NOVATEUR PUBLICATIONS
INTERNATIONAL JOURNAL OF INNOVATIONS IN ENGINEERING RESEARCH AND TECHNOLOGY [IJIERT]
ISSN: 2394-3696
VOLUME 7, ISSUE 3, Mar.-2020

APPROACH TO COMPUTER SECURITY VIA BINARY ANALYTICS

RAVI TEJA YARLAGADDA
DevOps SME & Department of Information Technologyy,USA
yarlagaddaraviteja58@gmail.com

A**BSTRACT**
Currently, software organizations face critical issues associated with computer security, considering software developments contain mistakes and errors (Abel, 2018). These errors develop conditions that favor the emergence of serious vulnerabilities in computers. The exploitation of vulnerabilities impacts the security of the system that uses vulnerable products and the organization's activities as a whole. Many attacks on computers have been directed to software companies, and they are based on preliminary disassembly and the assessment of the protection of the binary code mechanism. There are various types of binary analysis which operate with source code via manual, automated assessment. The technique can show the precise area of vulnerability in a computer or a software code.

KEYWORDS : Computer Security, Binary Analytics, Information Technology, Malware and Data Storage

INTRODUCTION

When considering malware diversity in this century, many computer systems are at significant threat of security breaches (Benson, 2019). Such threats have surpassed the ability of malware analysts to detect them. Therefore, to detect the big malware, the amount of data needs to be assessed to determine the potential threats. The typical techniques cannot comply with the high demand of malware writers who utilize different methods and strategies such as obfuscation and polymorphism to outwork detection systems. Normal approaches depend on the signature detection whereby the file's signature is assessed against known malicious ones. The utilization of such approaches is less demanding and susceptive to simple evasive methods. Typical detection methods depend on dynamic and static assessment. Static analysis is perceived to be computationally effective and safe because it does not need the execution of codes to be analyzed. However, it is easily impacted by evasive techniques. Moreover, dynamic assessment has greater chances of sensing undetected samples but significantly impacts the system's performance considering the file is assessed in a virtual environment. The research article discusses a new approach to computer security via binary analytics.

When utilizing binary analysis, it is important to understand both the control-flow structure and information flow of a big codebase (Abel, 2018). Many disassemblers focus on recovering the control-flow structure and various research systems. There are not sufficient automated methods that can be used to understand the flow of data between data structures and the association between code and data structures. Therefore, there is a need for a ne

w technique that would be used to combine information-flow assessment with the recovery of the control-flow graph. The technique would be utilized to scale clear binary assessment to large software systems and use to generate polymorphic assaults against programs that sustain complex input transformations.

Computer applications have become larger and even more complex because of increased functionality which provides a more sophisticated software stack with new concepts and abstractions that simplify development (Abel, 2018). Such applications have become difficult to debug, and they have very complex vulnerabilities. When assessing vulnerability, a faulty program's location may be discovered after a certain input is passed via several buffers. Symbolic may seem a good tool to assess an application's security properties. However, this technique does not provide good scaling to greater contexts because of the explosions of various possible paths being evaluated. Another specialist proposes the utilization of fuzz testing that utilizes crash logs to find vulnerabilities. However, because the technique has probabilistic input, it is not able to reach the vulnerability guarded by tough low-probability conditions. In that case, this paper proposes a better approach to computer security through binary analysis. This paper would provide a critical observation that the root cause for many security problems as the major key to their solutions which directly lie in the relevant program such as

malicious codes and vulnerable programs. Therefore through the creation of tools and techniques that automatically extract interrelated security properties from device solutions and programs, it would be possible to use a principled approach to various issues. The approach would focus on the root cause and provide more effective solutions than the previous approaches that depend on symptoms or heuristics of an attack.

