Reverse engineering is a practice of analyzing a software program, either in part or in whole, in order to extract information about its design and implementation. A typical scenario for reverse engineering involves a software component that was implemented for years and carried several business rules in its line of code; but unfortunately, the source code was lost and what remained was the “binary” or “native” code. Additionally, reverse engineering is also used for detecting and neutralizing malware and for protecting intellectual property.

Also known as back engineering, the process involves deconstruction of individual components of larger products in order to extract design information from them.

This concept was originally applied only to hardware, but now it is also applicable to software, databases and even architectural structures. Hence, it enables you to determine how a part was originally designed so that it can be recreated. Analyzing malware through reverse engineering can help identify the details of a breach, see how a hacker entered the system, and what steps they took to breach the system’s security.

## Reverse engineering malware methods

When conducting a malware analysis, the following approaches or techniques are commonly used.

* Static analysis: During this process, the malware or binary is analyzed without actually running it. It can be something very simple, such as looking at file metadata. It can go from disassembling or decompiling malware code to a symbolic execution of a binary without essentially executing it in an actual environment.
* Dynamic analysis: Conversely, during this process, a piece of malware is analyzed while it is running in a live environment. The malware behavior and its side effects are observed. It involves running of tools such as [Sysmon](https://www.varonis.com/blog/sysmon-threat-detection-guide/) and process monitor to check the artifacts a malware creates after running.
* Automated analysis: Automated malware analysis may help in speeding up the process, but it may miss many things due to its generic nature.
* Manual analysis: If the malware has things such as anti-analysis mechanisms or anti-debugging routines, it is preferable to conduct a manual analysis and pick the relevant tools to do it.

## Reverse engineering malware tools

Cyber criminals are well equipped with computer intrusion techniques when attempting to breach into a corporate network. They know their specific objectives, are motivated, skilled, organized and launch funded attacks. Reverse engineering these malware attacks can enable us to identify their techniques for future prevention. Let’s have a look as at some of the tools.

### IDA Pro

[IDA Pro](https://out7.hex-rays.com/demo/request) is a static disassembler that analyzes various malware samples with diverse backgrounds It also comes with an add-on known as [HEX Rays Decompiler](https://www.hex-rays.com/products/decompiler/), which converts assembly language into legible pseudocode. This helps in understanding the code functionality easily.

By taking a sample in IDA Pro, you can see the malware’s entry point. The source code of software that we use in routine is not always available. IDA Pro disassembler creates maps of execution to illustrate the binary instructions that the processor executes in a symbolic representation known as assembly language. This disassembling allows a software specialist to analyze programs suspected to be malicious in nature.

Its graphical view gives a quick representation of mapping the execution flow of the code. IDA Pro also comes with an SDK for developing plugins, automating and extracting useful information. If you prefer Python it also comes with a Python API. Though it can also be used for debugging, it is mainly used for starting reverse engineering analysis of malware.

### Apktool

APKtool is used for reverse engineering third-party binary Android apps. Static ~~It’s a third-party tool~~ that decodes resources to their almost original form and recreates them after a few adjustments. It has the functionality to debug smali code step-by-step and also makes it easier to work with an app because of its project-like file structure. It also enables automation of repetitive tasks such as building an apk.

### EDB Debugger

EDB debugger runs on Linux and focuses mainly on modularity. Some of its noteworthy features include an intuitive GUI, conditional breakpoints, debugging operations (step-over/step-into/break/run), viewing/dumping memory regions and address inspection. The debugging core is implemented in the form of a plugin to allow for drop-in replacements. Data dump can be viewed in tabs, hence letting you have different views of memory open simultaneously and switching between them quickly.

### CFF Explorer

PE viewers extract vital information from executable files, where PE stands for Windows Portable Executable file. CFF Explorer is designed for PE editing and gives full support to .NET binary files, without simultaneously losing sight of internal structure of the portable executable. It includes notable features such as a Hex editor, full support for PE 32/64, quick disassembler (x86, x64, MSIL), drivers and windows viewer, PE rebuilder process viewer, etc.

The CFF Explorer also reveals details about the executable files such as the name, development environment, file type, size, hashing format and PE size. It also allows for performing many operations such as resource modification, PE integrity checks, signature updates, address conversion and rewriting or rebuilding a file.

### Wireshark

Wireshark is a network protocol analyzer that tells an engineer how the program interacts with other machines. This interaction could be what connections the program makes and what data it attempts to send.

Wireshark enables deep inspection of numerous protocols, with more being added periodically. It has multi-platform functionality, as it runs on Linux, Windows, OS X and many other operating systems. The network data that is captured can be browsed with the help of a GUI or a TTY-mode TShark utility. It also allows you to read live data from Ethernet, Bluetooth, IEEE 802.11, [ATM](http://www.cardzgroup.com/ContactSmartCard.html), [Token Ring](https://searchnetworking.techtarget.com/definition/Token-Ring), FDDI, USB, and others.

Wireshark allows filtering the log either before or during the analysis, thus letting you narrow down your search in the network trace. For instance, you can filter out TCP traffic between two IP addresses and set it to show you the packets that are sent from one computer. This filtering functionality is one of the biggest reasons Wireshark is commonly used for packet analysis.

## Conclusion

Automating malware analysis has helped security experts to mitigate advanced malware more effectively and quickly, allowing them to focus more on the very difficult tasks such as understanding reverse communication protocols, new encryption schemes or surgically analyzing the internals of sophisticated threats where intelligence beyond automated sandboxing is required.