indicating no infringement case are usually reported from genuine cases where the infringer had exhaustively reworked the source code to beat forensic analysis. Findings from this research indicate that application of reverse engineering in software forensics could reveal the modifications made by the infringer to produce the alleged infringing software thereby empowering the prosecuting team to prove the case beyond reasonable doubts in courts, hence highly recommended for automation.

Index Terms — source code, software forensics analysis, correlation scores, infringement pattern, reverse engineering

I. INTRODUCTION

The continuous rise in source code theft and IP infringement has threatened the global software industry leading to financial loss to criminals up to about $500 billion per annum. To recover from such loses, developers must prosecute infringers. However, proving the ownership of such stolen software and its similarity is usually difficult as the infringer may have reworked the software to make it appear completely different (Frantzeskou, 2018).

With software forensic analysis, evidence could be obtained from the alleged infringing software when the codes are compared for correlations. The objective of software forensic analysis is to find evidence for legal proceedings by examining the literal expression and the functionality of software Zeidman (2014). Software forensic is a branch of Forensic Science which involves the use of scientific or technical approaches to identify, collect, analyse, and interpret evidence to support legal proceedings (Arshad et al., 2018).

Manuscript received September 26, 2020; revised May 5, 2021. This work was supported in part by the Tertiary Education Trust Fund (TETFund), Nigeria under Research Grant provision.

Bassey Asuquo Ekanem is a Chief lecturer with Delta State Polytechnic Ozoro, Nigeria where he teaches software engineering and related courses. Phone number: +2348026905532; email address: basseyekanem99@gmail.com.

Jacob Meye is a Chief Lecturer and Researcher with Delta State Polytechnic, Ozoro. He has participated as a team member in many software engineering projects undertaken in the Institution. Email address: dr.ma.jacob@gmail.com.

In most cases, forensic analysis from genuine infringement cases usually result in low correlation scores which points to no infringement case especially when the infringement has exhaustively reworked the source code making it appear different. With this most genuine cases are usually dropped after forensic analysis when it appears that such low correlation scores will not yield fruitful results in proving the case in courts. Some experts usually take a step further by applying reverse engineering in the forensic analysis undertaken manually in attempt to gather additional evidence need to prove the case.

Cipresso and Stamp (2010) define software reverse engineering as the process of analyzing a software system to understand it as well as extract its design and implementation information. In computer forensics, reverse engineering is widely applied using automated tools like IDA-pro, Sandboxie, Die and many others in the areas of digital, mobile and memory forensics but not in software forensics since automated tools for this are lacking (Hendricks, 2020). In digital and mobile forensics, it is used to retrieve evidence from computer and mobile devices respectively by unearthing the actions performed by the infringers whereas in memory forensics, it is used to analyze volatile memory to unearth actions performed by malwares in order to resolve attacks

In view of the above, this research work was undertaken to examine the importance of reverse engineering into software forensics and how it could be incorporated into existing software forensic applications to enhance the quality evidence generated for litigation processes.

II. REVIEW OF RELETED WORKS

In Fakhar (2019), findings from a review on source code infringements cases is presented showing continuous rise in source code plagiarism and litigation cases in courts and emphasizes the need for enhanced tools to effectively deal with the situation. Also, a review of the impact of software forensics tools indicate that they have come a long way in terms of design, development, upgrading and enhancement of applications used for forensic analysis which include CodeSuite, Autopsy, EnCase and others (Upcounsel, 2020; Comodo, 2018).

Also, in Stim (2017) the use of these tools in forensic analysis has been widely reported with successes and failures experienced particularly with cases won in lower courts but upturned at the upper courts due to technical reasons not adequately addressed by the forensic analysis.

In Chien and Lin (2016), and GreB Services (2020) copyright registration issues and dishonest testimonies by defendants' team in courts are presented as additional factors responsible for loss of genuine source code infringement cases in courts.

The need for enhanced computer forensic methods and techniques capable of providing valid and reliable evidences to prove cases beyond reasonable doubts is emphasized (Venčkauskas et al, 2015). Whereas in Ekanem (2015) reengineering of legacy software used in modern applications is recommended to incorporate enhanced features. Kilinc (2015) identifies renaming of identifiers, functions and parameters; adding and removal of comment lines, and restructuring of code blocks as some of key characteristics of stolen source codes and recommends a methodology for calculating code similarity ratio based on N-gram similarity, Vector Space Model (VSM) and Cosine Normalization (CN) methods to detect such.

In Comodo (2020), application of reverse engineering in computer forensics is reported as being widely applied in in digital, mobile and memory forensics whereas lacking automated tools to apply it in software forensics, hence undertaken manually by experts. Where it is applied, its capable of revealing the infringement made by the infringer to make the source code appear different.

