Code Reverse Engineering

Yeah, reviewing a ebook **Code Reverse Engineering** could grow your close associates listings. This is just one of the solutions for you to be successful. As understood, carrying out does not recommend that you have fabulous points.

Comprehending as without difficulty as concord even more than new will come up with the money for each success. neighboring to, the publication as skillfully as sharpness of this Code Reverse Engineering can be taken as competently as picked to act.

Covert Java Springer
More practical less theory KEY FEATURES ? In-depth practical demonstration with multiple examples of reverse engineering concepts. ? Provides a step-by-step approach to reverse engineering, including assembly instructions. ? Helps security researchers to crack application code and logic using reverse engineering open source tools. ? Reverse engineering strategies for simple-to-complex applications like Wannacry ransomware and Windows calculator. DESCRIPTION The book 'Implementing Reverse Engineering' begins with a step-by-step explanation of the fundamentals of reverse engineering. You will learn how to use reverse engineering to find bugs and hacks in real-world applications. This book is divided into three sections. The first section is an exploration of the reverse engineering process. The second section explains reverse engineering of applications, and the third section is a collection of real-world use-cases with solutions. The first section introduces the basic concepts of a computing system and the data building blocks of the computing system. This section also includes open-source tools such as CFF Explorer, Ghidra, Cutter, and x32dbg. The second section goes over various reverse engineering practicals on various applications to give users hands-on experience. In the third section, reverse engineering of Wannacry ransomware, a well-known Windows application, and various exercises are demonstrated step by step. In a very detailed and step-by-step manner, you will practice and understand different assembly instructions, types of code calling conventions, assembly patterns of applications with the printf function, pointers, array, structure, scanf, strcpy function, decision, and loop control structures. You will learn how to use open-source tools for reverse engineering such as portable executable editors, disassemblers, and debuggers. WHAT YOU WILL LEARN ? Understand

different code calling conventions like CDECL, STDCALL, and FASTCALL with practical illustrations. ? Analyze and break WannaCry ransomware using Ghidra. ? Using Cutter, reconstruct application logic from the assembly code. ? Hack the Windows calculator to modify its behavior. WHO THIS BOOK IS FOR This book is for cybersecurity researchers, bug bounty hunters, software developers, software testers, and software quality assurance experts who want to perform reverse engineering for advanced security from attacks. Interested readers can also be from high schools or universities (with a Computer Science background). Basic programming knowledge is helpful but not required. TABLE OF CONTENTS 1. Impact of Reverse Engineering 2. Understanding Architecture of x86 machines 3. Up and Running with Reverse Engineering tools 4. Walkthrough on Assembly Instructions 5. Types of Code Calling Conventions 6. Reverse Engineering Pattern of Basic Code 7. Reverse Engineering Pattern of the printf() Program 8. Reverse Engineering Pattern of the Pointer Program 9. Reverse Engineering Pattern of the Decision Control Structure 10. Reverse Engineering Pattern of the Loop Control Structure 11. Array Code Pattern in Reverse Engineering 12. Structure Code Pattern in Reverse Engineering 13. Scanf Program Pattern in Reverse Engineering 14. strcpy Program Pattern in Reverse Engineering 15. Simple Interest Code Pattern in Reverse Engineering 16. Breaking Wannacry Ransomware with Reverse Engineering 17. Generate Pseudo Code from the Binary File 18. Fun with Windows Calculator Using Reverse Engineering

Gray Hat Python No Starch Press

Decompiling Android looks at the the reason why Android apps can be decompiled to recover their source code, what it means to Android developers and how you can protect your code from prying eyes. This is also a good way to see how good and bad Android apps are constructed and how to learn from them in building your own apps. This is becoming an increasingly important topic as the Android marketplace grows and developers are unwittingly releasing the apps with lots of back doors allowing people to potentially obtain credit card information and database logins to back-end systems, as they don't realize how easy it is to decompile their Android code. In depth examination of the Java and Android class file structures Tools and techniques for decompiling Android apps Tools and techniques for protecting your Android apps

