—- Want to contact me? Find me

Totally_Not_A_Haxxer

This book, "REMF – Reverse Engineers Field Manual," is intended solely for educational purposes. It explores security-related programs, tasks, operations, techniques, and work within an ethical and legal framework. The author and publisher do not endorse or promote any form of malicious activity, and all techniques described herein should be used responsibly and legally. Any resemblance of characters, businesses, places, events, or incidents to actual persons, living or dead, or actual events is purely coincidental.

Section 0x00 > The Void

Copyright	|p1-2
This	|p3-7
Welcome New Reader	|p9
$whoami	|p9-11
About The Artwork	|p11-14
How To Read This Book	|p14-15
Who is This Book For	|p15-16
Structure of This Book	|p16-18
Why no 'Real' Book	|p18-19
What This Book Won't Cover	|p19-20
Personal Authors Note	|p20-21
Why This Series Was Created	|p21-25
Huge Cyberhugs & Thanks To…	|p25-29
Experimental Clusterfuck	|p29

Section 0x01 > Intro To RE

Introduction	|N/A
What is RE	|p33-34
Different Worlds of RE	**|p35-40**
Software RE	*|p35-36*
Firmware RE	*|p36-37*
Hardware RE	*|p37-38*
Cryptanalysis	*|p38-39*
Protocol RE	*|p39-40*
Cybersecurity x RE	|p40-43
RE Outside of Cybersecurity	|p43-44
Programming for RE	|p44-48
Why Should I Learn It?	|p48-51
RE & The Learning Curve	|p51-53

What you should know for RE |p53-55
Different worlds of RE |p55-59
SW Commonly Used |p59-62
SW This Book uses |p63-64

Section 0x02 > SRE (Software RE)

Introduction (*cover*) |p66
SRE, The Most Popular |p67
What does SRE Assist You With? |p68
Comfort and SRE Sessions |p69-71
Good Health is FUCKING Important |p72-74
SRE IN Stages |p74-77
SO Many Variables |p77-80
Useful Things to Know **|p80-85**
SRE Wonderful Resources |*p81-84*
Small Tips **|p85-94**
Tip: Research Your System |*p85*
Tip: Self Built Training |*p86-92*
Tip: Training with malware/other |*p92-94*
SRE To Smart Contracts |p94-99
Binary->Web2.0->We3.0? |p99-100

Section 0x03 > Applications of SRE

Introduction |p102-103
Reversing Windows Applications |p103-104
SRE Methadologies |p104-106
Static Analysis **|p107-116**
Strings |p108-113
Control Flow Analysis |p113-115
Static Taint Analysis |p115-116

Dynamic Analysis **|p116-119**
Being Proper **|p119-135**
Sandboxing Apps *\p121-122*
MicroV *\p122-123*
Virtual Machine's *\p123-135*
Air Gapping? *\p125-126*
Emulation *\p126-127*
Memory Analysis |p127-129
Debugging |p129-131
Dynamic Taint Analysis |p131-132
Binary Fuzzing |p132-133
Environment Analysis **|p133-135**
API Calls **|p135-144**
File System *\p136-140*
Process Mgmt *\p140-142*
Network Related API Calls *\p142-144*
Reversing Bloaty Software |p144-146
Real World SRE |p146-153
Finally - Start Small |p155

Section 0x04 > Software Protections

Introduction |p157-158
Two Sides of the Same Story |p159-161
Anti-Analysis Techniques **|p161-191**
Anti-X *|p162-174*
Binary Obfuscation *|p174-189*
Binary Packing *|p189-192*
Anti Tampering Techniques **|p192-228**
Compile Time *\p194-198*
Load Time *\p198-203*

Runtime \p204-211
Code Level \p211-213
Environment Level \p213-228

Section 0x05 > Binary Auditing

Introduction |p230-239
A KickStart |p239
Binary Vulnerabilities |p240
Where to Start |p240-252
Looking for Binary Vulns |p252-258
BOFs |*p253-261*
String Formatting Issues |*p261-266*
Software Based Flaws |*p266-271*
Double Free |*p271-276*
Integer Overflow |*p276-279*
Binary Bugs |*p279-285*
Finishing |p285-286

