

Reverse Engineering

A Swiftly Growing Technology in Software World

A. Preet Inder Singh B. Richa Gupta

Guru Nanak Dev University, Amritsar, India

Email: preet.inder16@gmail.com; richa_gndu@yahoo.co.in

Abstract: With an initial aim of modernizing legacy systems, often written in old programming languages, software reverse engineering has now extended its applicability to virtually every kind of software system. Existing system's modification or replacement often applies reverse engineering for understanding of that system. Reverse engineering is nowadays used for more than one purpose that is sometimes illegal also. This paper presents an insight on this exciting and swiftly growing technology in the computer world.

I. INTRODUCTION

Reverse engineering is the attempt to recapture the top level specification of established product by analyzing it. The practice, taken from older industries, is now frequently used on computer hardware and software. Hardware reverse engineering involves taking apart a device to see how it works whereas software reverse engineering is the process of recovering or reconstructing functional and technical specifications of a software system at a high level of abstraction. In an industrial context, high level architectural descriptions are an essential means of support for software system development. Reverse engineering should aim at recovering architecturally significant views of the system, which can help keep track of the evolution of software architecture. Software reverse engineering is done for various reasons such as, to study how the program performs certain operations, to improve the performance of a program, to fix a bug, to identify program written for use with one microprocessor for use with another. Reverse engineering generally consists of the following stages:

1. Analysis of the product
2. Generation of an intermediate level product description
3. Human analysis of the product description to produce a specification
4. Generation of a new product using the specification.

The specification is made as abstract and functional as possible by the reverse engineers, and is then handed over to a "clean room" design team who have no other contact with the old product, or the team who analyzed it, and who will then design the new product using as little low-level information as possible from the old product. Several different approaches have been proposed by the research community. However, in practice, reverse engineering still presents numerous challenges: scalability of the techniques, usability of tools, visualization, dealing with multiple perspectives, level of abstraction, and so on. Reverse engineering for the purpose of copying or duplicating programs may constitute a copyright violation. This paper initially tries to explain the motivation behind this engineering with a practical example and then explains in details the code and data

reverse engineering and the tools used. The copyright and security breaching issues are also discussed.

II. PURPOSE OF REVERSE ENGINEERING

Reverse-engineering is used for many purposes: as a learning tool; as a way to make new, compatible products that are cheaper than what's currently on the market; for making software interoperate more effectively or to bridge data between different operating systems or databases; and to uncover the undocumented features of commercial products etc. The major motivations behind usage of this technology are as following:

Interoperability: Interoperability refers to production of a product which operates with the product to be analyzed. Examples of such products are applications which need to interoperate with operating systems; software controlled exchanges which need to operate with others; and programs which are locked by hardware "dongles."

Updating/Correcting: Sometimes software need to be updated or corrected as according to the current need, at those times it is required specially in case of no or insufficient documentation.

Lost documentation: It is often done because the documentation of a system has been lost (or was never written), or the developer is no longer available.

Product analysis: To examine how a product works, what components it consists of, estimate costs, etc.

Competition: Competitors may want to produce a product which competes with the product to be analyzed or they may use it for the purpose of competitive technical intelligence (understand what your competitor is actually doing versus what they say they are doing)

Learning: It may be used for learning from others' mistakes. So that same mistakes that others have already made and corrected, are not made by the analyzer.

Military or commercial espionage: It may be used for learning about an enemy's or competitors latest research by stealing or capturing a prototype and dismantling it.

Creation of unlicensed/unapproved duplicates: It is used for producing illegal duplicates.

Cracking: Breaching security is a task performed by crackers in which it plays a vital role and is used by the crackers frequently.

III. A PRACTICAL EXAMPLE

A famous example of reverse-engineering involves San Jose-based Phoenix Technologies Ltd., which in the mid-1980s wanted to produce BIOS for PCs, compatible with the IBM PC's proprietary BIOS. To protect against charges of having simply (and illegally) copied IBM's BIOS, Phoenix reverse-engineered it using what's called a "clean room," or "Chinese wall," approach. First, a team of engineers studied the IBM BIOS—about 8KB of

code—and described everything it did as completely as possible without using or referencing any actual code. Then Phoenix brought in a second team of programmers who had no prior knowledge of the IBM BIOS and had never seen its code. Working only from the first team's functional specifications, the second team wrote new BIOS that operated as specified. The resulting Phoenix BIOS was different from the IBM code, but for all intents and purposes, it operated identically. Using the clean-room approach, even if some sections of code did happen to be identical, there was no copyright infringement. Phoenix began selling its BIOS to companies that then used it to create the first IBM-compatible PCs.