The workability of the approach has one defy that ensures the security of computer applications does not need to include binary code (Benson, 2019). There are many programs such as COTS where users do not access their source code, and when malicious code is encountered, hackers will fail to connect source code with the perceived attack. In order to be safe, binary code needs to be executed. Therefore, assessing the binary code would only provide a ground truth that is critical for security applications. In contrast, assessing source code may provide misguided results because of the optimizations and compiler errors. However, assessing the binary code is typically challenging because of the involved complexity and lack of critical semantic information. Therefore, few tools exist for binary analyses powerful enough for general security applications. This has become a big hurdle that prevents security specialists from undertaking the approach above to determine the root cause. This observation builds a unified binary assessment, and it can be utilized to give solutions to a wide spectrum of various security issues. This paper would provide and discuss the development of an extensive binary assessment infrastructure to be used to be security in computer security

LITERATURE REVIEW

Binary analysis is utilized for security and threat assessment, and it helps developers assess the application's raw binaries that involve complete applications helping to assess code security (Zolotarev, 2017). Binary analysis is utilized to assess libraries integrated into a computer. It performs a manual assessment to find vulnerable patterns when running deep vine assessment. Binary analytics assess and detect various security issues, hence identifying its source. There are many computer systems vulnerabilities because of integrators such as source code, plugins, and components used to lower development. Moreover, current enterprises use a software development process with open-source code that exposes organizations to many security issues because of incorrect code analysis. This shows the need for binary analytics to perform security analysis. Binary analysis is widely referred to as effective threat assessment testing and vulnerability. It assesses raw binaries that have a complete application. Since the analysis helps evaluate and expose binary code, a computer can be audited without coder cooperation. Before developing binary analytics, malware detection had been a persistent challenge for security specialists.

Traditional attempts are concentrated on dynamic and static analysis, but research specialists compelled the development and evolution of malicious code to derive an analysis to find solutions (Cascarino, 2017). The researcher has already done many works that focus on creating frameworks for analysis getting static features. Since the 1990s, there have been various research interests on binary analysis. The BitBlaze project is among the pioneers of such research. The researchers have been researching binary analysis works such as instrumentation, disassembly, and symbolic execution used by the active user community. In the current years, the growth of cyberinfrastructure has ignited a strong interest in computer and software security, resulting in a significant surge of research activities in the binary assessment. Researchers have identified and acknowledge Fuzzing as a well-known technique used to find software vulnerability by semi-randomly generating inputs to assess software and computer programs. This has been the main focus by various researchers from Korea, Singapore, and China. Carnegie Mellon University pioneered the research of automatically locating and exploiting various vulnerabilities of computers and programs that have analysis technics such as Fuzzing and symbolic execution. The strategy employed by the university was aimed at finding vulnerabilities and loopholes in a computer program. Regardless of researchers finding Binary analysis to be an effective method, it faces various challenges because of cybersecurity responders' and researchers' ever-elevated demands.

The dramatic improvement of binary analysis tasks needs frameworks that lower human efforts and improve productivity that can be scaled to achieve the actual world demands (Choi, Park, & Chung, 2018). According to the common abstractions utilized in the binary analysis, such as instruction semantics, intermediate

representation, control flow, and data flow, researchers have aimed to develop automatic methods used to optimize analysis processes, generate abstractions, and ensure assessment by various tools to operate. Various researchers have made many efforts from the Asian region that has the aim to develop solid building blocks for mountable binary assessment frameworks. For instance, writing instruction specifications from scratch is greatly an error-prone job. Instruction manuals that have been set comprise a thousand pages with descriptions written in natural language. When writing binary analysis that translates binary code IR, there is a need to read manuals carefully and devise the logic. In this process, tremendous engineering effort is required, and researchers perceive it to be costly to investigate. In the year 2019, KAIST researchers developed a general binary analysis framework which was an attempt to create a binary assessment framework. The research focused on optimizing performance and guaranteeing the accuracy of binary analysis. The researchers showed various optimization techniques that include big-integer splitting and parallel lifting, which achieved a certain improvement in the performance. These researchers were motivated by the growth of vulnerabilities in computer and software security issues. These issues caused delicate problems such as data loss and software crashes.