Section 0x06 > Networks & RE

Introduction |p288
Protocol RE (*REcap lol*) |p288-293
What are Custom Protocols |p288-293
How does learning this help? |p293-294
How is the skill applied? |p294-310
Setting up the right env |*p297-299*
Analyzing env activity |*p299-301*
Static/Dynamic Analysis |*p301-305*
Furthering To Exploitation |*p306-310*

Concluding |p310-312

Section 0x07 > RE x Crypto (extension)
Introduction |p314-321
Analyzing Implementations |p321-326
Attacking Implementations |p326-335

Section 0x08 > Practical from New Angles
Introduction |p338-339
Theory Gets You Far But.. |p339-341
Simulated to Real World |p341-342
Legal Stuff |p342-345
RE is a constant exercise |p345-346
RE & Different Angles |p346-348
RE Left Over Tips and Tricks |p349-380

Section 0x09 > The End
Introduction |p382-383
A Raccoon Call |p383-385
Final Words |p386

Section 0x0
The Void

Welcome New Reader ^_^!

New or not, welcome to the book! The Reverse Engineers Field Manual is a book that is a part of a series that walks you through beginner basics all the way to advanced and master's related topics in the field of reverse engineering. I would like to also state that this is the first official set of books I am doing, unlike the first 3 experimental books (*BHGM, BHPM, GHFM*). Since this is the first book I will be introducing you to the field of reverse engineering, primarily focusing on Software Reverse Engineering (*SRE*) while exploring other fields.

Moving forward ....

$whoami

Before I start every book, I like introducing you to myself. Hi, I go by Totally_Not_A_Haxxer and in short, I am a cybersecurity researcher & reverse engineer with over 4 years of experience, carrying

other roles such as security analyst, author, and software engineer. I spend a lot of my time building programs, helping people at hacker cons by volunteering my time, managing communities, and I also spend a lot of time training, studying and working on projects that are expanding my knowledge in the reverse engineering & engineering space as these are both two fields I have the most passion for! If I was to mention something I enjoy about this field, it is the depth of knowledge that it is possible to obtain. This honestly keeps my studying going and quite energetic. Pertaining to reverse engineering, I have spent a HUGE portion of my cybersecurity career in reverse engineering. From game reverse engineering & cheat development all the way to proprietary network protocol reverse engineering, I have done it...During those times, I also took an interest and deep dive into cryptological reverse engineering and just never made it super deep, but deep enough to know surface level knowledge. Additionally, I have spent my time working on many different malware reversing operations, malware

development operations, both game cheat development and game cheat reversing and even spend time reversing new types of products that claim they are *'anti reverse engineer*'. This skill then traverses into the need to find better education. So, I introduce you to me, hopefully we can chill now O_O, and glide into the book!

About The Artwork

Many people who have gotten this book's design or had a chance to look at it before were estranged to the new art style. Before, it was all ASCII text and different terminal like styles, now it's a more modern, old dark style.

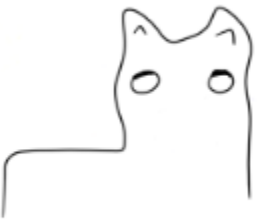
So why the change?

The artwork on these booklets is all drawn by the author of the book, me. And each set of art comes with its own set of personal meaning. For example, the character below, who is also on the cover of the book, Rovnax,

is known to hold a personal meaning that states

~ Life is about forgiving but now always forgetting

< Character is shown on next page to save on spacing and formatting issue, thanks Amazon and my current writing self >



(Rovnax – Planet C-3 Entity)

The phrase "**life is about forgiving but not always forgetting**" combats the original phrase "**forgive and forget**" by emphasizing that when we forget the pain in our life, we are bound to repeat the same mistakes but it

is the fear of repetition that will hold us back from not forgetting.

I thought of this series like a music album, something I could feel is deeply my own, custom in style, fluid in structure, smooth in flow, and loud in tone. Most importantly, it saved me from copyright or making money off other people's art, plants are cheaper right?

With that being said, I hope that the reader (*you*), reading this feels welcome to the book and can grasp it from my angle or at least the idea. I say this in hopes that it also inspires you to do what you want out of life, even if it's something a lot of people do not approve of *(okay do not murder anyone, I do not literally mean* ***anything*** *T_T.*) Anyway, you get the idea.