IV. CODE REVERSE ENGINEERING

In current research and practice, the focus of both forward and reverse engineering is at the code level. The importance of the code level is understood in legacy systems where important business rules are actually buried in the code. As a result, the process of reverse engineering has focused on understanding the code. In practice, two main types of reverse engineering emerge. In the first case, source code is already available for the software, but higher-level aspects of the program, perhaps poorly documented or documented but no longer valid, are discovered. In the second case, there is no source code available for the software, and any efforts towards discovering one possible source code for the software are regarded as reverse engineering. Latter case involves reversing a program's machine code back into the source code that it was written in, using program language statements. If the case when source code is already available, focus is there on obtaining software design information (like components and their relationships) and creating representations of a software system at a high level of abstraction. Knowledge about architecture and design trade offs, engineering constraints, and the application domain are obtained by the structural, functional and behavioral analysis of the source code. Knowledge of software architecture from multiple user perspective is needed to make large scale, structural changes, and the capability to perform architecture reconstruction is becoming increasingly important. Thus, reverse engineering techniques are used as an attempt to regain useful understanding and insight.

V. DATA REVERSE ENGINEERING

Most software systems for business and industry are information systems, that is, they maintain and process vast amount of persistent business data. While the main focus of code Reverse Engineering is on improving human understanding about how this information is processed, data reverse engineering tackles the question of what information is stored and how this information can be used in a different context. Recently data reverse engineering concepts and techniques have gained increasing attention in the reverse engineering arena. This has been driven by the requirements for data oriented mass software changes resulting from needs such as the Y2K problem, the European currency conversion or the migration of the information system to web and towards

electronic commerce. Researchers recognize that the quality of the legacy system's recovered data documentation can make or break strategic information goals. The increased use of data warehouse and data mining techniques for strategic decision support systems has also motivated an interest in data reverse engineering technology. Incorporating data from various legacy systems in data warehouse requires a consistent mapping of legacy data structure on a common business object model. Data reverse engineering techniques can also be used to assess the overall quality of software system. An analysis of the data structures can help companies make decision on whether to purchase (and maintain) commercial-off-the shelf software packages. However, current data reverse engineering tools need to overcome the following two significant problems that are Imperfect Knowledge due to dealing with uncertain assumptions and heuristics about legacy data models and Customizability which is because of the various hardware and software platforms and programming languages on which legacy systems are based.

VI. SOFTWARE REVERSE ENGINEERING TOOLS

There is no one particular method for reverse engineering. Depending upon the program different techniques are employed such as: Debugger, Disassembler, Hex-editor, Unpacker, File Analyzers, Registry monitor, File monitor etc. Some tools are described as below:

Debugger: Debugger is a utility that programmers use to find bugs in their programs. Debugger is only tool by which we can trace/break a function or code live. To debug any program, first we put a breakpoint on the required statement and then we run the program. When this instruction is near to be executed the program stops and we can see values. There are many debuggers available in the market but one of the most popular and a powerful debugger is SOFTICE from NUMEGA CORPORATION. After seeing its misuse Numega Corporation has kept some restriction on the sale of this great debugger and a buyer must show that he will not use this debugger for illegal activities. This is a system level debugger, which works directly between a computer's hardware and windows. It cannot be loaded within windows. It must be loaded before windows loads in to the memory. It can monitor every process, threads silently in memory until it is called up using hotkeys. It allows to patch memory at runtime (not permanently and hence we have to use hex editor.) viewing the contents of the register, contains at memory address etc.

Disassembler: As an executable file is in binary format so a normal user cannot understand the instruction in this file. Also any exe or executable is generally in PE format (which is a standard format for exe file, decided by the committee of software companies like MICROSOFT, IBM, and AT&T.). Disassembler converts the binary file in its equivalent assembly language instruction's most of program is written in high level language. There are many disassemblers available, two most commonly used, are WIN32DASM and IDA. IDA is a powerful debugger than WIN32DASM. But WIN32DASM is most widely used. It allows to disassemble any file which is in PE format, disassembly can be saved, it can tell which function is

imported, which function is exported, jump can be executed, string data reference and dialog reference can be found easily, etc.