Moreover, software engineers have been facing many bug-related issues, and the cost to contain such threats is huge (Edmonds & Palmore, 2017). Therefore the researches were conducted to find intuitive ways of finding vulnerabilities in computers and program binary through the combination of various techniques such as grey-box Fuzzing and black box through symbolic execution approaches. In these techniques, the program logic is not assessed, whereby the approaches of symbolic execution precede using program semantic analysis behavior. Creating an executable program consists of the translation of source code. Object code signifies the original instructions but in the form of binary essential for CPU execution. It is also important to understand that every operating system defines the format where binary data is stored. According to various researchers, at the center of the reverse engineering process is the assessment of binary files used to govern the source program's functioning. Reverse engineering is divided to two opposite but distinct activities, namely dynamic and static analysis, critical in binary coding. Dynamic assessment techniques consist of code execution and program behavior monitoring used to determine the target program's functionality when optionally reconnoitering for susceptibilities. Static techniques consist of the binary files analysis without the implementation and need disassembly of the target software to determine functionality.

In another research, instruction set architecture is discussed and defined as the machine language inferred by a computing machine (Zhu and Liu, 2018). High-level languages are amassed into supported binary formats for the specialized operating systems in the instruction set architecture. The assembling of an executable program results in the disappearance of meaningful information originating from source code, and it is rejected as redundant in subsequent machine code. This involves information such as variable and function names. This results in complexity when undertaking reverse engineering activities as this evidence must be determined by a reverse engineer in its absenteeism. Another research was conducted, and it was found the creation of accurate binary program representation via static disassembly is a problem not yet to be solved. Two involved prevailing techniques are existing when code analysis and static disassembly is undertaken, they include recursive traversal and linear sweep. Linear sweep starts during the first executable code byte and begins disassembling. On the other hand, recursive traversal starts at the first byte; however, it follows the program control flow instead of stripping linearly. This is undertaken by assessing any control flow instructions such as jump or call defining the instruction's conceivable position. The positions are stripped, and any flow of control instruction is contained and analyzed; this process reiterates recursive until the binary is entirely assessed. Each of these approaches causes its unique complications such that linear sweep cannot account for the intermixed data with instructions that result in data disassembly as executable code and hence questionable results.

Recursive traversal meets hitches when categorizing the appropriate target position of control flow commands and may miscue executable code sections (Edmonds & Palmore, 2017). The disassembly process is complicated by new techniques that remain to emerge to disrupt the disassembly process, whether by malicious actors or security researchers. A certain study projected a new procedure used to obscure the binary source's disassembly

procedure. Such an approach utilizes the ability to give arbitrary values within a NOP instruction, and it takes advantage of adjustable length property. This would allow for supplementary execution paths inside a single program, the main execution path that depends on the external environment, such as the incidence of virtualization software. Through the utilization of Hidden execution, path disassembly can generate incorrect results and cause an analyst to make incorrect conclusions and assumptions. On the other hand, the Binary file format represents PE format whereby the disassembler starts parsing executable instructions rather than data. The distribution of malware prevalence uses the PE file format because of the prominence of OS's Microsoft Windows. In order to grow their efficiency, malware authors introduce malformations that permit a program to be executed by an operating system. However, it complicates the dynamic and static assessment tools utilized for detection and the detection of program functionality. Other malware specialist uses obfuscation or encryption to hinder the assessment process, and this can be important when assessing system.

BINARY ANALYSIS FRAMEWORK

Specialists developed binary analyzing frameworks to streamline the process of undertaking binary analysis, and they show standard features in performance, output, and support to end-user (Graham, 2017). Specialists developed the binary frameworks to address various limitations discussed earlier in this paper. Specialists designed the framework to address limitations exhibited by frameworks such as PEV that provide final output disassembly. The core interaction with such framework involves assessing the output that gives context whereby analyst springs program meaning. Researches projected the Binary assessment framework to address various core limitations. Data persistence architecture is created through an open-source database system. This design's projected advantage involves reduced time in undertaking program analysis, which is a design used to group related binaries and novel research styles according to disassembly output. Creating a relational database to resolve data permits the program to load data from already disassembled programs; this contrast with the performance of file input disassembly whenever a program assessment is desired. When correlated input files are assembled in a project concept, it increases the efficiency of analysis associated with input files.

