IC 101P: Reverse Engineering
Lecture (N-1)/N: Software Reverse Engineering

Varun Dutt

School of Computing and Electrical Engineering
School of Humanities and Social Sciences
Indian Institute of Technology Mandi, India

Scaling the Heights

Software Reverse Engineering

- Software reverse engineering is about opening up a program's "box," and looking inside
- No screwdriver is needed: It is a purely virtual process on the computer
- The process is used by a variety of different people for a variety of different purposes: prevalent in industry to outpace competitors

Software Reverse Engineering: Applications

- Security-Related Reversing:
  - Malicious software: used by both developers and defenders
  - Cryptographic Algorithms: to find encryption and decryption algorithms
  - Digital Rights Management: to find cracks and copyright infringement for copyrighted software, music, and movies

Software Reverse Engineering: Applications

- Reversing in Software Development :
  - Developing Competing Software
  - Evaluating Software Quality and Robustness

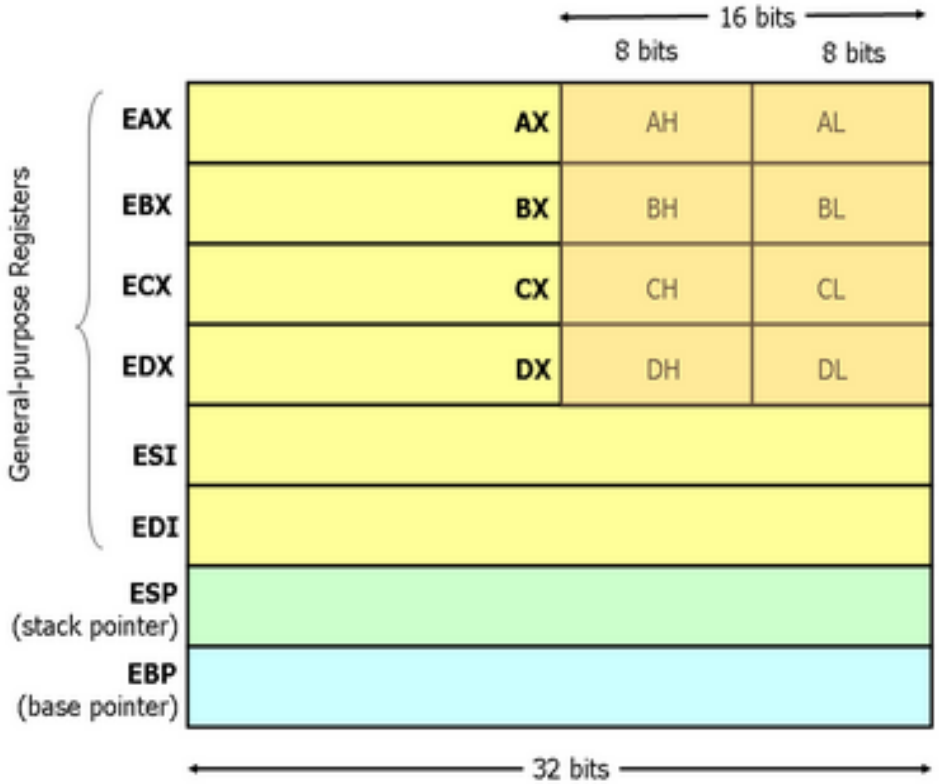
Software Reversing is done through Reverse-Engineering Tools

- Decompilers: going from executable to source code. E.g., Sothink, JAD (for Java), dotPeek (for .NET) (mostly done for managed software)
- Disassemblers: going from executable to assembly code. E.g., OllyDbg, IDA Pro, SoftICE (mostly done for unmanaged software)
- Debuggers: Allows to observe the assembly code while it is running on the CPU.

Little About CPU Registers in Intel Architecture

The CPU registers

The CPU registers are the fastest memory located in the CPU itself. They are basically used for every low – level operation, they are the super-fast data storage of the processor. For x86 architectures there are usually 8 32 bit long registers, 2 of which hold the base pointer and the stack pointer that are used for navigation between the instructions. The registers are even faster than the *Static RAM* (SRAM, known as the cache) and, of course, the *Dynamic RAM*.



Assembly Language Primitives

Quick overview of the Assembly language

For this article we need to know few basic things about the assembly language so we can actually understand what we are doing. The ***Assembly language*** is *unstructured* and is based on very primitive instructions, which are divided in the following general types (I'll describe only the basic operations) :

Data movement instructions

mov – used to copy data from one cell to another, between registers, or between a register and a cell in the memory
push/pop – operates on the memory supported stack

Arithmetic instructions

add/sub/inc – arithmetic operations. Can operate with constants, registers or memory cells

