# RE/Binary Analysis - Zero to Hero Learning Path

## 1. Basics

When first learning anything, it is best to build a broad and solid foundation. With Binary Analysis, it is best to keep in mind what you are actually doing: dissecting a mass of bytes and attempt to extract meaning. To do this, you must be informed about the technologies that created that chunk of bytes and how the various sub-chunks relate to each other.

### Programming Basics

Most malware is built in C++, which is like C on steroids. It is best to start with C, which is still the foundation of the modern computing stack and the paradigms you learn here will inform the rest of your journey.

- C Programming Tutorial for Beginners - freeCodeCamp.org
- Pointers in C / C++ - freeCodeCamp.org
- Harvard's CS50 Lecture on C language

### Systems Intro

You will be dealing with assembly, and as you will see, assembly language causes direct modifications to internal hardware state, so it is best to cover both of these topics now. These topics are rather difficult if you have no prior exposure, so repeated readings and practice may be necessary.

- Stack/Heap Explanation
- Von-Neumman vs. Harvard Architecture
- Process and Threads explained
- x86/64 Assembly tutorial 24:02-45:13
- Daax Rynd's RE Blog Series - Intro to x86 Architecture
- Syscall Intro - ask nicely and the spirit of the OS will do your bidding

## 2. Next Steps

By this point, you should be comfortable writing basic C programs (definitely beyond the complexity of Hello World) and also with reading x86 assembly code. During this section, we will expand on these topics to provide an even further foundation that will be necessary for further exploration of binary analysis.

### Intermediate Programming

- The Cherno's C++ Series - C++ is absolutely massive and so is this series, I would recommend skimming essential parts of this series such as OOP, the STL, and maybe even some template meta-programming. Learning and mastering the fundamentals of classes will be particularly useful down the line
- TechWithTim's Python course - Eventually, when we get into Ghidra to do anything non-trivial you will want to start writing your own scripts that can automate much of your analysis, Python will be the tool of choice for this job, unless you happen to be a Java guy(gross)

### Beginner Systems

Hardware is a world of its own, and best believe that malware authors will know the deep dark corners of the system and how to abuse them. For this reason the system must be explored further.

- Daax Rynd's RE Blog Series - The Stack
- Daax Rynd's RE Blog Series - Accelerated Assembly p1
- Daax Rynd's RE Blog Series - Accelerated Assembly p2
- Wikipedia - x86 Calling Conventions
- Windows 64 bit calling conventions
- OpenSecurityTraining2 x86 Course - OST is a gem

### Reverse Engineering Intro

At this point, I believe that it is useful to start getting your hands wet. You really have to understand the prior concepts for the information presented here to mean anything to you. Particularly make sure you're comfortable in C, and have a firm mental model of how x86 works.

- 0xRick's blog series on the PE File format
- Spicy Python libary for PE Format - Working with the PE file format hands on is the best way to really get to know its structure and functions. Using Python for this task is a much gentler intro than using C++
- Practical Malware Analysis ~ Chapter 1 - This gives a vary brief overview of what to look for in the PE file format when performing analysis
- Rich Header Explanation - Master this obscure artifact for attribution and more
- Compiler Explorer - Write some basic C programs here and analyze the output, this is the quickest way to become adept at reading assembly
- **PE File Format Projects** - This is an easy topic to grasp, but a tough one to master. I suggest working on a few projects to understand the workings of the PE format and how it can be abused
    - TryHackMe PE Header practical - analyze some binaries with various PE explorer tools
    - Basic Cryptor - This is one of my first PE Format projects. I implemented a very basic cryptor in C++
    - IAT Hooking in Code is a slightly more advanced project that I did that uses PE format manipulation during runtime to perform IAT hooking, a popular malware technique
- HackADayU - Ghidra Intro Series - This is a great place to start tinkering with binaries

- GuidedHacking RE Tutorials - These focus primarily on game hacking, but the novelty of hacking games is a great motivator to keep going
- TryHackMe Debugger Module - Knowing your way around a debugger is essential, this intro covers x64dbg which is my favorite because it is open source and has a vast array of plugins such as Scylla/Titanhide
- OpenSecurityTraining2 - Windbg 1 - WinDbg is the Window's debugger, it comes strapped with symbols for Windows binaries, which makes it worth mastering for studying Windows Internals, and visualizing process manipulation
- OpenSecurityTraining2 - Windbg 2 - Do not feel the need to go forward to WinDbg 3, it is unnecessary at this point

**Windows Intro**

User mode malware will have to make use of operating system functionality to accomplish anything meaningful. Get to know the wide and wonderful world of Windows and your reverse engineering skills will reap the benefits.