Exploiting Software: How To Break Code Packt Publishing Ltd

Mobile software development is evolving rapidly. Software development includes computer programing, documenting, testing and bug fixing processes. These processes need a detail understanding of the application logic which often requires reverse-engineering their artifacts. My thesis identifies and addresses the following three problems in mobile software development, specifically in program understanding and reverse-engineering for mobile application development. (1) There is no graphical on-phone debugger. (2) The second problem is that mobile software programmers have to manually re-implement the conceptual screen drawings or sketches of graphical artists in code, which is cumbersome and

expensive. (3) Companies try to ”go mobile” (by developing mobile apps). To do that understanding the high level business of their current legacy software systems is necessary but challenging. To address these three challenges, this dissertation introduces the following three innovations. (1) GROPG is the first graphical on-phone debugger. GROPG makes debugging mobile apps more convenient and productive than existing textbased on-phone debuggers. (2) REMAUI is a mobile digital screenshot and sketch reverse-engineering tool. REMAUI makes developing mobile user interface code easier. (3) RengLaDom is a legacy application reverse-engineering tool. RengLaDom can infer domain concepts from legacy source code. Specifically, (1) debugging mobile phone applications is hard, as current debugging techniques either require multiple computing devices or do not support graphical debugging. To address this problem we present GROPG, the first graphical on-phone debugger. We implement GROPG for Android and perform a preliminary evaluation on third-party applications. Our experiments suggest that GROPG can lower the overall debugging time of a comparable text-based on-phone debugger by up to 2/3. (2) Second, when developing the user interface code of a mobile application, a big gap exists between the sketches and digital conceptual drawings of graphic artists and working user interface code. Currently, programmers bridge this gap manually, by re-implementing the sketches and drawings in code, which is cumbersome and expensive. To bridge this gap, this dissertation introduces the first technique to automatically reverse engineer mobile application user interfaces from UI sketches, digital conceptual drawings, or screenshots (REMAUI). In our experiments on third party inputs, REMAUI's inferred runtime user interface hierarchies closely resembled the user interface runtime UI hierarchies of the applications that produced REMAUI's inputs. Further, the resulting screenshots closely resembled REMAUI's inputs and overall runtime was below one minute. (3) Finally, a promising approach to understanding the business functions implemented by a large-scale legacy application is to reverse engineer the full application code with all its complications into a high-level abstraction such as a design document that can focus exclusively on important domain concepts. Although much progress has been made, we encountered the following two problems. (a) Existing techniques often cannot distinguish between code that carries interesting domain concepts and code that merely provides low-level implementation services. (b) For an evaluation, given that design documents are typically not maintained throughout program development, how can we judge if the domain model inferred by a given technique is of a high quality? We address these problems by re-examining the notion of domain models in object-oriented development and encoding our understanding in a novel lightweight reverse engineering technique that pinpoints those program classes that likely carry domain concepts. We implement our techniques in a RengLaDom prototype tool for Java and compare how close our inferred domain models are to existing domain models. Given the lack of traditional domain models, we propose to use for such evaluation existing object-relational data persistence mappings (ORM), which map program classes to a relational database schema. The original application engineers carefully designed such mappings, consider them valuable, and maintain them as part of the application. After manually removing such OR mappings from open-source applications, our RengLaDom technique was able to reverse engineer domain models that are much closer to the original ORM domain models than the models

produced by competing approaches, regardless of the particular ORM framework used. Additional experiments indicate that RengLaDom's ability to infer better domain models extends to a variety of non-ORM applications.

Reverse Engineering Packt Publishing Ltd

Python is fast becoming the programming language of choice for hackers, reverse engineers, and software testers because it's easy to write quickly, and it has the low-level support and libraries that make hackers happy. But until now, there has been no real manual on how to use Python for a variety of hacking tasks. You had to dig through forum posts and man pages, endlessly tweaking your own code to get everything working. Not anymore. Gray Hat Python explains the concepts behind hacking tools and techniques like debuggers, trojans, fuzzers, and emulators. But author Justin Seitz goes beyond theory, showing you how to harness existing Python-based security tools—and how to build your own when the pre-built ones won't cut it. You'll learn how to: – Automate tedious reversing and security tasks – Design and program your own debugger – Learn how to fuzz Windows drivers and create powerful fuzzers from scratch – Have fun with code and library injection, soft and hard hooking techniques, and other software trickery – Sniff secure traffic out of an encrypted web browser session – Use PyDBG, Immunity Debugger, Sulley, IDAPython, PyEMU, and more The world's best hackers are using Python to do their handiwork. Shouldn't you?

