A View of Reverse Engineering

Monika Srivastava
Mechanical Engineering Department
Rama University, Kanpur, India
02monika14@gmail.com

Manish Mishra
M.Tech. Scholar, Department of Electrical Engineering
H.B.T.I. Kanpur, India
manish.mishra2608@gmail.com

Abstract-In this paper; we present a advancement of reverse engineering allowing the integration and interaction of different analysis and visualization tools. The framework architecture that we propose uses a dynamic type system to guarantee the proper exchange of data between the tools and a set of wrapper classes to handle their communication.

I. INTRODUCTION

Reverse engineering, also called back engineering, is the processes of extracting knowledge or design information from anything man-made and re-producing it or reproducing anything based on the extracted information.[1]:3 The process often involves disassembling something (a mechanical device, electronic component, computer program, or biological, chemical, or organic matter) and analyzing its components and workings in detail.

The reasons and goals for obtaining such information vary widely from everyday or socially beneficial actions, to criminal actions, depending upon the situation. Often no intellectual property rights are breached, such as when a person or business cannot recollect how something was done, or what something does, and needs to reverse engineer it to work it out for them. Reverse engineering is also beneficial in crime prevention, where suspected malware is reverse engineered to understand what it does, and how to detect and remove it, and to allow computers and devices to work together ("interoperate") and to allow saved files on obsolete systems to be used in newer systems. By contrast, reverse engineering can also be used to "crack" software and media to remove their copy protection,[1] or to create a (possibly improved) copy or even a knockoff; this is usually the goal of a competitor.[1]

Reverse engineering has its origins in the analysis of hardware for commercial or military advantage.[2] However, the reverse engineering process in itself is not concerned with creating a copy or changing the artifact in some way; it is only an analysis in order to deduce design features from products with little or no additional knowledge about the procedures involved in their original production. [2]

In some cases, the goal of the reverse engineering process can simply be a redocumentation of legacy systems.[2] [3] Even when the product reverse engineered is that of a competitor, the goal may not be to copy them, but to perform competitor analysis.[4] Reverse engineering may also be used to create interoperable products; despite some narrowly tailored US and EU legislation, the legality of using specific reverse engineering techniques for this purpose has been hotly contested in courts worldwide for more than two decades.[5]

I. REVERSE ENGINEERING OF MACHINES

Computer-aided design (CAD) has become more popular, reverse engineering has become a viable method to create a 3D virtual model of an existing physical part for use in 3D CAD, CAM, CAE or other software.[7] The reverse-engineering process involves measuring an object and then reconstructing it as a 3D model. The physical object can be measured using 3D scanning technologies like CMMs, laser scanners, structured light digitizers, or Industrial CT Scanning (computed tomography). The measured data alone, usually represented as a point cloud, lacks topological information and is therefore often processed and modeled into a more usable format such as a triangular-faced mesh, a set of NURBS surfaces, or a CAD model. [8]

Reverse engineering is also used by businesses to bring existing physical geometry into digital product development environments, to make a digital 3D record of their own products, or to assess competitors' products. It is used to analyze, for instance, how a product works, what it does, and what components it consists of, estimate costs, and identify potential patent infringement, etc.

Value engineering is a related activity also used by businesses. It involves de-constructing and analyzing products, but the objective is to find opportunities for cost cutting.

II. REVERSE ENGINEERING OF SOFTWARE

The term reverse engineering as applied to software means different things to different people, prompting Chikofsky and Cross to write a paper researching the various uses and defining a taxonomy. From their paper, they state, "Reverse engineering is the process of analyzing a subject system to create representations of the system at a higher level of abstraction."[9] It can also be seen as "going backwards through the development cycle".[10] In this model, the output of the implementation phase (in source code form) is reverse-engineered back to the analysis phase, in an inversion of the traditional waterfall model. Another term for this technique is program comprehension. [3]

Reverse engineering is a process of examination only: the software system under consideration is not modified (which would make it re-engineering). Software anti-tamper technology like obfuscation is used to deter both reverse engineering and re-engineering of proprietary software and software-powered systems. 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. This second usage of the term is the one most people are familiar with. Reverse engineering of software can make use of the clean room design technique to avoid copyright infringement.

On a related note, black box testing in software engineering has a lot in common with reverse engineering. The tester usually has the API, but their goals are to find bugs and undocumented features by bashing the product from outside. [11]

Other purposes of reverse engineering include security auditing, removal of copy protection ("cracking"), circumvention of access restrictions often present in consumer electronics, customization of embedded systems (such as engine management systems), in-house repairs or retrofits, enabling of additional features on low-cost "crippled" hardware (such as some graphics card chip-sets), or even mere satisfaction of curiosity.