- Windows Virtual Memory Intro
- Windows Protected and Special Process Intro
- Windows API Intro
- The Windows way of binaries(x64 assembly) - Great intro to tools used by malware authors and other systems programmers
- Process Environment Block Overview - An important data structure for user mode processes
- Thread Environment Block Overview - Non trivial malware will have multiple threads, get to know this data structure as well
- Create an External Hack for AssaultCube - This tutorial introduces fundamental Windows API techniques that form the foundation of malware development

# 3. Not Quite 1337 but close

**Advanced Programming**

Any competent malware author is going to be a good programmer, learning some deeper programming skills will help you greatly in trying to dissect their code.

- freeCodeCamp.org ~ Algorithms and Data Structures
- freeCodeCamp.org - Advanced Data Structures
- MyCodeSchool - Great in depth algorithm explainers
- If you wanna go deeeep

**Intermediate Systems**

- Daax Rynd's RE Blog Series - Exceptions and Descriptor Tables
- OpenSecurityTraining2 - x86/64 OS Internals - This is the course I took that made it all click for me
- UMass OS Course - This course will greatly deepen your theoretical knowledge of how operating systems work
- Code Machine Windows x64 Internals - This article has amazing material on exception handling and stack unwinding, while tying in some WinDbg magic, malware and exploit authors abuse these techniques on the reg

**Intermediate Reverse Engineering**

- OpenSecurityTraining2 - Reverse Engineering C++ Binaries - learn how to recognize classes, constructor/destructors, inheritance, virtual functions, and more in assembly
- SentinelOne's jumpstart guide to Ghidra scripting - The documentation for Ghidra's scripting API is quite hard to navigate, I recommend scanning Github and seeing what other people have done, for example here is a script that I wrote for extracting windows API calls from a binary https://github.com/ColeStrickler/win32extract/blob/main/win32Extract.py .
- More Ghidra Scripts
- OpenSecurityTraining2 Intro to Vulnerabilities course - learning about vulnerabilities will teach you a lot about how programs work on a fundamental level, which will only help your reversing skills
- Blackhat Presentation On Defeating Packers - This presentation is a goldmine.
- x64dbg Tutorial on how to Call pre-existing functions

**Intermediate Windows**

- PE Format Documentation - Seriously, make sure you have most of this down
- Palantir Blog on ETW
- TryHackMe ETW Module
- Awesome ETW
- AMSI Intro
- AMSI Overview

**Malware Intro**

- 0xPat 9 part Malware development series
- Checkpoint Evasion Technique Database
- Checkpoint Anti Debug Technique Database
- TryHackMe Abusing Windows Internals Module
- GuidedHacking Process Injection Tutorials
- AMSI Bypass 1 - Here is my complete PowerShell implementation of this technique so you can test it yourself AMSI Patching
- AMSI Bypass 2
- Process PEB Manipulation

# 4. Advanced RE

The journey really only begins here, after you complete this section, you should have an idea of the RE landscape and be able to pursue further study on your own. This section will introduce more advanced malware techniques and deep systems level knowledge that will be useful for combatting advanced malware created by competent programmers.

### Advanced Systems

- OpenSecurityTraining2 - UEFI - Learn about how modern firmware works, its security model, and how bootkits can subvert the root of trust
- Hypervisor Development - Knowing a thing or two about this will be helpful for kernel level reversing and greatly expand your knowledge of computer architecture
    - OpenSecurityTraining2 - x86 Intel VT-x - rather outdated, but if the details no longer apply the concepts are still timeless
    - AMD-V Hypervisor Development intro - AMD does things a little bit differently when it comes to hypervisors with its processor intrinsics, see https://github.com/ColeStrickler/1337Visor for a full implementation
    - Rayanfam Hypervisor from Scratch
    - Daax Rynd's Virtualization Blog Series - The most in depth information, and also explainers for other features that can be created are located here. I found this resource the most useful when creating my own  hypervisor
    - Github - Satoshi Tanda - Crowdstrike Dev who specializes in Hypervisors. If reading source code is your thing you will find an absolute treasure chest here