Handbook of Information and Communication Security No Starch Press

Object-Oriented Reengineering Patterns collects and distills successful techniques in planning a reengineering project, reverse-engineering, problem detection, migration strategies and software redesign. This book is made available under the Creative Commons Attribution-ShareAlike 3.0 license. You can either download the PDF for free, or you can buy a softcover copy from lulu.com. Additional material is available from the book's web page at http://scg.unibe.ch/oorp

Code Reading Springer Science & Business Media

Current CASE technology provides sophisticated diagramming tools to generate a software design. The design, stored internal to the CASE tool, is bridged to the code via code generators. There are several limitations to this technique: (1) the portability of the design is limited to the portability of the CASE tools, and (2) the code generators offer a clumsy link between design and code. The CASE tool though valuable during design, becomes a hindrance during implementation. Frustration frequently causes the CASE tool to be abandoned during implementation, permanently severing the link between design and code. Current CASE stores the design in a CASE internal structure, from which code is generated. The technique presented herein suggests that CASE tools store the system knowledge directly in code. The CASE support then switches from an emphasis on code generators to employing state-of-the-art reverse engineering techniques for

document generation. Graphical and textual descriptions of each software component (e.g., Ada Package) may be generated via reverse engineering techniques from the code. These reverse engineered descriptions can be merged with system over-view diagrams to form a top-level design document. The resulting document can readily reflect changes to the software components by automatically generating new component descriptions for the changed components. The proposed auto documentation technique facilitates the document upgrade task at later stages of development, (e.g., design, implementation and delivery) by using the component code as the source of the component descriptions. The CASE technique presented herein is a unique application of reverse engineering techniques to new software systems. This technique contrasts with more traditional CASE auto code generation techniques.

Object-oriented Reengineering Patterns Springer Science & Business Media

No source code? No problem. With IDA Pro, the interactive disassembler, you live in a source code-optional world. IDA can automatically analyze the millions of opcodes that make up an executable and present you with a disassembly. But at that point, your work is just beginning. With The IDA Pro Book, you'll learn how to turn that mountain of mnemonics into something you can actually use. Hailed by the creator of IDA Pro as "profound, comprehensive, and accurate," the second edition of The IDA Pro Book covers everything from the very first steps to advanced automation techniques. You'll find complete coverage of IDA's new Qt-based user interface, as well as increased coverage of the IDA debugger, the Bochs debugger, and IDA scripting (especially using IDAPython). But because humans are still smarter than computers, you'll even learn how to use IDA's latest interactive and scriptable interfaces to your advantage. Save time and effort as you learn to: –Navigate, comment, and modify disassembly –Identify known library routines, so you can focus your analysis on other areas of the code –Use code graphing to quickly make sense of cross references and function calls –Extend IDA to support new processors and filetypes using the SDK –Explore popular plug-ins that make writing IDA scripts easier, allow collaborative reverse engineering, and much more –Use IDA's built-in debugger to tackle hostile and obfuscated code Whether you're analyzing malware, conducting vulnerability research, or reverse engineering software, a mastery of IDA is crucial to

your success. Take your skills to the next level with this 2nd edition of The IDA Pro Book.

Reverse Engineering Sams Publishing

This book answers two central questions: firstly, is it at all possible to verify electronic equipment procured from untrusted vendors? Secondly, can I build trust into my products in such a way that I support verification by untrusting customers? In separate chapters the book takes readers through the state of the art in fields of computer science that can shed light on these questions. In a concluding chapter it discusses realistic ways forward. In discussions on cyber security, there is a tacit assumption that the manufacturer of equipment will collaborate with the user of the equipment to stop third-party wrongdoers. The Snowden files and recent deliberations on the use of Chinese equipment in the critical infrastructures of western countries have changed this. The discourse in both cases revolves around what malevolent manufacturers can do to harm their own customers, and the importance of the matter is on par with questions of national security.This book is of great interest to ICT and security professionals who need a clear understanding of the two questions posed in the subtitle, and to decision-makers in industry, national bodies and nation states. This work was published by Saint Philip Street Press pursuant to a Creative Commons license permitting commercial use. All rights not granted by the work's license are retained by the author or authors.

