Reverse Engineering: A Brief Run-Down

[1]Chaitanya Thakur, [2]Parulpreet Singh
[1]Graduate Student Researcher, [2]Asst. Prof. of CSE
Software Engineering,
Baddi University, Baddi, India

Abstract- An overview of Reverse Engineering is that it is the procedure in which a given object is thoroughly examined, just to procure an abridged knowledge of its origination and functionality. This information usually contains structure charts, data description and PDL to describe processing details. Various steps within reverse engineering are followed to procure this knowledge. The reverse engineering starts with an executable program. The tools that Reverse Engineering procedure uses are dissembler and compilers, and some other tools are also used as per the need. In past, reverse engineering was drawn parallel with the shady and illegitimate uses in context to software piracy. But today, reverse engineering is applied for lots of legitimate applications. In this paper, a detailed view of the Reverse Engineering, Tools and techniques, Uses, and some History of Reverse Engineering is given.

IndexTerms: Reverse Engineering, PDL, dissembler, compilers, software piracy.

1. INTRODUCTION

At the present times, software is a part of human's daily lifestyle. Basically, a collection of programs, procedure, data and documentation creates software. In the engineering, systems are built and then brought to operation for further use, which can be commercial or private. Thus, engineering is considered as a constructive process. Implementation of engineering in software development, is Software Engineering. Software Engineering uses lots of techniques which should essentially have systematic, innovative and cost-efficient approach. Developers prefer Software Engineering to begin and evolve a software because it generates a rational and timely based environment, and also, it ensure a quality commodity along with the command over the various modification, that occur within the software[1]. Software Engineering is further classified into 2 categories.

1.1 FORWARD ENGINEERING

The task of engineering these softwares, from a top-level model to build in complexities and details is known as Forward Engineering This category of engineering has different principles in various software and database processes. Forward engineering is considered to be very great import in development, as it represents the 'normal' development process. For example, building from a model into an implementation language. Also it makes certain that the product built at the end is a quality product by comparing following factors:

- ***Portability:*** Software is considered to be portable if it is usable on different environments with no problem.
- ***Usability:*** If software is required by various modules of the product, the reusability characteristics can help in designing new products with available resources.
- ***Maintainability:*** If the errors present within are being easily corrected and the new functions are being easily added to the product and its functionalities, then product is reckoned maintainable.[2]
- ***Correctness:*** If various Requirements are correctly implemented in the Software Requirements Specification document, in a correct manner, a software product is considered correct.

1.2 REVERSE ENGINEERING

The First software ever written was by Ada Lovelace in the year 1842, though, it was never used. Since 1842, numerous softwares were written and developed. As today, big data storage is a common issue, it is advantageous to upgrade existing old software then to create a new one. But, the problem developers face in this process is that majority of software are hard to understand. This can be because of complex structure of software or incomplete documentation or any other reason. Thus, first developers needed to understand the software.

Reverse Engineering is the process in which a given object is thoroughly examined, just to procure an abridged knowledge of its origination and functionality [3].The procedure of scrutinizing a system to single out the system's components and their interrelationships, and to mould potraits of the system in another embodiment or at a top level of dissociation is Software Reverse Engineering. Various types of program information can be dissociated and examined. In an old system, the dissociate and record of its design is recovered by piecing and composing together information from the source code, existing documents, personnel experiences with the system and application domain knowledge. Design recovery deviates from reverse engineering by putting the stress on the recovery of application domain knowledge, helps to meet informative level of dissociations more than those created by scrutinizing the source code [20]. In past, reverse engineering has been drawn parallel with the shady and illegitimate

uses in the context of software piracy. This substandard image was due to public advertising and legal campaigns of large companies trying to protect their intellectual property. But today, reverse engineering is applied for lots of legitimate applications [1].The reverse engineering process starts with an executable program.

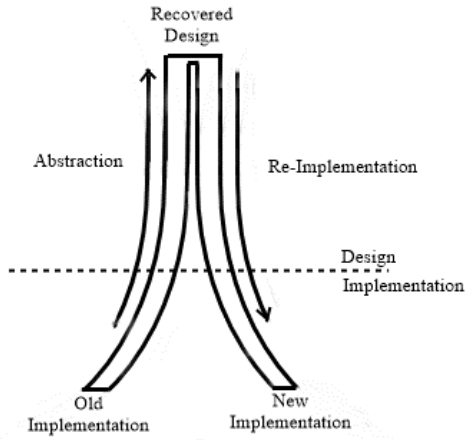

Fig 1: Reverse engineering and Re-implementation process

The process of reverse engineering initiates by extracting detailed design information and a high level design abstraction. The detailed design information is extracted from the source code of software and available documents related to the software. This information that has been extracted, usually contains structure charts, data description and PDL to describe processing details. The steps that are generally followed are described as follows:

- ***Collect Information.*** Collect the available information about the product. Personnel experience must be considered.
- ***Examine Information.*** Reviewing the collected information allows person to become familiar with the system and its components.
- ***Extract The Structure.*** Identify the structure and create set of structure chart using the extracted information.
- ***Record Functionality.*** For each node of structure chart, record the processing done in the program routine corresponding to that node. A PDL is put to use to reveal the program routine's functionality.
- ***Record Data-Flow.*** The recovered program structure and PDL can be analyzed to pick out data transformation in the software.
- ***Record Control-Flow:*** Pick out the top level control structure of the program and record them using control-flow diagrams.
- ***Review Recovered Design.*** Revise the recovered design for consistency with available information and correctness. Pick out any missing items of information and attempt to locate them. Revise the design to validate that it correctly represents the program[17].

