Chapter 8: Debugging.

Single-stepping: executing a program step by step. We can see what the variable values are being entered the registers using single stepping.

Single stepping sometimes takes a long time since we are walking through the sample. Sometimes we don’t need to go through every function call in the code; enter stepping over vs stepping into.

Stepping over: when a function is being called and if we step over, we move to the instruction after the call.

Stepping into: when a function is being called and if we step into, we single step through all the code in the function.

Single stepping helps us to set good break points in the code.

Break point: break points allow us to pause execution of a sample at a desired step. When a sample’s execution is paused at a break point, the sample’s execution is said to be broken. We can look at the registers, variables and their values, analyze mistakes in code using break points.

Software break points:

Break point in the execution of a sample. When a developer sets a break point, the first byte is edited to 0XCC the instruction for INT 3(Interrupt 3). Though you can set an unlimited set of software break points the code being edited is visible for other programs accessing this code. If the sample is self-modifying, and it happens to modify the break point, you won’t be able to stop the execution. Enter hardware break points

Hardware break points:

Hardware break points allow you to analyze samples which modify themselves and elude analysis through hardware breakpoint registers. Every time an instruction is executed, there is a check if EIP=Break point register. This is a significant advantage when analyzing malware which edits itself. They can be set on access (read, write or both) rather than execution. Only four registers store for hardware breakpoints. DR0 to DR3 and an extra register to manage the flow. The debug control(DR7) register manages the Dr0 to DR3 registers by storing values and whether the represent read write or execution break points. Malicious programs try to access these registers and edit the values, but fortunately we can set the general debug flag to trigger a breakpoint when a mov instruction is being executed on a debug register.

Exceptions:

An Exception is an unexpected happenstance, some body writes to an address which isn’t there. Someone divides by zero. When this happens, there is no way for the program to be able to handle what has happened and a rare moment of potential appears.

A sample being debugged might also throw exceptions (breakpoints utilize exceptions under the hood- How?) and the debugger needs a plan to handle the exception for the sample to continue to execute, they are; first chance and second chance exceptions. They can be explained in the context of a normal program execution and then with a debugger;

A program may contain certain bugs which might cause exceptions. Most modern programs have exception handler code to make sure that the program is running even if an exception occurs. If there is no exception handler code, the program crashes because of the exception. This is when the program is running without the debugger attached. When a debugger is attached; exception handling takes a different turn; here in where the first chance and second chance exceptions come into picture;

1. When an exception occurs in the sample being investigated, debugger is given the **first chance** to handle the exception. The debugger can choose to handle the exception or pass it on to the sample.
2. The sample, if it has any exception handler, could handle the exception on its own.
3. If the sample doesn’t have the code necessary to deal with an exception, the exception could be thrown back to the debugger. This is called a second chance exception, and the debugger must handle it.

Debugger functionality like single stepping and breakpoints also utilize exceptions to control the flow of execution (which many samples also utilize). As discussed above, a breakpoint is executed by INT 3 exception (which the debugger handles, but can also be passed onto the program, even if only to see if the sample has any self-modifying code). Single stepping is also iimplemented at the OS level as an exception, by setting the trap flag.

A debugger could be used to deconstruct the execution of a sample and force it to execute in ways that facilitates our analysis. We can edit the stack frame of a function and examine how it is working. We can skip over the execution of a function by stepping over its code and etc.

More practice is required to understand how the fuck to do this. We will probably revisit debugging in OS fundamentals.

Olly debug:

We need to take each section and practice separately; Also, the same with Java- you need to take each section and practice- with an IDE.