The Ghidra Book Addison-Wesley Professional

The book is logically divided into 5 main categories with each category representing a major skill set required by most security professionals: 1. Coding – The ability to program and script is quickly becoming a mainstream requirement for just about everyone in the security industry. This section covers the basics in coding complemented with a slue of programming tips and tricks in C/C++, Java, Perl and NASL. 2. Sockets – The technology that allows programs and scripts to communicate over a network is sockets. Even though the theory remains the same – communication over TCP and UDP, sockets are implemented differently in nearly ever language. 3. Shellcode – Shellcode, commonly defined as bytecode converted from Assembly, is utilized to execute commands on remote systems via direct memory access. 4. Porting – Due to the differences between operating platforms and language implementations on those platforms, it is a common practice to modify an original body of code to work on a different platforms. This technique is known as porting and is incredible useful in the real world environments since it allows you to not “recreate the wheel. 5. Coding Tools – The culmination of the previous four

sections, coding tools brings all of the techniques that you have learned to the forefront. With the background technologies and techniques you will now be able to code quick utilities that will not only make you more productive, they will arm you with an extremely valuable skill that will remain with you as long as you make the proper time and effort dedications. *Contains never before seen chapters on writing and automating exploits on windows systems with all-new exploits. *Perform zero-day exploit forensics by reverse engineering malicious code. *Provides working code and scripts in all of the most common programming languages for readers to use TODAY to defend their networks.

The Huawei and Snowden Questions Pearson Education India

Learn to find software bugs faster and discover how other developers have solved similar problems. For intermediate to advanced iOS/macOS developers already familiar with either Swift or Objective-C who want to take their debugging skills to the next level, this book includes topics such as: LLDB and its subcommands and options; low-level components used to extract information from a program; LLDB's Python module; and DTrace and how to write D scripts.

Reversing BPB Publications

The process of software reverse engineering and malware analysis often comprise a combination of static and dynamic analyses. The successful outcome of each step is tightly coupled with the functionalities of the tools and skills of the reverse engineer. Even though automated tools are available for dynamic analysis, the static analysis process is a fastidious and time-consuming task as it requires manual work and strong expertise in assembly coding. In order to enhance and accelerate the reverse engineering process, we introduce a new dimension known as clone-based analysis. Recently, binary clone matching has been studied with a focus on detecting assembly (binary) clones. An alternative approach in clone analysis, which is studied in the present research, is concerned with assembly to source code matching. There are two major advantages in considering this extra dimension. The first advantage is to avoid dealing with low-level assembly code in situations where the corresponding high-level code is available. The other advantage is to prevent reverse engineering parts of the software that have been analyzed before. The clone-based analysis can be helpful in significantly reducing the required time and improving the accuracy of static analysis. In this research, we elaborate a framework for assembly to open-source code matching. Two types of analyses are provided by the framework, namely online and offline. The online analysis process triggers queries to online source code repositories based on extracted features from the functions at the assembly level. The result is the matched set of references to the open-source project files with similar features. Moreover, the offline analysis assigns functionality tags and provides in-depth information regarding the potential functionality of a portion of the assembly file. It reports on function stack frames, prototypes,

arguments, variables, return values and low-level system calls. Besides, the offline analysis is based on a built-in dictionary of common user-level and kernel-level API functions that are used by malware to interact with the operating system. These functions are called for performing tasks such as file I/O, network communications, registry modification, and service manipulation. The offline analysis process has been expanded through an incremental learning mechanism which results in an improved detection of crypto-related functions in the disassembly. The other developed extension is a customized local code repository which performs automated source code parsing, feature extraction, and dataset generation for code matching. We apply the framework in several reverse engineering and malware analysis scenarios. Also, we show that the underlying tools and techniques are effective in providing additional insights into the functionality, inner workings, and components of the target binaries.