1.2.1 USES OF REVERSE ENGINEERI\NG

There are many uses of reverse engineering. Some of the applications of reverse engineering in field of software are discussed further:

- ***Malicious software.*** The detection and Understanding of malicious software (or malware) can be divided into two techniques: Static and Dynamic analysis. Dynamic analysis tries to analyze the program while it is running. It is done while closely monitoring its behavior with other components. In static analysis, the code is reverse engineered and studied. Here, reverse engineering can be used to detect vulnerability in OD or a protocol.
- ***Encryption system.*** One should try to reverse engineer private implementations of the algorithm, that was generate to encrypt the data. It will help to detect the mistake in the algorithm, that can create loop holes.
- ***Developing competing software***. A team of people use to reverse engineer a product, and collect all data related to product. This data is given to another team, keeping the source of data secret. And the team develops a new product using the available data. This method is used to avoid copyrights problem.

1.2.1 TOOLS AND TECHNIQUES OF REVERSE ENGINEERING

To implement reverse engineering, various parts of software are divided and analyzed. For this process, various tools and techniques are used. Some of these techniques are described as further:

- ***Machine code and assembly language.*** An assembly language is a language of which the semantics have a one-to-one written communication with a specific instruction set. The main reason is that machine code is made readable. Giving all possible instruction a name, so called mnemonic, makes it possible.
- ***Disassemblers.*** To map a code to assembly language, the opposite of an assembler is introduced, called a disassembler. The process itself is not exactly difficult, it could be implemented by linearly going through the binary op-codes, and retrieving the mnemonics with table lookups. Difficulties arise when trying to distinguish the actual code from data, since these two are combined. A simple linear scanning-and-translating algorithm therefore does not suffice, and some form of control flow analysis has to be performed.
- ***Compilers.*** Most software is written in top level languages, such as C++ or JAVA. A complier translates programs in these languages to machine code. Compiling is a loss-y process. To be more pinpoint, it is a many-to-many process.
- ***Decompilers.*** While dissemblers can give us assistance to get low level assembly language from machine language, it is still a low level of abstraction. To get a look of big picture here, representation of high level is needed. Because of these reason, it is not possible to obtain the exact same source code. Thus, decompiler is used to induce a top level source file which perchance recompiled with no failure [16].

REFRENCES:

[1] C. Willems and F. C. Freiling, "Reverse code engineering - state of the art and countermeasures," it - Information Technology, vol. 54, no. 2, pp. 53–63, 2012

[2] P. C. V. Oorschot, "Revisiting Software Protection," in ISC 2003. LNCS. Springer, 2003, pp. 1–13.

[3] S. E. Quadir et al., "A survey on chip to system reverse engineering," JETC, vol. 13, no. 1, pp. 6:1–6:34, 2016.

[4] P. Subramanyan et al., "Reverse Engineering Digital Circuits Using Structural and Functional Analyses," IEEE Trans. Emerging Topics Comput., vol. 2, no. 1, pp. 63–80, 2014.

[5] U. Guin et al., "Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain," Proceedings of the IEEE, vol. 102, no. 8, pp. 1207–1228, 2014.

[6] S. Bhunia et al., "Hardware Trojan Attacks: Threat Analysis and Countermeasures," Proceedings of the IEEE, vol. 102, no. 8, pp. 1229– 1247, 2014.

[7] L. Bintu, N. E. Buchler, H. G. Garcia, U. Gerland, T. Hwa, J. Kondev, and R. Phillips, "Transcriptional regulation by the numbers: models," Current Opinion in Genetics & Development, vol. 15, no. 2, pp. 116 – 124, 2005, chromosomes and expression mechanisms.

[8] Esa Fauzi, Bayu Hendradjaya, Wikan Danar Sunindy, "Reverse Engineering of Source Code to Sequence Diagram Using Abstract Syntax Tree", 2016, IEEE

[9] Marc Fyrbiak, Sebastian Strauß, Christian Kison, Sebastian Wallat, Malte Elson, "Hardware Reverse Engineering: Overview and Open Challenges", 2017, IEEE

[10] Andre´s F. Lopez-Lopera, and Mauricio A. Alvarez, "Switched latent force models for reverse-engineering transcriptional regulation in gene expression data", 2016, IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS

[11] Paula Fraga-Lamas, Tiago M. Fernandez-Carames, "Reverse Engineering the Communications Protocol of an RFID Public Transportation Card", 2017 IEEE International Conference on RFID (RFID)

[12] Tuan Anh Nguyen, Christoph Csallner, "Reverse Engineering Object-Oriented Applications Into High-Level Domain Models With Reoom", 2017 IEEE/ACM 39th IEEE International Conference on Software Engineering Companion

[13] Claudia Raibulet, Francesca Arcelli Fontana and Marco Zanoni, "Model-Driven Reverse Engineering Approaches: A Systematic Literature Review", 2017, IEEE

[14] MASOUD ROSTAMI, MEHRDAD MAJZOOBI, FARINAZ KOUSHANFAR, DAN S. WALLACH, AND SRINIVAS DEVADAS, "Robust and Reverse-Engineering Resilient PUF Authentication and Key-Exchange by Substring Matching", 2014, IEEE, Volume 2, No. 1

[15] Navid Asadizanjani, Mark Tehranipoor, and Domenic Forte, "PCB Reverse Engineering Using Nondestructive X-ray Tomography and Advanced Image Processing", 2017, IEEE TRANSACTIONS ON COMPONENTS, PACKAGING AND MANUFACTURING TECHNOLOGY

[16] Asish Kumar Dalai, Shakya Sunder Das, Sanjay Kumar Jena," A code obfuscation technique to prevent reverse engineering" 2017, IEEE WIRELESS COMMUNICATIONS, SIGNAL PROCESSING AND NETWORKING

[17] Eric J. Byrne," Software Reverse Engineering: A case study" Software–Practice and Experience, Vol. 21(12). 1349-1364 (Dec 1991)