Specialists can analyze them from the outstanding graphical user interface (Izumida, Futatsugi & Mori, 2018). The construct of the current frameworks needs an instance where an analysis program is opened in every file to enhance time requirement when switched between these instances. Another feature of the binary analysis framework critical in computer security is the search of architecture that permits the search through disassembly output. Therefore, this would create innovative search algorithms using string evaluation functions, database language, and regular expression features. Moreover, these searches are undertaken concurrently across numerous input files. Therefore, this permits an analyst to carry out a comparative assessment with disassembly output rather than depending on byte-sequence matching. This would ultimately increase the scope whereby an analyst assesses a computer program by providing a broader search. This step would build upon a limitation restricted to executing binary transfiguration of input search patterns before searching. This construct is that a more comprehensive analysis of the target input file is undertaken, which allows unknown vulnerabilities to be discovered hence securing the software and computer.

LIMITATION OF THE FRAMEWORK

The binary analysis framework would undertake an optimistic approach to validate input files (Jackson, 2018). Malware experts abuse PE file format to bypass detection and undertake other despicable activity on the computer system. The framework would not detect any intentional malformations, which may make it hard to parse the desired file or produce unfitting disassembly programs. This problem has been considered, and the framework would support the future expansion of more vigorous file authentication. Code obfuscation is a method that is employed in executable code to make it hard for the inner workings of software challenging to understand. It comes in many forms and can involve dynamic code generation and packing and the utilization of shellcode and encryption routing. Code obfuscation is utilized to achieve various goals, such as ensuring intellectual property is safe within the software or inhibiting the discovery of software vulnerabilities that is

leveraged to exploit computer systems. Malware experts also utilize code obfuscation to hide a particular program's functionality and escape signature-based detection. Despite the aim, such techniques cause the software to be obfuscated, disassembling complicated because the disassembly encounters may be obfuscated or encrypted data. In that case, de-obfuscated is needed to disassemble the code appropriately. It is essential to understand that the binary assessment framework would not support obfuscated disassembly or encrypted code because of such difficulties.

CODE ORGANIZATION

The binary analysis framework is in form of java application that uses JavaFX and SE. The selection of the two javas is aimed at supporting the development of the framework (Jane, 2019). JavaSE gives the core functionality of its programming language, and it defines basic objects and types of Java programming language used for security, networking, graphical user interface, and database access. JavaFX provides core libraries for the development of graphical user interfaces. The selection of java was considered due to its portability.

OVERVIEW OF THE FRAMEWORK COMPONENTS OF BINARY ANALYSIS

The framework will contain four major components: disassembly engine, input file validation, graphical user interface, and data storage design (Jiang, Feng, & Tang, 2018). The analysis process will start with the parsing and loading of the intended software. Initially, the binary analysis framework supports 32-bit binaries in the format file of PE. The interaction will begin when an input file is selected and analyzed by the framework (Joseph, 2019). If the framework supports the file format, it would be analyzed further to confirm whether it conforms to its specifications, such as the PE file format. Because the focus of this paper is on PE file format, the inputted file would be assessed in order to determine the validity of the 32-bit. If the framework detects an invalid PE file, it would not be put to the next stage of disassembly, and hence the analysis would stop. After a valid PE file is determined, the framework would then direct the input file for the proceeding state, the disassembling and parsing engine. The outcome of the stage would incorporate the file features via parsing of the code segment's files and documentation. After finding the code section, they are disassembled. The output is placed in a relational database given by the database architecture, which will then be used to show data in the graphical user interface that allows assessment of the imported file.