III. FINDINGS FROM THE REVIEW

The review reveals that reverse engineering techniques and tools as applied in forensic science are mostly in areas of digital, mobile and memory forensics whereas tools and techniques for its application in software forensic are lacking. If it were possible for investigators to perform reverse code analysis on alleged stolen software to uncover the exact changes that were made by the infringer to produce the alleged infringing software, it will be a lot easier for the prosecuting team to prove their case in court. It will help the team to demonstrate in court convincingly how the infringer modified the original software to produce the stolen copy. But this is not the possible, as automated tools for these important process are lacking.

This is the focus of this research to consider how reverse engineering could be effectively applied in software forensic analysis to gather evidence for source code infringement litigations.

IV. MATERIALS AND METHODS

The research work was designed as experimental research with CodeSuite software used for the experiment and analysis. The data used for the experiment is the original source code of a result management software used in a higher institution. Ten source files out of 47 were selected for the exercise and coded as Case Software 1. Also, the source code of the ten selected files were modified to produce another copy of the software for the purpose of this research which is coded Case Software 2.

The methods used in the research are outlined in the following processes.

i. ***Review of Relevant Literature.***
Research works and relevant literature were reviewed to keep abreast with existing body of knowledge in the area and gaps to fill.

ii. ***Pre-process stage***
This involves acquisition and installation of CodeSuite software selected as the software forensic application for the research. It was selected because of its ease of use and being one of the best software forensic applications.

iii. ***Acquisition and Preservation stage***
This involves the acquisition and preparation of data needed for the analysis. In this case, Case Software 1 and Case Software 2 were acquired.

iv. ***Code Analysis Stage***
This involved the following activities:
a) checking for source file identity using the **FileIdentity** program in CodeSuite.
b) Comparing Case Software 1 with Case Software 2 for correlations and possible infringements using **CodeSimilarity** and **Codematch** programs in CodeSuite Application.
c) code reverse engineering using static method where the source codes were examined side-by-side to detect identifiers and other program elements in Case Software 1 that were replaced in Code Software 2. Static (manual) method was used since there are no reverse engineering tools for software forensics.

v. ***Presentation Stage***
This involves interpretation, presentation and documentation of analysis results in a way that contributes to the body of knowledge in Software Forensic Analysis and success in IP infringement litigations.

vi. ***Post-process Stage***
This has to do the proper closing of the investigation exercise (research) with proper preservation of research data, documentation and publishing of research finding.

TABLE I: Selected Source Files in Result Management Software (i.e. Case Software 1)

S/n	Filename	Function of the file	Lines of Code
1	userLogin.vb	For user login process	19
2	Admin.vb	Admin functions by the administrator	16
3	lecturerRecords.vb	Management of lecturers' records	28
4	studentRecords.vb	Management of students' records	39
5	cgpaCompute.vb	Computation of CGPA	101
6	courseMgt.vb	Management of courses	67
7	resultsMgt.vb	Management of results	86
8	scoresUpload.vb	Uploading of raw scores	45
9	resultsPreview.vb	Preview of results	39
10	recordsSearch.vb	Searching of information in the system	63

TABLE II: Source Files in Case Software 2 and Modifications effected to create the file

S/n	Filename	Lines of Code	Type of Modifications made
1	userLogin.vb	28	i) Renaming of identifiers ii) Restructuring statements sequence and code blocks
2	Admin.vb	26	ii) Modifying comment lines ii) Renaming of identifiers
3	lecturerRecords.vb	41	i) Renaming of identifiers ii) Modifying comment lines iii) restructuring code blocks
4	studentRecords.vb	52	i) Renaming of identifiers ii) Restructuring statements sequence and code blocks
5	cgpaCompute.vb	135	i) Renaming of identifiers ii) Modifying comment lines
6	courseMgt.vb	74	i) Renaming of identifiers ii) Restructuring statements sequence and code blocks iii) Restructuring of code
7	resultsMgt.vb	114	i) Renaming of identifiers ii) Modifying comment lines iii) restructuring code blocks
8	scoresUpload.vb	63	i) Renaming of identifiers ii) Modifying comment lines ii) restructuring statement sequence and code blocks
9	resultsPreview.vb	51	i) Renaming of identifiers ii) Modifying comment lines iii) restructuring code blocks
10	recordsSearch.vb	68	i) Renaming of identifiers ii) Modifying comment lines

V. ANALYSIS AND RESULTS

Forensic analysis was conducted on the Case Software 1 and Case Software 2 using CodeSuite following the steps below:

i) File identity analysis was performed which identified the code files as Visual Basic program files with extension .vb as shown in tables I and II.

ii) Filtering out of non-protectable elements (standard keyword, procedures and methods etc.).as always the case in source code infringement cases.

iii) File similarity analysis was performed using CodeDiff program of CodeSuite which shows the % similarity of files in both software as given in table III.

iv) Codematch Analysis was performed using CodeSuite to detect correlations in terms of matching statements, comments & strings, instruction sequence and identifiers. The results are also presented in tables III.