Control flow instructions

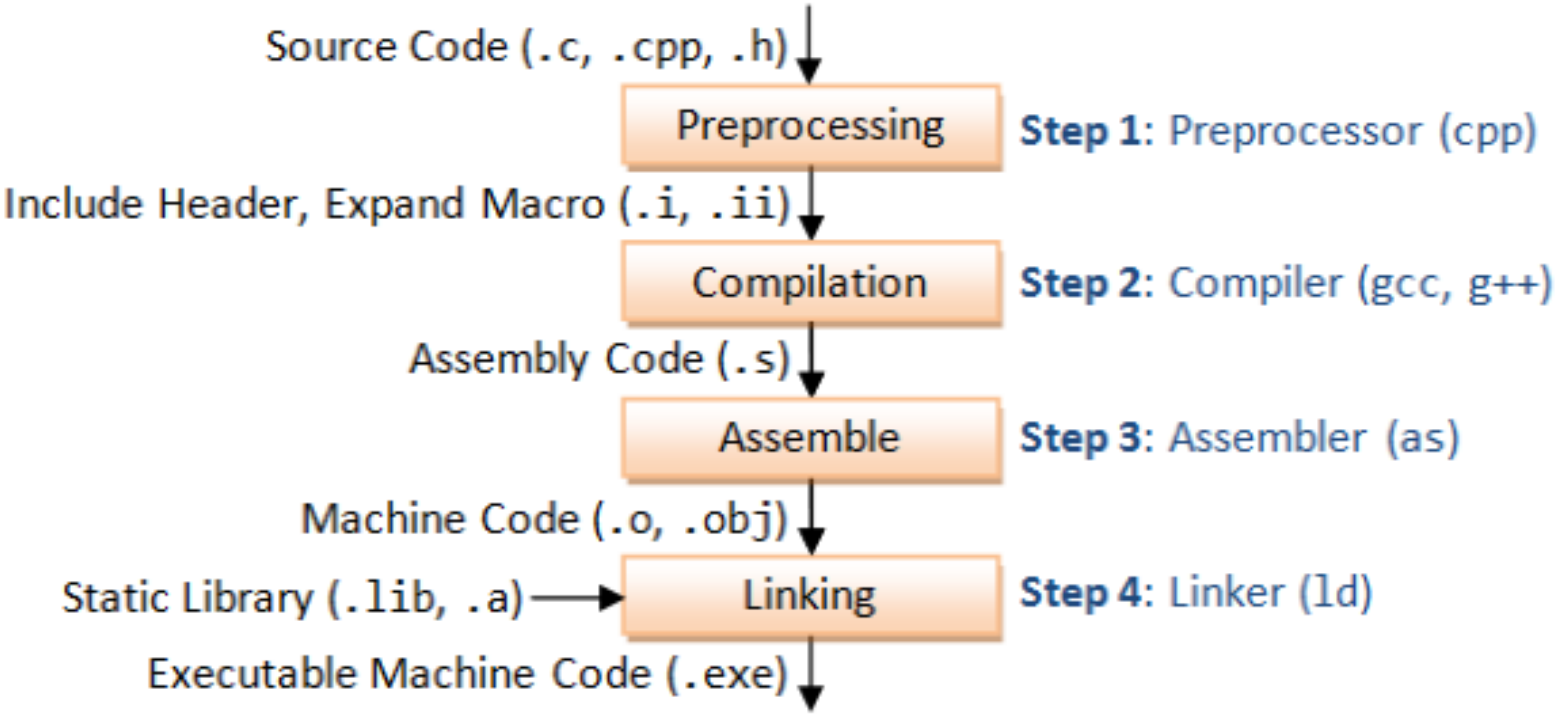
jmp – jump to label or a cell in memory
jb – jump if condition is met
je - jump when equal
jne - jump when not equal
jz - jump when last result was zero
jg - jump when greater than
jge - jump when greater than or equal to
jl - jump when less than
jle - jump when less than or equal to
cmp – compare the values of the two specified operands
call/ret – these two implement the routine call and return

The Control flow instructions are what we are most interested in here.