III. BINARY SOFTWARE

This process is sometimes termed *Reverse Code Engineering*, or RCE. [12] As an example, recompilation of binaries for the Java can be accomplished using Jad. One famous case of reverse engineering was the first non-IBM implementation of the PC BIOS which launched the historic IBM PC compatible industry that has been the overwhelmingly dominant computer hardware platform for many years. Reverse engineering of software is protected in the U.S. by the fair use exception in copyright law.[13] The Samba software, which allows systems that are not running Microsoft Windows systems to share files with systems that are, is a classic example of software reverse engineering,[14] since the Samba project had to reverse-engineer unpublished information about how Windows file sharing worked, so that non-Windows computers could emulate it. The Wine project does the same thing for the Windows API, and OpenOffice.org is one party doing this for the Microsoft Office file formats. The ReactOS project is even more ambitious in its goals, as it strives to provide binary (ABI and API) compatibility with the current Windows OSes of the NT branch, allowing software and drivers written for Windows to run on a clean-room reverse-engineered GPL free software or open-source counterpart. Windows SCOPE allows for reverse-engineering the full contents of a Windows system's live memory including a binary-level, graphical reverse engineering of all running processes.

Another classic, if not well-known example is that in 1987 Bell Laboratories reverse-engineered the Mac OS System 4.1, originally running on the Apple Macintosh SE, so they could run it on RISC machines of their own.[15]

IV. REVERSE ENGINEERING OF PROTOCOLS

Protocols are sets of rules that describe message formats and how messages are exchanged (i.e., the protocol state-machine). Accordingly, the problem of protocol reverse-engineering can be partitioned into two sub problems; message format and state-machine reverse-engineering.

The message formats have traditionally been reverse-engineered through a tedious manual process, which involved analysis of how protocol implementations process messages, but recent research proposed a number of automatic solutions.[16][17][18] Typically, these automatic approaches either group observed messages into clusters using various clustering, or emulate the protocol implementation tracing the message processing.

There has been less work on reverse-engineering of state-machines of protocols. In general, the protocol state-machines can be learned either through a process of offline learning, which passively

observes communication and attempts to build the most general state-machine accepting all observed sequences of messages, and online learning, which allows interactive generation of probing sequences of messages and listening to responses to those probing sequences. In general, offline learning of small state-machines is known to be NP-complete,[19] while online learning can be done in polynomial time.[20] An automatic offline approach has been demonstrated by Comparetti et al.[18] and an online approach very recently by Cho et al.[21]

Other components of typical protocols, like encryption and hash functions, can be reverse-engineered automatically as well. Typically, the automatic approaches trace the execution of protocol implementations and try to detect buffers in memory holding unencrypted packets.[22]

V. REVERSE ENGINEERING OF INTEGRATED CIRCUITS/SMART CARDS

Reverse engineering is an invasive and destructive form of analyzing a smart card. The attacker grinds away layer after layer of the smart card and takes pictures with an electron microscope. With this technique, it is possible to reveal the complete hardware and software part of the smart card. The major problem for the attacker is to bring everything into the right order to find out how everything works. The makers of the card try to hide keys and operations by mixing up memory positions, for example, bus scrambling. [23][24]

In some cases, it is even possible to attach a probe to measure voltages while the smart card is still operational. The makers of the card employ sensors to detect and prevent this attack.[25] This attack is not very common because it requires a large investment in effort and special equipment that is generally only available to large chip manufacturers. Furthermore, the payoff from this attack is low since other security techniques are often employed such as shadow accounts.

VI. REVERSE ENGINEERING FOR MILITARY APPLICATIONS

Reverse engineering is often used by people in order to copy other nations' technologies, devices, or information that has been obtained by regular troops in the fields or by intelligence operations. It was often used during the Second World War and the Cold War. Well-known examples from WWII and later include:

Jerry can: British and American forces noticed that the Germans had gasoline cans with an excellent design. They reverse-engineered copies of those cans. The cans were popularly known as "Jerry cans".

Panzerschreck: The Germans captured an American Bazooka during World War II, and reverse engineered it to create the larger Panzerschreck.

Tupolev Tu-4: In 1944, three American B-29 bombers on missions over Japan were forced to land in the USSR. The Soviets, who did not have a similar strategic bomber, decided to copy the B-29. Within three years, they had developed the Tu-4, a near-perfect copy.

V-2 rocket: Technical documents for the V2 and related technologies were captured by the Western Allies at the end of the war. The American side focused their reverse engineering efforts via operation Paperclip, which led to the development of the PGM-11 Redstone rocket.[26] The Soviet side used captured German engineers to reproduce technical documents and plans, and work from captured hardware in order to make their clone of the rocket, the R-1. Thus began the postwar Soviet rocket program that led to the R-7 and the beginning of the space race.

K-13/R-3S missile (NATO reporting name AA-2 Atoll), a Soviet reverse-engineered copy of the AIM-9 Sidewinder, was made possible after a Taiwanese AIM-9B hit a Chinese MiG-17 without exploding in September 1958.[27] The missile became lodged within the airframe, and the pilot returned to base with what Russian scientists would describe as a university course in missile development.

BGM-71 TOW Missile: In May 1975, negotiations between Iran and Hughes Missile Systems on co-production of the TOW and Maverick missiles stalled over disagreements in the pricing structure, the subsequent 1979 revolution ending all plans for such co-production. Iran was later successful in reverse-engineering the missile and is currently producing their own copy: the Toophan.

China has reversed engineered many examples of Western and Russian hardware, from fighter aircraft to missiles and HMMWV cars.