To sum up this tiny section, the artwork is mainly personal to me but designed to be fit to hold meaning that can apply to everyone. Considering the purpose and intention of the book, I think it's appropriate to include the

characters for the personal goals in this booklet.

How To Read This Book

On previous manuals, I had to add the solution to this question because some people did not quite understand how to read the book or how to use it, considering it's the manual format that gets them mixed up.

Say you are taking a reverse engineering course, if the course did not go deep on a specific topic, word, or missed something, maybe this book comes in handy.

Or you may be a college student with a class that teaches reverse engineering but only scratches the surface level and you want to totally kick ass in the field- while his book won't help you be this sick bad ass in RE, it will give you a lot of different forms of knowledge all of which can be applied to either the field, standard coursework and content work, or general studying. Who knows, maybe you are just a knowledge

fiend. Either way, take this book as just a mass collection of notes and personal experiences that you can read on set in a specific order to follow beginner advanced.

Who Is This Book For

This **series** is fit specifically for all types of reverse engineers- whether beginner or advanced. As the contents are designed for more ‘niche’ scenarios. But, I will note that when you read through this book, topics may get more and more advanced if you are literally only two or three days into reverse engineering because this book does try to build on a LOT as fast and as formatted as possible.

Specifically, from the community, it seems as if this book, *Reverse Engineers Field Manual - v><⊐□⊡>* edition, is fit well for…

- *Beginner reverse engineers*
- *Students in cybersecurity overall*
- *Developers picking up RE for software dev*

- *Newcomers who want to explore a different field*

But, if you are a professional reading this, seriously would love to hear further thoughts, hell, if you want, tear this book to shreds and eat it out in terms of issues & language standards, always up for that kind of conversation + I could use free editing. Lol

Structure Of This Book

Before this book begins, it is important to define some of the general structure, and notes that have gone into the design of the book. For many, it may not be new, but for the ones that are new to my books, reading this may help you understand what specific symbols or specific sections of texts may mean. I say this because I am kind of just here for fun, community and yapping. As you can imagine, we are not here to make the most professionally written book in the world but just provide quality content!

Moving forward, this book contains many different sections and formats, all of which have their own meaning. They are shown in the list below.

- **[W]** - Sections marked like this are usually a **W**arning. This can be a warning on executing a command, writing specific scripts, be an ethics mention, legal or code cite, etc. Either way, these are usually important if you are new.

- **[B]** - Indicates a section of text that is ***B***ias. The reason such a book (*an educational book*) has bias in it, is to ensure that the experiences of the own author are put into place to combat specific points and to expand the ideas of the reader. By marking section **[B]**, I expand that the intention behind the bias content is not some random scam bullshit trying to be passed off as real knowledge but

instead stems from experience.

- **[IN]** - ***IN****formational* sections contain expansions on existing sections or contain information that is an extension of existing information in the section.

- **[R]** - One important element you get with self-published books is something called *'**R**anting'*. Sections marked with an R are usually a rant. So, if you are here for the rant, then look for these sections.

- **[M]** - Indicates a **M**isconception that is being discussed or broken down.

Why No 'Real' Book?

Initially, when this series was designed, I had created the idea of splitting the reverse engineering series up into multiple manuals. However, I finished the outlines for the second and third manual, then realized-what

is the worth? Why am I trying to mash this much content into this format? Plus, I have overdone this format if I have 4 books out onto of the other 2 planned to be written. It just feels over saturated. And, to a point, technical content does not belong in very specific formats, as it will fit way better in others. So, I decided to switch it up and create the Reverse Engineering Field Manual series to bridge a series of full books on reverse engineering going through my own CTFs, walking through some unique sides and aspects of the reversing space, and so much more! That way, these manuals can act as a sort of guide to the bigger books and there is a series which is forever expandable. Of course, this organization could be optimized, and I have figured some ways, but oh well, all in good learning, right? LOL

What Will This Book NOT Cover

Before we get into the body content of the book, I figure that it is important I mention this to my readers. This book will **NOT** cover

basic and beginner computer science theory, the concepts of memory on a base level, teaching assembly, etc. So, if you are looking for a foundation, I suggest understanding those components on their own before continuing. However, I designed the content in a way that **MAY** be able to suit people who do not have that foundation, as the goal is to make the content somewhat digestible for beginners.