TABLE III: Forensic Analysis Results after Standard Elements were Filtered Out from Software Case Software 2

s/n	Filename	Similarity - CodeDiff (%)	Correlation - CodeMatch
1	userLogin.vb Admin.vb	4	3
2	lecturerRecords.vb	2	1
3	studentRecords.vb	3	2
4	cgpaCompute.vb	4	2
5	courseMgt.vb	2	2
6	resultsMgt.vb	3	4
7	scoresUpload.vb	1	3
8	resultsPreview.vb	2	2
9	recordsSearch.vb	2	4
10	userLogin.vb	3	5

Analysis also indicate none matching statement, comments & Strings, instruction sequence and identifiers except for some partial matching identifiers like cmdLI**tem** matching Sys**tem,** Parame**ter**s matching ebMa**ter**ial and ebwri**ter;** and others which are quite negligible**.**

v) Static Reverse Engineering (Manual)

Following the low correlation results as presented in the above tables, static reverse analysis was performed following the steps below:

a) line-by-line comparison of the source codes in Case software 1 and Case Software 2 to discover elements in Case Software 2 that were used to replace program elements in Case Software 1.

b) By using Find and Replace tool in a Code Editor (in this case ATOM), identified program elements in (a) above that were replaced in Case Software 2 were correspondingly reversed using the corresponding elements in Case Software 1. The reverse engineering code obtained through this process is coded as Case Software 3. The code editor is used because, reverse engineering features are not available in existing software forensic applications.

c) Inspection (Manual) of the source codes in Case Software 1 and Case Software 3 was performed. This was done by side-by-side review and comparison of the code for similarities.

vi) Repeat Forensic Analysis

Forensic analysis was repeated in this case on Case Software 1 and Case Software 3 using CodeSuite. The result of the analysis is presented in table IV.

TABLE IV: Forensic Analysis Results on Case Software 1 and Case Software 3 after Static Reverse Engineering was performed

s/n	Filename	Similarity - CodeDiff (%)	Correlation - CodeMatch
1	userLogin.vb Admin.vb	100	100
2	lecturerRecords.vb	100	100
3	studentRecords.vb	100	100
4	cgpaCompute.vb	100	100
5	courseMgt.vb	100	100
6	resultsMgt.vb	100	100
7	scoresUpload.vb	100	100
8	resultsPreview.vb	100	100
9	recordsSearch.vb	100	100
10	userLogin.vb	100	100

vi) The correlation results were examined and interpreted accordingly to determine their relevance in proving alleged infringement.

vii) Research findings were documentation and distributed accordingly.

VI. DISCUSSIONS

The low correlation scores from forensic analysis as shown in Tables III and IV is misleading as they point to no source code infringement case whereas the code were actually copied from Case Software 1. However, following the application of Code Reverse engineering to identify the replaced elements and correspondingly reversing them, a repeat of the forensic analysis resulted in high (100%) correlation scores which confirm source code infringement.
In effect, application of reverse engineering in software forensic as demonstrated in this research has clearly confirm the importance of reverse engineering in software forensics. It further emphasizes the need to incorporate code reverse engineering features in software forensic applications to assist investigators in revealing the modifications made by infringers to create alleged infringing software. Algorithms for this process could be designed and developed into code reverse engineering programs to be incorporated into existing software forensic software to fully automate the process.

VII. CONCLUSION

The importance of reverse engineering in software forensics cannot be overemphasized. It is capable of revealing the changes that were made by the infringer to produce the stolen software. With this tool, irrespective of the efforts by infringers to rework the source code and make the stolen software appear different, the changes will be uncovered and used as evidence for litigations

VIII. RECOMMENDATIONS

The following recommendations are made based on the research findings:

i. Application of reverse engineering in software forensics is highly recommended as a process to enhance the quality of forensic analysis reports.
ii. Research efforts towards designing algorithms needed to apply reverse engineering in software forensics and developing such algorithms into computer programs to be incorporated into existing software forensic applications are highly recommended.

ACKNOWLEDGMENT

B. A. Ekanem thanks the Chairman of Software Analysis and Forensic Engineering (SAFE) Corporation for guidance and provision of free certification course on use of CodeSuite to the researchers as well as free monthly license to use the software for the research. Prof. Akpodiete, O. J., the Rector of Delta State Polytechnic, Ozoro is also knowledge for creating the enabling environment for the research to thrive.