### Advanced Windows

- Geoff Chappell - Windows Exposed - This crazy dude reverse engineered a ton of the Windows OS. invaluable resource
- ReactOS - An open source Windows clone, This will come in handy sometimes when you run into a real sticky issue. The code and functionality is surprisingly close to the real thing
- Windows API Reference
- Pavel Yosifovich's Windows Kernel Programming - I recommend buying his book here: [Amazon]
- Windows Kernel Hacking Tutorial - Intro to Windows Kernel programming, learning how to operate here is useful because advanced malware resides here, and a lot of the functionality of EDR software is implemented here as well
- Kernel Callback Objects - MSDN
- Kernel Callbacks
- Windows Internals Part 1 and 2 - a rather dry read, use this resource when you need a deeper background on a specific topic
- Windows Driver Code samples - invaluable if you want to get anything done in kernel mode, this heavily commented code IS the documentation for this topic.
- Kernel Patch Protection
- Syscall Detection Implementation - EDR Internals

### Advanced Reverse Engineering

- secret.club - these guys are truly incredible
- Back Engineering - really unique and diverse RE work happening here
- HyperDbg Resources → White Paper, Documentation → i told you that learning about hypervisors would be worth your while

### Intermediate Malware

- Hooking Intro
- Function Hooking - This is fundamental, as this is how malware subverts visibility, and also how EDR's gain telemetry that isn't available through Kernel callbacks. I used this post for inspiration to make a library that works for 64 and 32bit Windows processes. Look this code over(its heavily commented) and understand how it works. Please do not use in malware https://github.com/ColeStrickler/ExtendoClip/blob/main/ExtendoClip/Hook.cpp
- Direct Syscalls - evade function hooks
- Modern Spin on direct syscalls
- Unhooking Functions - blind EDRs
- Call Stack Spoofing - intro to a modern technique used by commercial C2s
- Reflective DLL Injection 1
- Reflective DLL Injection 2
- Reflective DLL Injection 3
- Reflective DLL The GuidedHacking Way
- Windows Shellcoding Guide
- Position Independent Shellcode
- Pre Empt Maelstrom Series 1-7
- Boku Blog - probably the world's foremost authority on shellcoding

### Advanced Malware

- VX Underground - the world's biggest resource on malware. Includes source code, binaries, white papers, APT malware, techniques, tutorials, and everything else you could dream of
- Black Angel Rootkit
- UEFI Bootkit
- Improved Reflective DLL Injection
- Sleep Obfuscation
- BOF Intro - These are the wave in modern C2s. Knowing how they work is a must
- BOF Loader
- Writing BOFs
- EfiGuard

**Recommended Paid Resources**

- Rootkits and Bootkits[Book] - I found this book to be an enlightening read. You will walk through reversing kernel and firmware level malware that was the bleeding edge of its time. For a free PDF click here [PDF]
- The Rootkit Arsenal[Book] - This is my all time favorite book on malware, it is old and focuses on 32bit Windows so that the actual implementations need updating, but the ideas it teaches will change the way you think about malware. This is the bible. On O'Reilly: [O'Reilly]
- Windows Kernel Programming[Book] - This book introduces Windows Kernel concepts through hands on projects, learning these concepts will greatly deepen your knowledge of EDR internals and rootkit technology.
- GuidedHacking - An amazing forum for video game hackers. This is where I got my start in programming and RE. In the ultra-niche community of RE it is a running joke that video game hackers would run circles around the cyber security community. Find out why this is true(lol) here

There are further topics in this field, such as symbolic execution and binary instrumentation, that I do not have sufficient experience to comment on. If you want to be a true expert in RE these subjects are the frontier and there are numerous whitepapers and research findings being published in journals where you can read more on these topics.

# 5. FAQ

1. **Do I really need to learn programming?**
    a. My brother in bytes, how do you expect to reverse code if you don't know how the original code works? Yes programming knowledge is a necessity.
2. **Do I need to follow this roadmap/learning path in exact order?**
    a. No, although following this roadmap in order will prepare you for successive stages sometimes it is good to look ahead to see how different pieces of knowledge come together. Seeing the big picture helps with motivation and in digesting the information you are receiving. Jumping around in this list will also be necessary when you need to go back and reference things.
3. **Do I need to learn both 32bit and 64bit x86 assembly?**
    a. Yes, malware authors will take advantage of both. They are also very similar and learning one makes learning the other much easier.
4. **How long does learning all of this take?**
    a. This depends entirely on the effort put in. You could read all of this within a month and absorb nothing. I recommend taking your time and completing various projects that implement the various techniques you read about. I have attempted to add examples of ones that I completed along my journey for motivation and reference.
5. **I am struggling with (insert concept here), what can I do?**
    a. If you are struggling with a concept it is must likely due to weak foundational knowledge in other concepts. You may want to review and solidify concepts listed formerly in the roadmap. Additionally, seek other resources, there is tons of information that is freely available. This list is only an attempt to synthesize this information into a linear path that can be followed.