Having the background does help, and you won't get far in reverse engineering anyway without learning those elements further, so if you skip it, you may set yourself up for more bumps down the line that can be prevented by just studying it now. But I am not here to complain about what you are going to study or when and either way, I will not be going over the very depth of basic underlying concepts.

Personal Authors Note

Personal authors notes are something I like leaving to the community members,

supporters, and people I have in the background that have supported me for a time. It is insane to me to think it has only just been 3 years at the time of writing this book that I have actually had attention in the community, its wild to see the growth, and additionally, it is even wilder to see the amount of people that are KILLING it in the community. 2024, the year I got to see the what is called "*underground*" (*feel like this word has been killed by content creator BS*) and hidden parts of the cybersecurity community shine- from absolutely world changing operations, to seeing DefCon behind the scenes, it's been truly fucking absurd to think that those two things do not even encapsulate 0.2% of the things I have seen from the community this year. So, if it helps any of the people reading, if you are a part of this community, I appreciate the ongoing support, it has been truly mind-blowing to be in the positions I am in here, and to do the things I am doing- for the love, and for the community. See you at the ending author's note.

Why This Series Was Created

I must be real here; the reason this entire series was created was due to a deep amount of dislike with the educational systems ongoing in the world. From my own high school to the cybersecurity specific education routes. And no, I am not talking about the typical teenage annoying '*I hate school*' shit. I am talking about serious bones to pick.

When I started out in reverse engineering, even despite the courses, resources, etc. that were existing 3-4 years ago, the reverse engineering field still did not have good education. The reason I say this is primarily due to...

- *Copy & pasted content*
- *Cookie cutter content*
- *Stolen content*
- *Overly expensive content*
- *Old outdated information*

This was huge and still is today.

My biggest and most laughable example of this is the game hacking community. I spent a lot of time there, I have love and respect for the true hacking and people of that community *(such as the folks at GuidedHacking*), but I do have a bone to pick with the content creators. If you Google game hacking, the content is the same, its either ripped off from paywall platforms like GuidedHacking, is ChatGPT generated, or is stolen from another content creator who stole it from somewhere else. Even then, the content is never "*what*" something is, it's always "*how to do something*" like "*how to bypass EAC*".

The problem I have with this and even struggled with myself is primarily that for beginners, it leaves people in a large range of confusion later down the line when they go to balance out or advance their skills.

For example, you spend all this time watching videos of people building game cheats, you write the code despite not knowing what it does, you reverse despite

not knowing what you are truly doing under the surface and build a semi successful game cheat. Later down the line, if you were asked to do this but instead of game cheats apply the same skill to exploit development, you would be completely lost (*despite this being on you for not bothering to do anything like Google but just copy/paste content*.)

Because the video never explained what was happening and you never cared to do the digging, did you truly learn?

This is the same issue I was stuck in, and often, I was stuck later frustrating asking myself "***how***" because I could not always find the best resources that were up to date. So, because of this, I felt that maybe I could try to do something about it at least to my own community and provide a decent series or collection of resources and information that can assist them during their learning paths, especially from my own perspective.

I want to also assert that I am not trying to state that *absolutely all forms of education* in reverse engineering are bad, but there are a lot of them that are *rinse/repeat, copy/paste, stolen* content that is not quality to help assist and teach genuine, caring and passionate students.

Huge Cyberhugs & Thanks To...

Before I started the first official series of the *Reverse Engineers Field Manual*, I had so many different experiences, people, organizations, and groups pop in. All of which I would like to acknowledge as a reason for pushing me forward.

SkyPenguinLabs

SkyPenguinLabs is my own organization and something I wanted to startup for my own sake which is in the process of being turned into an

active company!!! Woo!! But I mainly want to say thank you to the wonderful people we have on standby supporting the community and actually being interactive with our servers and channels!

This seriously goes a long way and assists me in getting projects such as this one down and out for the buckets!