During the Second World War, Polish and British cryptographers studied captured German "Enigma" message encryption machines for weaknesses. Their operation was then simulated on electro-mechanical devices called "Bombes" that tried all the possible scrambler settings of the "Enigma" machines to help break the coded messages sent by the Germans.

Also during the Second World War, British scientists analyzed and defeated a series of increasingly sophisticated radio navigation systems

being used by the German Luftwaffe to perform guided bombing missions at night. The British countermeasures to this system were so effective that in some cases German aircraft were led by signals to land at RAF bases, believing they were back in German territory.

REFERENCES

[1] Eilam, Eldad (2005). Reversing: secrets of reverse engineering. John Wiley & Sons. ISBN 978-0-7645-7481-8.
[2] Chikofsky, E. J. & Cross, J. H., II (1990). "Reverse Engineering and Design Recovery: A Taxonomy".IEEE Software 7 (1): 13–17. doi:10.1109/52.43044.
[3] A Survey of Reverse Engineering and Program Comprehension. Michael L. Nelson, April 19, 1996, ODU CS 551 – Software Engineering Survey.arXiv:cs/0503068v1
[4] Vinesh Raja; Kiran J. Fernandes (2007). Reverse Engineering: An Industrial Perspective. Springer Science & Business Media. p.3. ISBN 978-1-84628-856-2.
[5] Jonathan Band; Masanobu Katoh (2011). Interfaces on Trial 2.0. MIT Press. p. 136, ISBN 978-0-262-29446-1.
[6] Internet Engineering Task Force RFC 2828 Internet Security Glossary
[7] Varady, T; Martin, R; Cox, J (1997). "Reverse engineering of geometric models–an introduction" (PDF). Computer-Aided Design 29 (4): 255–268. doi: 10.1016/S0010-4485(96)00054-1.
[8] "Haman Engineering Solutions".
[9] Chikofsky, E. J.; Cross, J. H. (January 1990). "Reverse engineering and design recovery: A taxonomy" (PDF). IEEE Software 7: 13–17. doi:10.1109/52.43044.
[10] Warden, R. (1992). Software Reuse and Reverse Engineering in Practice. London, England: Chapman & Hall. pp. 283–305.
[11] Shahbaz, Muzammil (2012). Reverse Engineering and Testing of Black-Box Software Components: by Grammatical Inference techniques. LAP LAMBERT Academic Publishing. ISBN 978-3659140730.
[12] Chuvakin, Anton; Cyrus Peikari (January 2004). Security Warrior (1st Ed.). O'Reilly.
[13] Samuelson, Pamela & Scotchmer, Suzanne (2002). "The Law and Economics of Reverse Engineering". Yale Law Journal 111 (7): 1575–1663. doi: 10.2307/797533. JSTOR 797533.
[14] "Samba: An Introduction". 2001-11-27. Retrieved 2009-05-07.
[15] Lee, Newton (2013). Counterterrorism and Cybersecurity: Total Information Awareness (2nd Edition). Springer Science Business Media. p. 110.
[16] W. Cui, J. Kannan, and H. J. Wang. Discoverer: Automatic protocol reverse engineering from network traces. In Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, pp. 1–14.
[17] W. Cui, M. Peinado, K. Chen, H. J. Wang, and L. Irún-Briz. Tupni: Automatic reverse engineering of input formats. In Proceedings of the 15th ACM Conference on Computer and Communications Security, pp. 391–402. ACM, Oct 2008.
[18] M. Comparetti, G. Wondracek, C. Kruegel, and E. Kirda. Prospex: Protocol specification extraction. In Proceedings of the 2009 30th IEEE Symposium on Security and Privacy, pp. 110–125, Washington, 2009. IEEE Computer Society.
[19] Gold, E (1978). "Complexity of automaton identification from given data". Information and Control 37 (3): 302–320. doi:10.1016/S0019-9958(78)90562-4.
[20] D. Angluin (1987). "Learning regular sets from queries and counterexamples". Information and Computation 75 (2): 87–106. doi:10.1016/0890-5401(87)90052-6.
[21] C.Y. Cho, D. Babic, R. Shin, and D. Song. Inference and Analysis of Formal Models of Botnet Command and Control Protocols, 2010 ACM Conference on Computer and Communications Security.
[22] Polyglot: automatic extraction of protocol message format using dynamic binary analysis. J. Caballero, H. Yin, Z. Liang, and D. Song. Proceedings of the 14th ACM conference on Computer and communications security, p. 317-329.
[23] Wolfgang Rankl, Wolfgang Effing, Smart Card Handbook (2004)
[24] T. Welz: Smart cards as methods for payment (2008), Seminar ITS-Security Ruhr-Universität Bochum
[25] David C. Musker: Protecting & Exploiting Intellectual Property in Electronics, IBC Conferences, 10 June 1998
[26] "Redstone rocket". Centennialofflight.net. Retrieved 2010-04-27.
[27] "The Chinese Air Force: Evolving Concepts, Roles, and Capabilities", Center for the Study of Chinese Military Affairs (U.S), by National Defense University Press, pg. 277