Debugging Malicious Binaries

A debugger is a program that gives you the ability to inspect malicious code at a more granular level.

1. General Debugging Concepts

1.1 Launching and attaching to processes

Two ways to debug:

- Attaching the debugger to a running process
  - Not able to monitor or control the process's initial actions. (When attached, all of startup and initialization will have already been executed)
- Launching a new process
  - Allows the debugger to monitor every action the process takes
  - Able to monitor processes initial actions

1.2 Controlling Process Execution

Important abbilities:

- Ability to control execution
- Ability to interrupt execution

Common execution control options:

- **Continue (Run):** Executes all of the instructions until a breakpoint is reached or an exception occurs.
- **Step into and Step over:** Allows you to execute a single command. The difference occurs when you are executing a command that calls a function. Step into, will stop at the start of the function, whereas Step over will execute the entire function and the debugger will pause at the next instruction.
- **Execute till Return (Run until return):** Execute all of the instruction of a given function, until it returns
- **Run to cursor (Run until selection):** Execute instructions until the current cursor position, or until the selected instruction is reached.

1.3 Interrupting a Program with breakpoints

Allows you to interrupt the program execution at a very specific location within a program.

Types of breakpoints:

- **Software Breakpoints:** Implemented by replacing the instruction at a breakpoint address with a SW breakpoint instruction (Such as `int 3`

instruction (having an opcode of `0xCC`)) When a SW instruction is executed, the control is transfered to the debugger. The dissadvantage is that a malicious program can look for the instruction and alter the debugger's default behavior

- **Hardware Breakpoints:** Breakpoints through the CPU's breakpoint registers: `DR0 - DR7`. Max of 4 breakpoints, the other registers are used to pass parameters. No instruction is replaced. The CPU decides whether the program should be interrupted.
- **Memory Breakpoints:** Pause the execution when a program accesses the memory, rather than the execution.
- **Conditional Breakpoints:** You specify the conditions that must be satisfied to trigger the breakpoint. (Feature offered by the debugger)