A Proposal for Reverse Engineering CASE Tools to Support New Software Development Elsevier

Attacks take place everyday with computers connected to the internet, because of worms, viruses or due to vulnerable software. These attacks result in a loss of millions of dollars to businesses across the world. Identifying Malicious Code through Reverse Engineering provides information on reverse engineering and concepts that can be used to identify the malicious patterns in vulnerable software. The malicious patterns are used to develop signatures to prevent vulnerability and block worms or viruses. This book also includes the latest exploits through various case studies. Identifying Malicious Code through Reverse Engineering is designed for professionals composed of practitioners and researchers writing signatures to prevent virus and software vulnerabilities. This book is also suitable for advanced-level students in computer science and engineering studying information security, as a secondary textbook or reference.

Microsoft Office Visio 2007 Inside Out Springer Science & Business Media

Beginning with a basic primer on reverse engineering-including computer internals, operating systems, and assembly language-and then discussing the various applications of reverse engineering, this book provides readers with practical, in-depth techniques for software reverse engineering. The book is broken into two parts, the first deals with security-related reverse engineering and the second explores the more practical aspects of reverse engineering. In addition, the author explains how to reverse engineer a third-party software library to improve interfacing and how to reverse engineer a competitor's software to build a better product. * The first popular book to show

how software reverse engineering can help defend against security threats, speed up development, and unlock the secrets of competitive products * Helps developers plug security holes by demonstrating how hackers exploit reverse engineering techniques to crack copy-protection schemes and identify software targets for viruses and other malware * Offers a primer on advanced reverse-engineering, delving into "disassembly"-code-level reverse engineering-and explaining how to decipher assembly language

Implementing Reverse Engineering John Wiley & Sons

Reverse Engineering Code with IDA ProElsevier

Ghidra Software Reverse Engineering for Beginners

"O'Reilly Media, Inc."

Implement reverse engineering techniques to analyze software, exploit software targets, and defend against security threats like malware and viruses. Key FeaturesAnalyze and improvise software and hardware with real-world examplesLearn advanced debugging and patching techniques with tools such as IDA Pro, x86dbg, and Radare2.Explore modern security techniques to identify, exploit, and avoid cyber threatsBook Description If you want to analyze software in order to exploit its weaknesses and strengthen its defenses, then you should explore reverse engineering. Reverse Engineering is a hackerfriendly tool used to expose security flaws and questionable privacy practices.In this book, you will learn how to analyse software even without having access to its source code or design documents. You will start off by learning the low-level language used to communicate with the computer and then move on to covering reverse engineering techniques. Next, you will explore analysis techniques using real-world tools such as IDA Pro and x86dbg. As you progress through the chapters, you will walk through use cases encountered in reverse engineering, such as encryption and compression, used to obfuscate code, and how to to identify and overcome anti-debugging and anti-analysis tricks. Lastly, you will learn how to analyse other types of files that contain code. By the end of this book, you will have the confidence to perform reverse engineering. What you will learnLearn core reverse engineeringIdentify and extract malware componentsExplore the tools used for reverse engineeringRun programs under non-native operating systemsUnderstand binary obfuscation techniquesIdentify and analyze anti-debugging and anti-analysis tricksWho this book is for If you are a security engineer or analyst or a system programmer and want to use reverse engineering to improve your software and hardware, this is the book for you. You will also find this book useful if you are a developer who wants to explore and learn reverse engineering. Having some programming/shell scripting knowledge is an added advantage.

Mastering Reverse Engineering Elsevier

As a Java developer, you may find

yourself in a situation where you have to maintain someone else's code or use a third-party's library for your own application without documentation of the original source code. Rather than spend hours feeling like you want to bang your head against the wall, turn to "Covert Java: Techniques for Decompiling, Patching, and Reverse Engineering." These techniques will show you how to better understand and work with third-party applications. Each chapter focuses on a technique to solve a specific problem, such as obfuscation in code or scalability vulnerabilities, outlining the issue and demonstrating possible solutions. Summaries at the end of each chapter will help you double check that you understood the crucial points of each lesson. You will also be able to download all code examples and sample applications for future reference from the publisher's website. Let "Covert Java" help you crack open mysterious codes!