Car Hacking Village

This is a heavy one for me. At my last job, I had met someone truly wonderful, the Chief President of the Car Hacking Village, Justin Montalbano who assisted me a long way in my automotive cybersecurity career, and also invited me to come help out at the Car Hacking Village for *DEF CON 32*, primarily for the Car Hacking Village Kids operations in coordination with *DCNeXtGen with additional volunteer aid. DCNeXtGen was DefCon's attempt to bring kids*

from around the world into the world of cybersecurity, and we did this by introducing a fun set of pins, badges, and various exercises kids could be walked through throughout the Car Hacking Village space at DefCon and learn about different parts of the automotive cybersecurity world!

The reason I chose to acknowledge both Justin, and the team at the CHV is not only because of such a wonderful show they put on every year, but because of the extreme hard working, loving and supportive community the CHV fosters every year which I had the honor and privilege of contributing to. DefCon's CHV brought me out of some burnouts by giving me the chance to explore new routes in the car hacking world, meet people that genuinely cared and shared an interest in things I was doing, and brought me into a room with some amazing industry experts

from different areas. I was honestly completely blown away by this and just HAD to acknowledge it and pass it back on my own end.

Additionally, I was even happier to see the SkyPenguinLabs logo get dropped on the CHV Kids badge as a surprise from the badge development team!

Contributing to a village felt like the only true way to enjoy DefCon, and it was one of the best experiences I can say I had at a conference which kicked off my interest in volunteering at community events!

HVCK Magazine

While not a direct supporter of the book. I would like to give Ryan William and HVCK magazine some light here because of the amount of community respect I have for them and how much they have let me slam their magazine with article contributions lol! Hats off to Williams for creating one of the best Cybersecurity magazines out there and one of the most community driven ones out there as well. Seriously, I have seen too many people create "*community driven*" magazines that are by no means community driven haha! The work put into magazines is by far the best I have seen for an online magazine in regard to cyber, and contributive art forms. To check them out, visit their site to pick up their free e-mags! https://hvck-magazine.github.io/

This Will Be A Clusterfuck

Since this is the very first book of this series, things start off a little bit wonky and definitely may be a considerably clusterfucky with how much I pushed into the main content sections, for example, we go from tackling *Software Reverse Engineering (SRE)* to jumping straight into *RE->BCRE* (*SRE to Block Chain Reverse Engineering*) and then hop into SRE from a base standpoint. Additionally, I want to just mention that by design these books are supposed to be quite 'clusterfucky' and 'chaotic' in an organized way if that makes sense.

So, it may be kind of awkward at first to get used to, but when you walk more through the book, the design and flow should be quite easy and clear to grasp.

Section 0x01
Intro To RE

What Is Reverse Engineering?

Reverse Engineering, in a tech specific context is defined as the process of taking a specific component of a given piece of tech,

then using specific methodologies to figure out the underlying architecture, design, idea of operation, and technical operations. The *goal* behind reverse engineering is to simply figure out what something does.

Like hacking, the term *'hacking'* can really be broken down into a personal definition – what does hacking mean to YOU? Similar thing here, what does RE mean to you? To me, reverse engineering is more than just a skillset, it is an art form that takes quite the fuck load of patience, it takes a ton of experience and constant practice (*like most things*), but most importantly, links itself with other fields, such as psychology.

This is very true and is often missed. However, when reverse engineering a piece of software (*especially during malware investigations) you* can look at it in a way that you are trying to get inside of the developer's headspace and mentality when developing the program.

Of course, this will change depending on your goals, but that small element does change a lot down the line. You may be

asking right now, though, *how does it become the study of someone else's head?*

Well, when reverse engineering say a really good game cheat, if you want to find out what specific libraries a piece of tech may use, or may even want to find more geographic-specific information, for example: keystrokes, the way a program is designed, the type of design of a program, will all play factors into your reversing process *(again, depending on goals.*)

Now, you see me often representing types of software such as **game cheats**, and malware which point out software.

However, despite me pointing out software, there are other forms of reverse engineering out there, all of which go into some really badass areas.

Different Worlds Of RE

There are many different applications of reverse engineering, or the bare concept of

reverse engineering, from software all the way down to the bare bones of hardware. However, you may most come across the following fields of reverse engineering, all of which this series will be covering.

Software Reverse Engineering

Software Reverse Engineering, or simply 'SRE' as an abbreviation, is the process of taking a specific piece of software, and applying a combination of technical methodologies, knowledge foundations, techniques, and other areas of skill to get to understanding what the unknown piece of software. Software reverse engineers, ironically enough, use software and sets of frameworks such as IDA-Pro, Ghidra, system tools, etc. to assist them during their reverse engineering. By using task-specific tools, the reverse engineer can gain a better foothold.

