Inspiratron.org - Natural language processing, machine learning and cybersecurity

Introduction to reverse engineering

by Nikola Miloševi? - Thursday, October 15, 2015

https://inspiratron.org/blog/2015/10/15/introduction-to-reverse-engineering/

1. Introduction

Reverse engineering is the process of discovering the technological principles of a device, object, or system through analysis of its structure, function, and operation. This involves taking some device, system or software and breaking it apart, analyze it and conclude how it works. When reverse engineer conclude how system works he can take advantage of it, he can recreate it, document it or fix flaws. Reverse engineering is commonly used to document system that is poorly documented and designers are no longer available, to retrieve lost source code and fix problems, to use undocumented API for interoperability, to create competitive project similar as reversed project or to overcome protection. Here we will focus on software reverse engineering in both PC and mobile applications.

2. Reverse engineering use and practice

There are several fields, in which reverse engineering is often used. One of most important is interoperability of systems. Since many larger systems are made by several smaller, there is need to connect them. Not always designers of systems think about connecting system with others into larger system, and not always system has well documented interfaces. In this case reverse engineers has to analyze system and come with interfaces that will make possible to integrate into larger system.
One of the most important applications of reverse engineering is in security analysis of systems and also in malware analysis. Almost we would not be able to talk about this field without reverse engineering. Both attackers and defenders use reverse engineering. Attackers use reverse engineering to realize the weaknesses of systems that they can attack. Also they might analyze operating system undocumented features and options to build malicious applications. On the other side, defenders also analyze and mitigate vulnerabilities of systems. Analyze the attacks, but also analyze malicious applications so they are able to make defenses.
Also other uses are if someone wants to create competitive product with existing. Many open source projects took advantages from reverse engineering such as vine or reactOS. Big companies use reverse engineering to check if some products are violating their patents or copyright. Software testers are also sometimes using reverse engineering to find software flaws and errors. In some cases reverse engineering are prohibited by law, but still used. In EU it is permitted to use reverse engineering for interoperability, but it is forbidden to use it for creating concurrent product.

3. Reverse engineering techiques

When we are talking about software engineering there are several approaches to reverse engineering. In some cases all of them are used to analyze software well. Basically we can separate dynamic and static analysis of some software. Dynamic analysis includes executing software and observing what that software is doing. Also, this include several things like observing GUI of software, software output and input, disk changes, disk read and writes, memory changes, memory reads and writes, network traffic, processor use, debug output if available, application and system logs. Dynamic analysis also includes interaction with applications and observing application behavior.
Static analysis includes decompiling of application and analyzing source code of application. Often it is impossible to retrieve original source, since application was compiled in languages such as C,C++, Delphi or any other lower programming languages. In that case it is possible to retrieve assembler code. Analysis of assembler code is much harder, but not impossible. For that task experience is needed. For some higher programming languages such as .NET languages or JAVA it is possible to retrieve original or almost original source code using decompilers from bytecode. There are also debuggers available that decompiles application and are able to execute command by command. In this case is used some hybrid technique, since we are analyzing source code during execution.

4. Reverse engineering in software

In this section we will focus on desktop software. When software is analyzed dynamically it is important to know what you are dealing with. If you don't really know it is better to have virtual machine with some operating system and all tools for analysis needed installed. Before starting analysis take a snapshot of your virtual drive, since it might be very useful for backup if something goes wrong. It will save you couple of hours of installation process. You can use any virtual machine environment such as VirtualBox, VMWare or HyperV. Since VirtualBox is free and open source I would recommend it. Virtual machine is especially important if you are reverse engineering some piece of malware.
Before you start your analysis it is also important to have all tools that are needed set up. For analysis of network traffic best tool I have dealt with is WireShark (also free and open source). Here you will be able to lively monitor all tcp and udp traffic from and to your PC. Also you will be able to save traffic for some period and analyze it later. Usually operating systems have tools for monitoring performance and load, so you won't need any additional tools for that, but still you may use some dedicated software such as Sandra. Useful tool is also Debug View if

you use windows and if application has debug output (can be downloaded from here) Equipped with these tools dynamic analysis can be started by starting application and interacting with it. Monitor output, logs, windows or linux logs etc. Try to connect input to outputs. Write down your observations. Many times you would not be able to conclude how something works from just one run, so it is iterative process. There is other set of tools needed for static analysis. First useful tool I would recommend for Windows applications is dependency walker. This tool will show you what methods can be seen in which exe or DLL file. Also it will show dependencies of that files (all other referred DLL files). Very useful tool if you are trying to identify missing DLLs or why application is showing some weird error. Other interesting tool is CFF Explorer. This tool is useful for PE editing and process dumping. The Portable Executable (PE) format is a file format for executables, object code and DLLs, used in 32-bit and 64-bit versions of Windows operating system. The PE format is a data structure that encapsulates the information necessary for the Windows OS loader to manage the wrapped executable code. This includes dynamic library references for linking, API export and import tables, resource management data and thread-local storage (TLS) data. On NT operating systems, the PE format is used for EXE, DLL, SYS(device driver), and other file types. PE is a modified version of the Unix COFF file format. PE/COFF is an alternative term in Windows development. If you are going deeper into static analysis you will need some hex editor likewinHex, this will show any file content or number in hexadecimal form, in form that all applications are stored in. Since you want to see source code you will need a decompiler or debugger. If analyzed application is written in .NET you can use Reflector. It used to be free, but it is not any more. There is no other free .NET decompiler I know of. If you analyzed application is written in Java you may use free Java Decompiler. For any lower language application I would suggest to use OlyDbg or IDA Debugger. Both tools are actually debuggers and can be used also in dynamic analysis, but they will give you assembler code that you can further analyze. Also you will be able to execute command by command or to see changes in memory using these two debuggers. With IDA it is possible to debug iOS applications remotely on iPhone.

5. Reverse engineering in mobile world

We will examine reverse engineering and decompiling of major mobile platform - Android.
Android is using apk files for applications. Android applications are written in Java so whole code can be easily retrieved. All you need is Java Decompiler and dex2jar tool. Using dex2jar it is possible to change apk file to jar. It can be done using this command:
linux: sh /home/panxiaobo/dex2jar-version/dex2jar.sh /home/panxiaobo/someApk.apk
windows: C:\dex2jar-version\dex2jar.bat someApk.apk
[reference]
Then it is needed to decompress jar file using some decompresser with ZIP algorithm such as winrar. Then open .class files using Java Decompiler.