If you want to learn more assembly language…

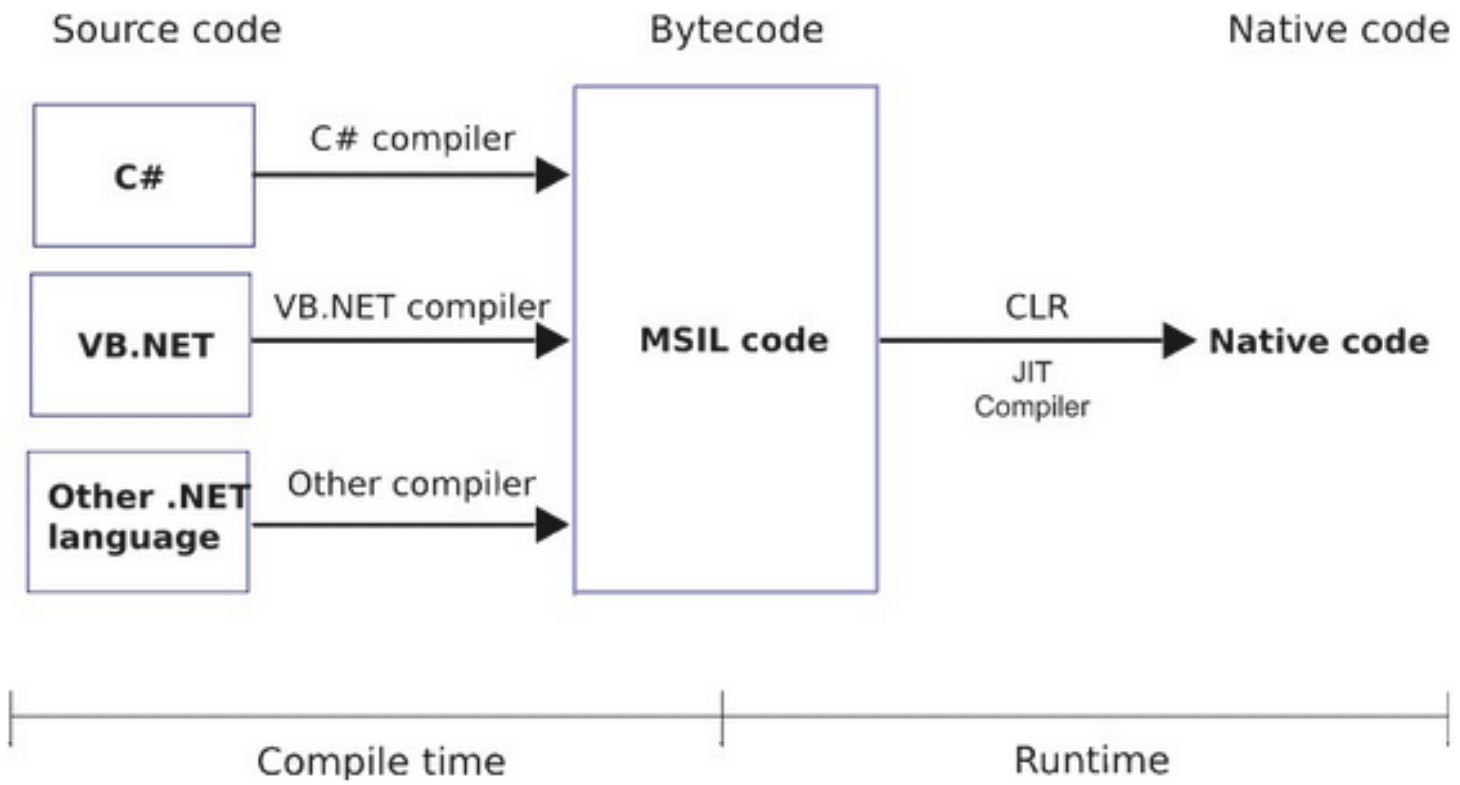
- Refer to: http://www.cs.virginia.edu/~evans/cs216/guides/x86.html

An unmanaged compilation process: C/C++



Reversing mostly done through Disassemblers (e.g., Ollydgb)

A managed compilation process: .NET



Reversing mostly done through Decompilers

Reverse Engineering a C++ program

- Tutorial Execute

(As unmanaged: Ollydgb will be used.)

Reverse Engineering a Java program

- Tutorial Hello World!

(JAD decomplier could be used – Java is under the managed category)

Reverse Engineering a .Net program

- Secams

- Assignment for next class for each group:
  - Please decompile Secams and give me a 5 minute presentation on how it works [I want to see the code-level details]
  - You can use either codereflect or dotPeek for this purpose

Reverse Engineering a Flash Program

- Banner tutorial

- Assignment for next class for each group:
  - Please decompile Climate Change and give me a 5 minutes presentation on how it works [I want to see the code-level details]
  - Use SoThink Decompiler for this purpose

Last Assignment: A Reverse-Engineering challenge

- Try to bypass the username and Registration window in tknm3.exe
  - Please suggest the smallest change you will make in the Exe to accept any username and Registration ID greater than 3 characters
  - You need not save the change in the .exe file (just show it to me in OllyDgb in the next lecture)
  - Hint: please use OllyDbg

So it is actually not so hard to crack an application?

- Thumb Rule:

  “Every application can be cracked, if you have access to its native image [through reverse engineering], just like every computer password can be broken, if you have physical access to the machine.” (Kosta Hristov, a writer and a software engineer based in Ireland)

Thank you!

Comments and Questions most welcome!

References

- Reversing: Secrets of Reverse Engineering. Eldad Eilam. 2005. http://www.amazon.com/Reversing-Secrets-Engineering-Eldad-Eilam/dp/0764574817
- http://www.tekdigest.com/decompile-java-class-file-using-jad-decompiler.html
- http://www.developingthefuture.net/disassembling-decompiling-and-modifying-executables/
- http://www.cs.virginia.edu/~evans/cs216/guides/x86.html
- http://en.wikipedia.org/wiki/Reverse_engineering