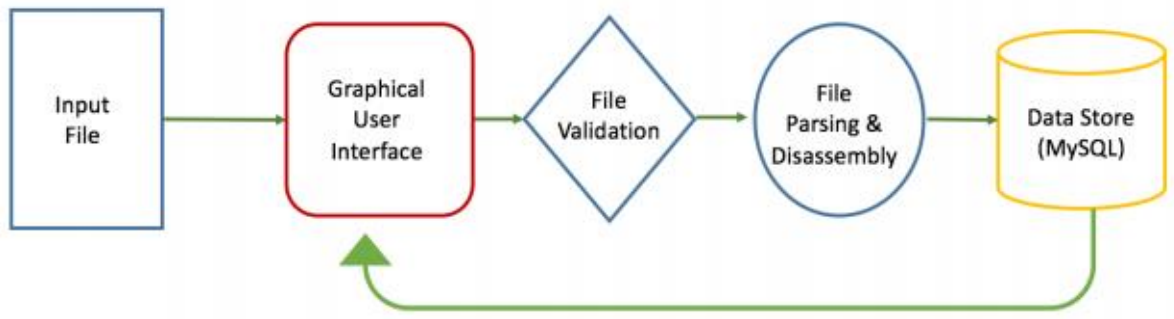

Figure 1: Framework components of binary analysis

After the disassembly and parsing have been successful, the data will then be directed to data storage. The framework of binary analysis will process the disassembly engine results to match the data store schema. This data will be taken to storage in the relational database through MySQL. The data model schema would be crafted to permit granular inspection for each imported file's individual instructions. The needed results for this architectural design are classified as twofold that allow novel search techniques developed using relation database and evade the prerequisite to disassemble files inputted every time binary analysis is executed. Moreover, the binary analysis framework will permit construction projects where related software is grouped in a logical collection. This would ensure increased efficiency in exploring the imported software and instruction-

level evaluation of the imported files in a project. The database model would be used to develop the functionality of framework search and use the disassembly output instead of performing binary searches on the machine code. Therefore this would enable high-level abstraction when assessing software as it searches for operands and assembly instructions rather than byte or binary sequences.

GRAPHICAL USER INTERFACE

The preliminary interaction with binary assessment starts with a graphical user interface (GUI) (Liu, Tan & Chen, 2018). When the application begins, the user is presented with the user interface shown below. The graphical interface is created using FXML documents defined within JavaFX standard, and they are based upon XML language, and they provide user interface components. It is desirable to provide abstraction or decoupling in software development to ensure the ease at which the software is maintained and developed. To start using binary analysis, the user creates or loads a program. In order to accomplish project management, file menu options are expanded, and the user then accesses a dialog listing that has been previously developed projects. The following diagram shows the dialog.