Firmware Reverse Engineering

Wait, did we not just cover software reverse engineering (firmware=type of software)? Well, yes, we did, but different types of software might also have their own unique branches of reverse engineer due to the large differences between software and firmware, a type of software. Firmware reverse engineering involves an entirely new set of skillsets to help in reverse engineering or finding flaws within the firmware of a system. Like standard software reverse engineering, the reverse engineer may have to use a set of methods to unpack the firmware (*software*), emulate it (*execute it*), or take extra steps even at the baseline level to get information about the firmware they are looking at. Usually, firmware reverse engineering is used in an IoT context, where attackers or researchers attempt to find flaws within the firmware, software, or hardware within a device that includes embedded with sensors, software, and more [*defines:IoT Device]*. Firmware reverse engineering may be used to find default credentials, flaws in the required software or versions of specific software,

default configurations, vulnerable configurations for the system and more.

Hardware Reverse Engineering

Hardware reverse engineering or HWRE (*in this book, lol*), as you can imagine, consists of taking a specific electronic component and doing a series of physical and digital tests on the hardware that can result in spitting out information about the system, design, or more to help the reverse engineer understand the piece of hardware they are looking at. Usually, someone may be using tactics or techniques from HWRE when they want to find or leverage that information to obtain a successful exploitation of the device, but there are other genuine reasons why a person may be applying those techniques to a piece of hardware for more 'positive' needs, such as forensics. Often times, HWRE is considered a 'advanced' field of reverse engineering that takes a lot of time, effort, practice, and one thing for sure- money. Due to the need for specific machines and systems such as

ChipWhisperer, despite being recognized as ‘fairly cheap’ are not affordable for your ‘everyday working person’ and you will for sure have to invest in more than just one tool. I would like to emphasize (*despite my mistake in the previous statement*), HWRE will not be talked about in this series, primarily because the author does not have enough experience in that field, nor funding at the current moment.

Cryptanalysis

Out of all the time I spent within the reverse engineering world, I have to state that the idea of taking cryptographic algorithms, and reverse engineering them is one of the harder fields of reverse engineering, but the most rewarding. Thats right, cryptanalysis, or as I like to phrase it (*despite not being recognized or a real word in the industry*) is the process of taking a specific cryptographic custom or general algorithm and finding ways to uncover, or *‘crack’* the algorithm. The purpose behind cryptanalysis is extremely important, as it ensures that encryption truly

is as strong as it is, and also allows researchers to discover new ways to break or reverse existing cryptographic algorithms. Not only does it help out with the security of the algorithms, but it also assists in finding new ways to improve the algorithm.

NP Reverse Engineering

Short for Network Protocol Reverse Engineering (NPRE), is the process of taking a custom or proprietary network protocol and reverse engineering it to understand how the protocol works/interacts, how it was designed, what the protocol does, and many different things which can all assist someone in finding flaws, finding bugs or issues within the protocol, finding ways to abuse it in programs (*will get to this in a second*), and more. I did mention abusing the protocol in a program. What I mean by this is using the combination of standard SRE and mixing it with NPRE. Ideally, a reverse engineer would not only reverse engineer the network protocols design and inner workings, but would then reverse engineer the program

that receives, and parses data that uses that protocol to find ways to exploit it. By understanding the protocol, a reverse engineer could slap together a program to send a packet which contains that protocol to the software that parses it and see if exploitation is possible. Network protocol reverse engineering is one of the more difficult areas, and it is not talked about nearly as enough as it should be, but it's something that comes in handy as a security researcher or simply if you are just trying to expand your security skillset.

Now of course, we have reached a point in the book where we should start touching further into the security portion, so let's plungeee lol!

CyberSecurity x RE

Reverse Engineering, on its own is not specific to cybersecurity, and many people often forget this. Reverse Engineering is

simply just trying to figure out what a piece of software, hardware, or design is. However, this is not to say that reverse engineering cannot apply to cybersecurity, because it can!

Reverse Engineering is mainly applied to a lot of exploit-development related fields, such as game hacking, malware development, and other related fields. But exactly **why** does it get implemented into cybersecurity?