Hex Editor: As SOFTICE can change the value at memory location only at the run time it is not useful if value has to be changed each time the program is run. Therefore hex editors are used. A hex editor allows changing the contents of any file in hex format. It displays the contents of the file in hex format. Simply the value at memory location has to be changed, which can be found using SOFTICE. Now there are a lot of hex editor available such as ultredit, biew, hiew and a lot. Amongst these HIEW which stands for "Hacker's VIEW" offers a lot of facilities such as editing in hex or ASCII format, searching any string in hex or ASCII format. There is another good facility which makes it different from others is that, it offers to write the assembly code and it can automatically convert this code in to equitant hex format.

VII. COPYRIGHT ISSUES

It is widely accepted that copyright does not protect "ideas", but only "their expression". Reproduction or translation of the whole or a substantial part of a copyright work will constitute an infringement of copyright. Reverse engineering or the copying or duplicating a program may constitute a copyright violation. In some cases, the licensed use of software specially prohibits reverse engineering. Reverse software systems which is done for the purposes of interoperability is mostly believed to be legal, though patent owners often contest this and attempt to stifle any reverse engineering of their products for any reason. In Europe, special codes of protection exist for computer programs, semiconductor topographies, and databases. Each of these contains special definitions of infringement which are binding across the EU, and which (for computer programs and semi-conductor products) mirror those created in the US. A given act of reverse engineering may involve several of these provisions; if so, it needs to be clear of infringement under each different head of copyright work. With copyright infringement, both the creation of the intermediate copy of the original design documents (which takes place after analysis of the product) and the ultimate products created from it may be infringements of copyright, as we will see from the cases.

VIII. REVERSE ENGINEERING A BREACH TO SECURITY

Though reverse engineering is a boost in many cases, it also has some drawbacks attached with it. A major one being: Cracking. "Cracking is a method of making a software program function other than it was originally intended by means of investigating the code, and, if necessary, patching it." Reverse Engineering promotes cracking in many ways as crackers use reverse engineering tools to generate a source-code from an executable. Reverse engineering is used to understand how a program does an action, to bypass protection etc. Usually it's not necessary to disassemble all code of the application, only the part of the application that crackers are interested is reversed. Reverse engineering is used by

a cracker to understand the protection scheme and to break it, so it's a very important thing in the whole world of the cracking. Nowadays there are several cracking groups specialized in reverse web scripts. There is nothing of new in this because the web pages are written in java or CGI scripts or something else. So, they can be considered as small programs. The web cracker usually reverses the protection schemes of web pages creating cracked passwords, which are distributed on the web. Thus, it indirectly promotes cyber crimes. For avoiding such things steps have been taken such as some copyrights do not allow reverse engineering and some reverse engineering tools manufacturers have put restriction on their products. But still crackers have found alternatives for these things. An action against is required so that this boon does not get converted to bane.

IX. CONCLUSION

Investing in program understanding technology is critical for the software and information technology industry to control the inherent high costs and risks of legacy system evolution. There is also a desire, amongst some software users, to re-engineer old software, to make it more modular, re-useable, accessible or reliable. Thus, Reverse engineering is a truly exciting field of research that is ready to be taught and used as a beneficial technology in the industry. Infrastructures for tool integration have evolved in recent years. It is expected that control, data, and presentation integration technology will continue to advance at amazing rates.

REFERENCES

- Ref. [1] Anil Panghal, Pawan Kumar, Sharda Panghal (2009) "Reverse Engineering" Proceeding of 2nd National Conference on "Recent trends and Advancements in Computing", Sirsa, February.
- Ref. [2] Muller H, Jahnke J, Smith D, (2000) Reverse Engineering: a roadmap, Proceeding of the 22nd International Conference on Software Engineering, ACM Press, New York.
- Ref. [3] K. Wong, Reverse Engineering Notebook. PhD. Thesis, Department of Computer Science, University of Victoria, October 1999.
- Ref. [4] Jenkin | Reverse Engineering
<<http://www.jenkins.eu/articles/reverse-engineering.asp>>
- Ref. [5] Software Security and Reverse Engineering
<http://www.infosecwriters.com/text_resources/pdf/software_security_and_reverse_engineering.pdf>