Reverse Engineering Code with IDA Pro

A guide to using the Ghidra software reverse engineering tool suite. The result of more than a decade of research and development within the NSA, the Ghidra platform was developed to address some of the agency's most challenging reverse-engineering problems. With the open-source release of this formerly restricted tool suite, one of the world's most capable disassemblers and intuitive decompilers is now in the hands of cybersecurity defenders everywhere -- and The Ghidra Book is the one and only guide you need to master it. In addition to discussing RE techniques useful in analyzing software and malware of all kinds, the book thoroughly introduces Ghidra's components, features, and unique capacity for group collaboration. You'll learn how to: • Navigate a disassembly • Use Ghidra's built-in decompiler to expedite analysis • Analyze obfuscated binaries • Extend Ghidra to recognize new data types • Build new Ghidra analyzers and loaders • Add support for new processors and instruction sets • Script Ghidra tasks to automate workflows • Set up and use a collaborative reverse engineering environment Designed for beginner and advanced users alike, The Ghidra Book will effectively prepare you to meet the needs and challenges of RE, so you can analyze files like a pro.

Sockets, Shellcode, Porting, and Coding: Reverse Engineering Exploits and Tool Coding for Security Professionals No Starch Press

If you want to master the art and science of reverse engineering code with IDA Pro for security R&D or software debugging, this is the book for you. Highly organized and sophisticated criminal entities are constantly developing more complex, obfuscated, and armored viruses, worms, Trojans, and botnets. IDA Pro’s interactive interface and programmable development language provide you with complete control over code disassembly and debugging. This is the only book which focuses exclusively on the world’s most powerful and popular took for reverse engineering code.

*Reverse Engineer REAL Hostile Code To follow along with this chapter, you must download a file called !DANGER!INFECTEDMALWARE!DANGER!... ‘nuff said. *Portable Executable (PE) and Executable and Linking Formats (ELF) Understand the physical layout of PE and ELF files, and analyze the components that are essential to reverse engineering. *Break Hostile Code Armor and Write your own Exploits Understand execution flow, trace functions, recover hard coded passwords, find vulnerable functions, backtrace execution, and craft a buffer overflow. *Master Debugging Debug in IDA Pro, use a debugger while reverse engineering, perform heap and stack access modification, and use other debuggers. *Stop Anti-Reversing Anti-reversing, like reverse engineering or coding in assembly, is an art form. The trick of course is to try to stop the person reversing the application. Find out how! *Track a Protocol through a Binary and Recover its Message Structure Trace execution flow from a read event, determine the structure of a protocol, determine if the protocol has any undocumented messages, and use IDA Pro to determine the functions that process a particular message. *Develop IDA Scripts and Plug-ins Learn the basics of IDA scripting and syntax, and write IDC scripts and plug-ins to automate even the most complex tasks.

Beyond the Code Oxford University Press

Malware analysis is big business, and attacks can cost a company dearly. When malware breaches your defenses, you need to act quickly to cure current infections and prevent future ones from occurring. For those who want to stay ahead of the latest malware, Practical Malware Analysis will teach you the tools and techniques used by professional analysts. With this book as your guide, you'll be able to safely analyze, debug, and disassemble any malicious software that comes your way. You'll learn how to: –Set up a safe virtual environment to analyze malware –Quickly extract network signatures and host-based indicators –Use key analysis tools like IDA Pro, OllyDbg, and WinDbg –Overcome malware tricks like obfuscation, anti-disassembly, anti-debugging, and anti-virtual machine techniques –Use your newfound knowledge of Windows internals for malware analysis –Develop a methodology for unpacking malware and get practical experience with five of the most popular packers –Analyze special cases of malware with shellcode, C++, and 64-bit code Hands-on labs throughout the book challenge you to practice and synthesize your skills as you dissect real malware samples, and pages of detailed dissections offer an over-the-shoulder look at how the pros do it. You'll learn how to crack open malware to see how it really works, determine what damage it has done, thoroughly clean your network, and ensure that the malware never comes back. Malware analysis is a cat-and-mouse game with rules that

are constantly changing, so make sure you have the fundamentals. Whether you're tasked with securing one network or a thousand networks, or you're making a living as a malware analyst, you'll find what you need to succeed in Practical Malware Analysis.

Practical Reverse Engineering

Apress

CD-ROM contains cross-referenced code.