

# Theoretical Basis of Software Reverse Engineering

---

## 1. Introduction to Software Reverse Engineering (RE)

Reverse engineering is the analytical process of working backward from a finalized, compiled product to deduce its original design, architecture, and logic. In software security, Reverse Engineering (RE) focuses on taking a compiled, executable binary and translating it back into a readable format so that analysts are able to understand undocumented procedures, discover vulnerabilities, or analyze malware behavior without access to the original source code.

## 2. Compilation vs. Decompileation

The standard software development lifecycle involves the **compilation** process, where high-level human-readable source code (like C or C++) is translated by a compiler into machine code (binary instructions) executable by the CPU. This process intrinsically strips away context, such as comments and formatting.

**Decompilation** is the inverse of this process. While tools like **objdump** or generic disassemblers can map machine code bytes back to human-readable Assembly language mappings, they do turn it into high-level programming constructs. Advanced decompilers, such as **Ghidra** or **IDA Pro**, implement a way to parse assembly instructions and abstract them into C or C-like code, making control flows and data structures substantially easier to analyze.

## 3. Static Analysis and Symbol Re-abstraction

When analyzing a black-box binary, analysts perform **static analysis** which involves the examination of code without running it. Because compilation often strips symbols (variable and function names), analysts must manually reconstruct the semantics of the codebase which typically involves **Variable Renaming and Retyping** but also **Data Flow Analysis** (where does the data go).

## 4. Binary Patching and Control Flow Manipulation

One use case of software analysis is understanding how application restrictions work. But these tools also have the ability to aid in altering the program flow. We will also show how this is done by demonstrating how some checks can be circumvented.