The primary reason is because reverse engineering can be useful for numerous things often applied to security specific contexts.

For example, often, reverse engineers will be tasked to reverse engineer a piece of software and find specific indications of flaws within the programs code, flaws within the security *(such as plaintext credentials)*, missing security systems such as anti-debugging systems, and more.

The cross happens when the goal of reverse engineering is targeted from a security-specific angle. Below are some examples of scenarios that are and aren't security specific.

- A reverse engineer wants to document code for a legacy, binary application that no longer has source code available – **this is NOT a security context**

- A reverse engineer wants to find a loophole in a piece of software that allows them to bypass software license entry to use the software – **this IS a security context**, particularly, malicious, known as software cracking.

I feel like you get the idea between the two, or you should. Reverse engineering can be used for more than just debugging apps, writing documentation and more- but instead be

used for leverage on pretty much anything from online to offline environments.

Reverse Engineering, however, is primarily associated with software security because **most** of the context, in which reverse engineering is mentioned usually aligns with a goal to exploit or assess the security of a piece of software and this is quick to realize when you go to Google, type '*reverse engineering*' and see videos authored by game hackers, or cybersecurity specialists, thus, the reason people get quite confused and mix reverse engineering as a field of cybersecurity, when it is its own thing.

RE Outside of Cybersecurity

Along with the confusion between reverse engineering as a field, or as a subfield of cybersecurity, comes the use of it outside of cybersecurity. But believe it or not, RE has genuine uses outside of cybersecurity. Take the following examples below.

- *Debugging*
- *Documenting (as mentioned before)*
- *Design Analysis*
- *Information Forensics*
- *Proprietary Software Interfacing*

The last one is by far my favorite to bring up. Proprietary software interfacing is a way of saying that someone developed a method for interfacing with proprietary software. For example, the person that reverse engineered Apple's protocol suite known as Airplay in 2011, could have developed a set of software that interacted with the proprietary network protocols and services due to the information obtained from reversing it which led to exploitation or taking advantage of the server to bypass restrictions.

Do I NEED Programming For RE?

Out of every presentation I have done, every talk or even community lesson, such as the ones with the *Safer Internet Project* on

reverse engineering, I have had at **LEAST** one student ask if they **NEED** programming for reverse engineering. My immediate answer is no, but this is the short version.

The long version? While programming not being an absolute necessity to learn reverse engineering, you will not be able to grow extremely advanced without some experience in development, if not a lot. The reason being is because understanding both the flow of a programming language, understanding the different functionalities, building the different functionalities, and more would help assist you during both your studying and practice by exposing you to new angles of how systems get built from the ground up, engineering->implementation.

I came across a YouTube video posted in 2024 that was by a relatively popular youtuber who stated that programming was necessary, and if you are going to reverse engineer, you must know the language you

are reversing in by learning how to program in the language.

I formally disagree with this. The reason? To put it simply, the focus should be on understanding compilers and how they influence the final binary, rather than solely on the programming language used.

To put it a long way...

- **At the end of the day, it's not the programming language that matters, it's the compiler.**

I argue this all the time, at the end of the day, it is *not the programming language* that makes a difference in reverse engineering, rather, it is the compiler that changes the outcome. We see this when we use command line switches to change the result of the compilers *ready-to-execute* output.

If you go around telling people 'You *MUST learn how to program in rust to be good at REing rust*' then you might as well tell every reverse engineer they suck at RE because they cannot program in C++, but directly reverse C++ compiled binaries.

Though, I understand the point. The reason I go and argue that the compiler changes the output is because ultimately the compiler is specific to the design of the programming language. When saying *'the language changes the outcome'* you are essentially stating that the code you write in a text editor or code editor will change the output of the final binary, which is not what changes the outcome. While there are portions of a programming language, and different parts of its designs that **may** change the outcome of a binary, you want to focus more on the primary component that outputs the final executable code, since this is what you will be reversing.

Finally, knowing how to develop in the language will not make you a good reverse

engineer for binaries produced by the compiler that implements that language, but rather, it is the knowledge foundation you have on the underlying operations that go into the languages compiler which will make you good at understanding the structure, and code of the binaries that are produced by that compiler.