REFERENCES

[1] G. Frantzeskou, S. Gritzalis, and S. G. MacDonell, Source Code authorship analysis for supporting the cybercrime investigation Process, 2005 Available: https://pdfs.semanticscholar.org
[2] B. Zeidman, Software Forensics: Objectively Proving Infringement or Misappropriation; IP WatchDog 2014. Available: https://www.ipwatchdog.com/2014/10/27/software-forensics-objectively-proving-infringement-or-Misappropriation/id=51825/
[3] H. Arshad, A. B. Jantan, and O. I. Abiodun, Digital Forensics: Review of Issues in Scientific Validation of Digital Evidence; Journal of Information Processing Systems; 14(2), 346-376, 2018. https://www.researchgate.net/publication/327644306_Digital_Forensics_Review_of_Issues_in_Scientific_Validation_of_Digital_Evidence
[4] T. Cipresso, and M. Stamp, Software Reverse Engineering. In: Stavroulakis P., Stamp M. (eds) Handbook of Information and Communication Security. Springer, Berlin, Heidelberg. pp 659-696, 2010. Available: https://doi.org/10.1007/978-3-642-04117-4_31
[5] B. Hendricks, Reverse Engineering in Digital Forensics; Chapter 4- Lesson 9; Study.com, 2020. Available: https://study.com/academy/lesson/reverse-engineering-in-digital-forensics.html
[6] M. Fakhar, Computer Forensics: Overview of Software Forensics, 2019. Available: https://resources.infosecinstitute.com/category/computerforensics/introduction/areas-of- study/application-forensics/overview-of-software-forensics
[7] Upcounsel.com, Software Forensics: Everything You Need to Know, 2020. Available https://www.upcounsel.com/software-forensics
[8] Comodo, What is Forensic Analysis? Comodo Group, 2018. Available: https://enterprise.comodo.com/blog/what-is-forensic-analysis/
[9] R. Stim, When Someone Steals Your Copyright Code or Software, 2017. Available: https://www.nolo.com/legal-encyclopedia/how-do-you-know-if-you-have-valid-claim- someone-stealing.html
[10] H. Chien, and E. Lin, Copyright Infringement Issues concerning adaptations of computer software, Lee and Li Attorney at Law. 2016. Available https://www.lexology.com/library/detail.aspx?g=3802ceca-6595-447d-93c4-c80d978ccb38
[11] GreB Services, 14 Famous Patent Infringement Cases that changed US Patent Law; Greb Services, 2020. Available: https://www.greyb.com/famous-patent-infringement-cases/
[12] A. Venčkauskas, J. Toldinas, S. Grigaliunas, R. Damasevicius, and V. Jusas, Suitability of the digital forensic tools for investigation of cyber crime in the Internet of Things and Services; The 3rd International Virtual Research Conference in Technical Disciplines (RCTD) 2015. Available: https://www.researchgate.net/publication/299104454_Suitability_of_the_digital_forensic_tools_for_investigation_of_cyber_crime_in_the_Internet_of_Things_and_Services
[13] B. A. Ekanem, Assessment of Component Stability for Modernization Using Software Maturity Index, International Journal of Scientific Research and Engineering Studies (IJSRES) 2(12), 2015. Available: www.ijsres.com
[14] D. Kilinc, F. Bozyiğit, A. Kut, and M. Kaya, Overview of Source Code Plagiarism in Programming Course; International Journal of Soft Computing and Engineering (IJSCE); ISSN:2231-2307, volume 5(2), 2015.

Author: Dr. Bassey Asuquo Ekanem is a holder of B. Sc. And M. Sc. Degrees in Computer Science. He also obtained Ph. D. degree in Engineering and Technology Management in 2017 with research interest in Software Engineering – software components reusability. He is a member of Computer Professionals of Nigeria (CPN), Nigeria Computer Society (NCS) and IEEE. Ekanem is a certified Software Forensic expert using CodeSuite Application. He is a lecturer with Delta State Polytechnic Ozoro teaching software engineering and related courses. He has published manner articles in the areas of software engineering with particular note of "Legacy Components Stability Assessment and Ranking Using Software Maturity Index", Dealing with Components Reusability Issues as Cutting-edge Applications Turn Legacy and many others.

Author: Dr. Jacob Meye is a Chief Lecturer and Researcher in Delta State Polytechnic, Ozoro. Meye has undergone series of trainings in software development and forensic analysis which prompted his interest as a researcher in the area of software forensics. He has participated as a team member in many software development and implementation projects where he garnered practical experience in software development.