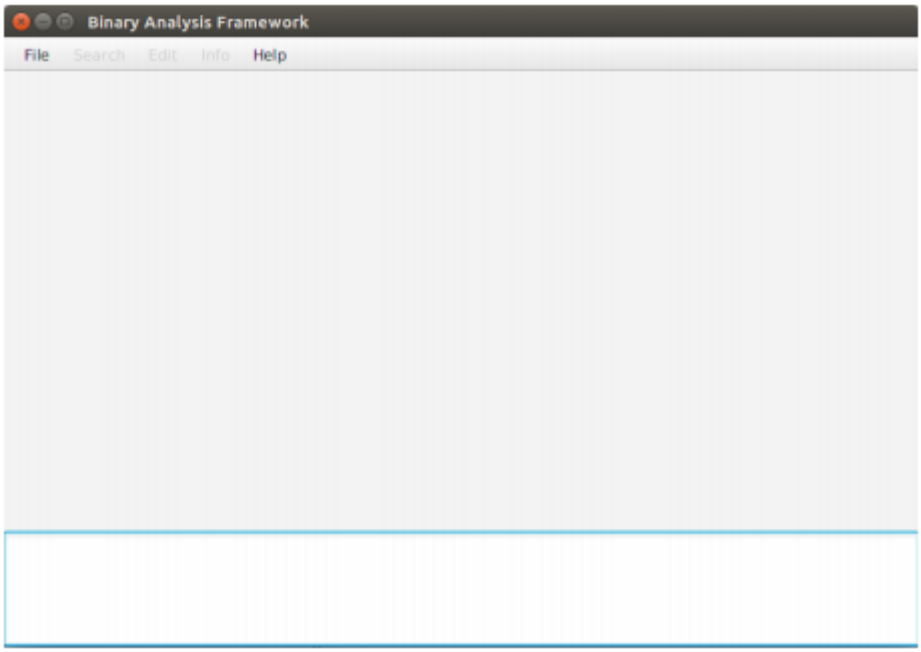

Figure 2: Graphical user interface dialog

The concept of this project is providing logical grouping between software for assessment (Michael, 2019). For instance, a project could be developed to assess a closed source application. Software and software dependencies would be included in this project. Another instance is an operating system profile whereby a related software version of an operating system is developed. The creation of a new project needs the user to offer the project name and the target platform, which is a statically generated list gotten from the database and used for the purpose of display. The selected target platform's choice does not impact the parsing, analysis, or disassembly of the input files. After a project is loaded, a pattern of fine menu options and the primary interface portion become available. If a project has imported software, the disassembly is displayed in the output in tabbed content, and imported functions are generated. This is where the user resumes analysis of the newly imported software. In order to ingress a new file in the project, the user is supposed to choose the import Binary menu from the file dialogue box. After a file is selected, it is presented to the user, where it allows the desired file to be selected in the file system based on the desired format.

The format of file identification permits the accurate determination of the executable code (O'neill, 2017). Its eventual data is populated in a graphical display used to assess software vulnerabilities—inaccurate information presented to the analyst when the framework does not parse the provided file format. After the suitable input file is selected, the binary analysis framework would determine the desired file format where the appropriate file format is identified. Accuracy in file format identification allows accurate determine how a code in a file is executed. When file validation becomes successful, the next step starts at file disassembly and parsing, whereby binary data is identified in the file representing executable code. The foundation of the data is presented to the analyst. After identifying the executable code, it is then disassembled, whereby the produced instruction is displayed. After disassembly becomes successful, the output is stored in a database such as MySQL, which gives both non-proprietary formats to share data. Data persistence is the final phase in disassembling and parsing the input file. The binary framework will utilize the database's data to update the presented graphical display. The binary analysis framework has identification security features such as control flow guard, the layout randomization of address space, data execution prevention, and stack guard, and they are determined during executable program construction (Peter, 2018). The framework identifies the information during parsing file format, and they are stored. After the project has been loaded, the File menu option is enabled, ensuring security information that opens the user interface. In order to determine the efficiency of the Binary Analysis framework, various window libraries were evaluated. Through Binary analysis structure, spot check validation was undertaken on disassembly—output giving the result is displayed in the table below. The results provided by the listing output are equated with the similar output generated by the binary analysis. The provided result is used to compare mnemonics sequence, and if they match the sequence, the starting address is printed and can be confirmed with the function address for generating the listing file. The figure below shows the results provided by the framework.

HOW THE RESEARCH WOULD HELP UNITED STATES

Considering that U.S companies are experiencing extremely high cases of cyber threats, the method of binary analysis would be of great significance to ensure safety (Wang, 2018). The framework of binary analysis is good at identifying vulnerabilities and help security specialists to eliminate threats by employing the most effective countermeasures. The binary analysis can be used in an organization to identify, explore and confirm security issues in any computer system. The framework can be used to assess information concerning various systems as well as third-party software. In that way, a security analyst can utilize the binary analysis to audit computer systems and software for vulnerabilities and provide remediation before further exploitation.

CONCLUSION

The binary assessment framework permits security researchers to undertake full program scrutiny in a binary state (Singh, 2018). This means this method can be used to assess any form of threat in software or a computer. Therefore it is important to understand the binary analysis is crucial to determine computer security. The workability of the framework is facilitated via a graphical user interface that helps in confirming, identifying, and exploring security issues in a targeted computer system or software.