Why Should I Learn It

Well, you are here, so you should already know why you want to learn it. But why **SHOULD** you learn RE? What's the point?

Ultimately, it depends on what you see yourself doing. If you see yourself primarily debugging applications, it may assist you in understanding how programs can become bugged, how to spot issues and even learn how to connect things logically a lot more easily.

If you see yourself breaking software……

Do I need to go further? That one should be obvious…

In general, though, for people reading this book who take a genuine interest and are new, reverse engineering can help you with many things, and that is both in and out of cybersecurity. *[B]* Here are some general examples of what it's been able to help me out with (*based on experience*).

- **Conceptual & Logical Thinking:** Something I have cherished a lot is the ability that reverse engineering gives you to properly pull things together. It's kind of just appears but I like to think of reverse engineering as a puzzle, in fact- it is. You have a set of questions that you need to answer to get to another result which is the full picture of the program or artificial object you are trying to figure out.

- ***Patience***: Patience is another huge thing which many people thankfully

point out. Honestly, if you thought programming was a bitch, try sifting through 10,000 lines of raw assembly LMAO. All in all, despite the fun, RE takes a lot of patience, you have to pay attention to everything and actually learn to take the necessary breaks, I feel that RE is one of those fields that takes considerably more thought processing to connect than others, thus I say, it definitely teaches patience in the same way that general security research does (*REAL research, not some basic ass XSS bug submitted to H1*)

- **Keeping It Simple Yo!:** Before RE it was quite easy for me to overcomplicate a lot of logical processes, especially during my own development time- but understanding the core foundations of RE, alongside of self-studying and learning x64 Assembly helped make me keep things a bit simpler, and also helped me look at logical operations from a

simplistic angle. This is probably due to the amount of logical work and breaking down you do.

There are some other things that RE will help you with, however, I find that it will depend on person to person, that above is just from my personal experience.

RE & The Learning Curve

I would like to warn for newcomers into reverse engineering who do not have the best background in tech. Reverse engineering has a steep learning curve- there are some web experts I know that seriously would not even comprehend where exactly to start with RE. However, it is often, overcomplicated. I say this a lot, people make RE out to be this big, scary, extremely complex field- and while it does involve a lot of technical knowledge, expertise, and experience to be considered *'good'* in it, I say that it simply looks scary *more than it actually is scary*.

Some of the concepts would be considerably noted as complex, and I agree with that, but making it out to be this impossible, highly skilled field is something I have seen way too much in *'beginner'* cybersecurity servers and communities. Another thing I will be doing throughout this series is attempting to make everything as digestible as possible, an actual learning material, not this showboat to be like "*yuhhh im a god at RE look at me writing all this complex shit*" like, I understand the pain T_T.

Moving forward from that, I highly recommend that you do NOT rush through RE. It is a pretty helpful field for anyone in cybersecurity who wants to go deep and teaches a lot about systems, system designs, technologies, etc.

Do not give up either- it may not click at first, but even I can say that wanting to leave the field because you do not understand it is a shitty feeling. It sucks and deeply knocks you

down but keep on pushing- because the end reward is so worth it.

What Should You Know for RE?

I honestly always tell people that it really depends on the field of RE and where you are going- but for all of it, I always recommend that people know the following:

- **Computer Science Fundamentals –** This includes computer memory, operating systems, binary numbering systems, compilers, Boolean algebra, etc. The usual CompSci with a mix of binary analysis and networking.

- **Development Fundamentals –** For reverse engineering, I would recommend C. I know it's the basic bitch everyone runs to, but at the same time C teaches users, especially beginner programmers how to respect a system, and gets you to

deal with the raw handed portions of a system and application development. It is because of this that it mixes well with the topics that you will often apply reverse engineering to. Such as Game Hacking & Exploit Development, Malware Development, and other fields alike.

- **Program Debugging & Testing:** This is another useful skill. Learning how to debug programs, run them dynamically, properly isolating them, and properly testing the application for edge cases is incredibly important to learn. This will help you poke and prod with programs dynamically when doing more particular software reverse engineering.

- **Application Engineering:** This is NOT by any means NEEDED, but I tossed it in here because it helps in general. Learning how programs are engineered, the standard processing for engineering applications, or just

understanding general development even when it comes to firmware may assist you *depending on what you are targeting*. Additionally with the