REFERENCES

1) Abel. (2018). CS 260: Binary Analysis for Computer Security. Www.cs.ucr.edu.
2) Benson. (2019). Eliminating Vulnerabilities in Third-Party Code with Binary Analysis Eliminating Vulnerabilities in Third-Party Code with Binary Analysis | 1. http://cdn2.hubspot.net/hubfs/582328/PDF/third-party-code.pdf?t=1457134441878
3) Cascarino, R. E. (2017). Data analytics for internal auditors. Auerbach Publications.
4) Choi, Y.-H., Park, M.-W., Eom, J.-H., & Chung, T.-M. (2018). Dynamic binary analyzer for scanning vulnerabilities with taint analysis. Multimedia Tools and Applications, 74(7),2301 2320.

https://doi.org/10.1007/s11042-014-1922-5
5) Edmonds, M., & Palmore, J. (2017). Defence and security analysis. Defense & Security Analysis, 32(1), 1–3. https://doi.org/10.1080/14751798.2015.1133163
6) Graham, B. (2017). Improving Quality and Security with Binary Analysis. Blogs.grammatech.com. https://blogs.grammatech.com/improving-quality-and-security-with binary-analysis
7) Izumida, T., Futatsugi, K., & Mori, A. (2018). A Generic Binary Analysis Method for Malware. Advances in Information and Computer Security, 199–216. https://doi.org10.1007/978-3-642-16825-3_14
8) Jackson. (2018). Binary Code Analysis. WhiteHat Security Glossary.
9) Jane. (2019). djb research: binary analysis. Cmu.edu. https://security.ece.cmu.edu/binary index.html
10) Jiang, Z., Feng, C., & Tang, C. (2018, March 28). An Exploitability Analysis Technique for Binary Vulnerability Based on Automatic Exception Suppression. Security and Communication Networks. https://www.hindawi.com/journals/scn/2018/4610320/
11) Joseph. (2019). Scholarly Resources: Author Credentials | Trent University Library & Archives. Www.trentu.ca. https://www.trentu.ca/library/help/scholarly/author
12) Liu, K., Tan, H. B. K., & Chen, X. (2018). Binary Code Analysis. Computer, 46(8), 60–68. https://doi.org/10.1109/mc.2013.268
13) Michael. (2019). Binary Image Analysis Technique for Preprocessing of Excessively dilated characters in Aged Kannada Document Images. International Journal of Recent Technology and Engineering, 8(4), 6660–6669. https://doi.org/10.35940/ijrte.d9101.118419
14) O'neill, R. (2017). Learning Linux binary analysis: uncover the secrets of Linux binary analysis with this handy guide. Packt Publishing.
15) Peter. (2018). Practical Binary Analysis. Practicalbinaryanalysis.com. https: practicalbinaryanalysis.com/
16) Singh, L. (2018). Improving Precision for X86 Binary Analysis Techniques. Theses and Dissertations Available from ProQuest, 1–77. https://docs.lib.purdue.edu/dissertationsAAI10844458/
17) Wang, S. (2018). Advanced Reverse Engineering Techniques for Binary Code Security Retrofitting and Analysis. Etda.libraries.psu.edu. https://etda.libraries.psu.edu/catalog 15591szw175
18) Zhu and Liu (2018). Constructing a Hybrid Taint Analysis Framework for Diagnosing Attacks on Binary Programs. Journal of Computers, 9(3). https://doi.org/10.4304/jcp.9.3.566-575
19) Zolotarev, V. (2017). The Technique of Dynamic Binary Analysis and Its Application in the Information Security Sphere. http://barbie.uta.edu/~xlren/BinaryAnalysis The%20Technique%20of%20Dynamic%20Binary%20Analysis%20and%20Its%20Applicatio%20in%20the %20Information%20Security%20Sphere.pdf